

PDP-11-70

11/70 MEM MGMT
CEKBEE0

AH-7976E-MC
FICHE 1 OF 2

SEP 1980
COPYRIGHT © 75 80
MADE IN USA



The main body of the document is a dense grid of approximately 15 columns and 25 rows of small, faint tables. Each cell in the grid contains a miniature version of the data presented in the header, likely representing a memory management table for the PDP-11-70 system. The text is too small to be legible in detail, but the overall structure is a regular array of data blocks.

PDP-11-70

11/70 MEM MGMT
CEKBEE0

AH 7976E-MC
FICHE 2 OF 2

SEP 1980
COPYRIGHT © 75 80
MADE IN USA



[Faint, illegible text visible on the left side of the page, likely bleed-through from the reverse side.]



IDENTIFICATION

B 1

SEQ 0001

PRODUCT CODE: AC-7975E-MC
PRODUCT NAME: CEKBEE0 11/70 MEM MGMT
DATE CREATED: MAY, 1980
MAINTAINER: DIAGNOSTIC ENGINEERING

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

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

COPYRIGHT (C) 1975,1980 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	DECX/11

TABLE OF CONTENTS

- 1) ABSTRACT
- 2) REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 STORAGE
 - 2.3 PRELIMINARY PROGRAMS
- 3) LOADING PROCEDURE
 - 3.1 METHOD
- 4) STARTING PROCEDURE
 - 4.1 STARTING ADDRESSES
 - 4.2 PROGRAM AND OPERATOR ACTION
 - 4.3 SPECIAL STARTING PROCEDURE
- 5) OPERATING PROCEDURE
 - 5.1 OPERATIONAL SWITCH SETTINGS
 - 5.2 SUB-ROUTINE ABSTRACTS
 - 5.3 PROGRAM AND OPERATOR ACTION
- 6) ERRORS
 - 6.1 ERROR HALTS AND DESCRIPTION
 - 6.2 ERROR RECOVERY
 - 6.3 SAMPLE ERROR MESSAGES
- 7) RESTRICTIONS
 - 7.1 STARTING RESTRICTIONS
 - 7.2 OPERATING RESTRICTIONS
- 8) MISCELLANEOUS
 - 8.1 EXECUTION TIME
 - 8.2 SOME IDEAS ON HOW TO GET MORE FROM THE PROGRAM
 - 8.3 ROM STATE DESCRIPTIONS
- 9) REVISION HISTORY

ABSTRACT

THIS PROGRAM WILL TEST ALL OF THE MEMORY MANAGEMENT LOGIC AND ENABLE THE FIELD SERVICE REPRESENTATIVE TO ISOLATE THE DETECTED FAILURES TO A REPLACABLE MODULE. IT IS ASSUMED THAT BOTH THE CPU AND THE CACHE HAVE BEEN TESTED, OR ARE KNOWN TO BE FUNCTIONING CORRECTLY, AND THAT THE PROGRAM IS STARTED FROM ADDRESS 200. THIS WILL PROVIDE THE EARLIEST DETECTION OF MEMORY MANAGEMENT RELATED ERRORS AND ENABLE LOOPING ON THE ERROR INVOLVING MINIMUM LOGIC. THIS PROGRAM MAY ALSO EXPOSE FAULTS THAT ARE ON THE INTERFACE BETWEEN MEMORY MANAGEMENT AND OTHER SECTIONS OF THE COMPUTER.

THIS PROGRAM HAS BEEN SEGMENTED IN THE FOLLOWING WAY: ALL DATA TABLES, ERROR MESSAGES, AND SUBROUTINES RESIDE IN LOW CORE (VIRTUAL PAGES 0 & 1 IE. ADDRESSES 001100 THRU 037776). RIGHT NOW THE END OF THE SUBROUTINES IS AROUND 030300, SO THERE IS SOME ROOM FOR FUTURE EXPANSION. THE TEST CODE STARTS AT VIRTUAL PAGE 2 (ADDRESS 040000) AND EXPANDS TOWARD PAGE 4 (ADDRESS 100000). THE END OF THE PROGRAM IS NOW AROUND ADDRESS 077600, SO SMALL MODIFICATIONS CAN BE MADE WITHOUT RE-SEGMENTING THE PROGRAM.

THE REASON FOR THIS SEGMENTATION IS TWO-FOLD, FIRST IT ENABLES THE OPERATOR TO TELL FROM THE ADDRESS LIGHTS EXACTLY WHERE THE PROGRAM HAS HALTED OR 'HUNG-UP'. THAT IS, DID IT HALT IN THE ERROR ROUTINE OR IN A TRAP ROUTINE BECAUSE OF A CONDITION IMPOSSIBLE TO RECOVER FROM (ON PAGE 0 OR 1), OR DID IT GET 'HUNG-UP' IN THE TEST CODE ON PAGE 2 OR 3. THE OTHER REASON IS THAT CERTAIN MEMORY MANAGEMENT FUNCTIONS LOCK UP THE VIRTUAL PC OF THE INSTRUCTION AND THE PROGRAM, IN ORDER TO OPERATE PROPERLY, MUST KNOW WHERE IT IS AT ALL TIMES. IT SEEMS MUCH SIMPLER FOR THE CODE TO START AT A PREDETERMINED BOUNDARY SO THAT IF THE MESSAGES CHANGE OR A NEW SUBROUTINE IS ADDED THE PAGE THAT THE CODE IS ON WILL REMAIN THE SAME.

EACH TEST WILL SET THE LOOP ON ERROR POINTER (\$LPERR) TO THE MINIMUM NECESSARY SETUP CODE, IF ANY, FOR THE FUNCTION UNDER TEST. A SYNCHRONIZATION INSTRUCTION (NOP) IS PROVIDED BEFORE THE INSTRUCTION(S) THAT TEST(S) EACH NEW FUNCTION. THIS WILL ENABLE THE FIELD SERVICE REPRESENTATIVE TO UTILIZE THE MICRO BREAK REGISTER TO GENERATE AN 'EXTERNAL SYNC' PULSE ON THE BACK PLANE FOR BETTER PULSE RESOLUTION.

SECTION 8.2 OF THIS DOCUMENT CONTAINS SOME IDEAS THAT I HAD WHEN I WAS WRITING THIS PROGRAM ON HOW TO EFFECTIVELY UTILIZE IT TO MAKE FAULT ISOLATION EASIER. IF THESE IDEAS ARE NOT CORRECT OR NEED TO BE EXPANDED TO PROVIDE MORE INFORMATION PLEASE WRITE DOWN YOUR SUGGESTIONS AND FORWARD THEM TO THE DIAGNOSTIC DEPARTMENT.

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

CONSOLE OR THE CONSOLE CABLES THAT PLUG INTO THE MEMORY MANAGEMENT BOARDS. THE PROGRAM ASSUMES THAT THOSE COMPONENTS HAVE BEEN TESTED OR ARE KNOWN TO BE GOOD.

THIS DIAGNOSTIC SUPPORTS THE KB11-B/C, AND KB11-CM PROCESSORS.

2. REQUIREMENTS

2.1 EQUIPMENT
THE BASIC PDP-11/70 COMPUTER, INCLUDING AN OPERATING CPU, CACHE, AND MEMORY. AN LA-30 OR EQUIVALENT DEVICE IS ALSO NEEDED FOR ERROR MESSAGES, AND END OF PASS REPORTS.

NOTE: THIS DIAGNOSTIC SUPPORTS THE PDP-11/74, AN EXPERIMENTAL, IN-HOUSE PROCESSOR.

2.2 STORAGE
THIS PROGRAM REQUIRES 16K OF MEMORY TO LOAD AND AT LEAST 20K OF MEMORY TO RUN IN. IT WILL SCAN MEMORY FROM 16K TO 124K ON 2K BOUNDARIES, AND FROM 120K TO THE TOP OF MEMORY FOUND BY THE SIZE ROUTINE ON 8K BOUNDARIES.

2.3 PRELIMINARY PROGRAMS
THE CPU AND CACHE DIAGNOSTICS SHOULD BE RUN BEFORE THIS PROGRAM. MAIN MEMORY SHOULD BE SCANNED FOR AT LEAST THE FIRST 28K TO SEE THAT A PROGRAM WILL EXECUTE CORRECTLY BEFORE ANY PROGRAM IS RUN.

3. LOADING PROCEDURE

3.1 METHOD
THIS PROGRAM CAN BE LOADED FROM ANY DEVICE THAT IS SUPPORTED BY XXDP, AND SHOULD BE LOADED USING THE XXDP PROCEDURE FOR THAT DEVICE.

4. STARTING PROCEDURE

4.1 STARTING ADDRESSES
200 THIS ADDRESS WILL RUN THE COMPLETE PROGRAM
204 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 2
TEST THE READ/WRITE BITS IN THE MEMORY STATUS REGISTERS
210 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 3
PAGE ADDRESS AND PAGE DESCRIPTOR TESTS
214 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 4
RELOCATION AND ADDER TESTS
220 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 5
MEMORY MANAGEMENT ABORTS AND TRAPS LOGIC TESTS
224 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 6
D-SPACE TESTS, CORRECT TIMING OF I & D SPACE
230 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 7
A & W BIT LOGIC TEST AND DUAL MAPPING TESTS
234 THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 8
MOVE FROM AND MOVE TO PVIOUS MODE INSTRUCTION TESTS

- 4.2 PROGRAM AND OPERATOR ACTION
AFTER THE PROGRAM IS LOADED, THE FIRST TIME IT IS RUN IT WILL IDENTIFY ITSELF AND RUN A QUICK VERIFY PASS. AT THE END OF EACH PASS THE PROGRAM WILL TYPE OUT THE PASS NUMBER AND THE TOTAL NUMBER OF ERRORS FOUND ON THAT PASS.
- 4.3 SPECIAL STARTING PROCEDURE
IF IT APPEARS THAT THE CACHE IS CAUSING SOME TROUBLE AND YOU STILL WANT TO RUN THIS PROGRAM, IT IS POSSIBLE TO RUN THIS PROGRAM WITH THE CACHE DISABLED. SIMPLY LOAD THE CACHE CONTROL REGISTER (17777746) WITH THE DESIRED NUMBER, THEN LOAD THE PC (17777707) WITH THE STARTING ADDRESS AND PRESS CONTINUE. THE PROGRAM WILL NOW RUN AS IF YOU HAD LOADED THE STARTING ADDRESS AND PRESSED START BUT NOW THE CACHE CONTROL REGISTER IS DISABLING THE CACHE.
- BIT00 -DISABLE TRAPS
 - BIT01 -DISABLE UNIBUS TRAPS
 - BIT02 -FORCE MISS ON READ GROUP 0
 - BIT03 -FORCE MISS ON READ GROUP 1
 - BIT04 -FORCE REPLACE GROUP 0
 - BIT05 -FORCE REPLACE GROUP 1

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

SW15 1= HALT ON ERROR
 SW14 1= LOOP ON THE TEST THAT YOU ARE IN
 SW13 1= INHIBIT ALL ERROR TYPE OUTS
 SW12 1= INHIBIT TRACE TRAP ON EVERY OTHER PASS
 SW11 1= INHIBIT ITERATIONS AFTER FIRST PASS
 SW10 1= RING BELL ON ERROR
 SW09 1= LOOP ON ERROR
 SW08 1= LOOP ON TEST IN SWR<06:00>
 SW07 1= INHIBIT MULTIPLE ERROR TYPE OUTS

5.2 SUBROUTINE ABSTRACTS

ALL SUBROUTINE ABSTRACTS APPEAR IN THE CODE, BEFORE THEIR EXPANSION, AND IN THE DOCUMENT THAT IMMEDIATELY FOLLOWS THIS. THE FOLLOWING IS A LIST OF THEIR TITLES.

5.2.1 MACRO LIBRARY SUBROUTINES (FOUND IN MOST PROGRAMS)

END OF PASS ROUTINE
 SCOPE HANDLER ROUTINE
 ERROR HANDLER ROUTINE
 ERROR MESSAGE TYPE OUT ROUTINE
 CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS
 SAVE & RESTORE R0-R5 ROUTINES
 TYPE ROUTINE
 BINARY TO OCTAL (ASCII) AND TYPE ROUTINE
 CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
 TRAP DECODER
 POWER DOWN AND UP ROUTINE
 DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

5.2.2 SUBROUTINES UNIQUE TO THIS PROGRAM

TURN OFF AND SAVE T-BIT
 RESTORE T-BIT TO ITS PREVIOUS CONDITION
 CLEAR 16 PAR'S OR PDR'S STARTING FROM ADDRESS IN R5
 CLEANUP LOCATIONS THAT HOLD LOGICAL 'AND' AND 'OR'
 P.A.R. OR P.D.R. ADDRESS TIMED OUT WHEN REFERENCED
 DUAL ADDRESSING WHEN LOADING A P.A.R. OR P.D.R.
 COUNT PATTERN ERROR IN P.A.R.'S OR P.D.R.'S

5.2.3 TRAP AND ABORT HANDLER ROUTINES

CPU TRAP HANDLER
 CACHE TRAPS AND ABORTS HANDLER
 MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER
 TRAP ROUTINES FOR ABORT IN SUPERVISOR AND USER MODE

6. ERRORS

- 6.1 **ERROR HALTS AND DESCRIPTION**
 WHEN THE PROGRAM DETECTS AN ERROR CONDITION IT ISSUES AN 'ERROR' (EMT) CALL. THIS CAUSES THE CPU TO TRAP TO THE 'ERROR HANDLER ROUTINE' WHICH PRINTS OUT THE ERROR MESSAGE, IF ANY, AND CHECKS THE SWITCH REGISTER FOR THE MODE SELECTED. THE PROGRAM WILL REACT AS FOLLOWS:
- | | |
|----------------------------|------------------|
| HALT ON THE ERROR | IF SW15=1, IS UP |
| INHIBIT ERROR TYPE OUT | IF SW13=1, IS UP |
| RING BELL ON THE ERROR | IF SW10=1, IS UP |
| LOOP ON THE ERROR | IF SW09=1, IS UP |
| INHIBIT MULTIPLE TYPE OUTS | IF SW07=1, IS UP |
- 6.2 **ERROR RECOVERY**
 IF SWITCH 09 IS UP, THE PROGRAM WILL LOOP BACK TO WHERE THE LOOP ON ERROR POINTER (\$LPERR) IS SET. THIS WILL PROVIDE THE TIGHTEST POSSIBLE SCOPING LOOP, AND PROVIDES ALL NECESSARY SETUP CODE TO RECREATE THE ERROR. IF A SYNC POINT IS DESIRED TO EXTERNAL SYNC THE SCOPE JUST PRIOR TO THE FAILING OPERATION LOAD THE MICRO BREAK REGISTER WITH '044' AND USE THE 'NOP' PROVIDED.
- 6.3 **SAMPLE ERROR TYPE OUTS**
 CPU TRAP OR ABORT THRU 'ERRVEC' (004) HAD INCORRECT CONDITION
 EXPECTD RECEIVD TESTNO PC AT ABORT
 000040 000020 000047 XXXXXX

THIS ERROR MESSAGE INDICATES THAT THE CPU TIMED OUT OVER THE UNIBUS WHEN IT WAS EXPECTING A CACHE NON-EXISTANT MEMORY TRAP. THIS TEST IS CHECKING THE CARRY PROPAGATION, THAT IS DETERMINED FROM LOOKING AT THE INDEX AT THE FRONT OF THE LISTING.

7. RESTRICTIONS

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

7.2 OPERATING RESTRICTIONS
NONE

8. MISCELLANEOUS

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

8.2 HINTS ON HOW TO GET MORE INFORMATION FROM THE PROGRAM

IF AN ERROR OCCURS THE FIRST THING THAT SHOULD BE NOTED IS WHAT PASS DID THE ERROR OCCUR ON. IF THE PASS HAS AN EVEN NUMBER AND SWITCH 12 IS DOWN THEN THE ERROR MIGHT BE T-BIT SENSITIVE. TRY TO RUN AGAIN WITH SWITCH 12 UP, THIS WILL INHIBIT T-BIT TRAPPING.

IF THE PASS NUMBER IS GREATER THAN ONE THE ERROR MIGHT BE ITERATION SENSITIVE. TRY TO RUN AGAIN WITH SWITCH 11 UP, THIS WILL INHIBIT ITERATIONS.

NOW THAT YOU HAVE DETERMINED HOW TO MAKE THE MACHINE FAIL LOOK IN THE INDEX AT THE FRONT OF THE LISTING TO FIND THE TITLE OF THE TEST THAT WAS RUNNING WHEN THE ERROR CONDITION OCCURRED. GO TO THE LISTING AND READ THE PARAGRAPH AT THE BEGINNING OF THE TEST SO THAT YOU KNOW WHAT THE TEST IS TRYING TO DO. NOW READ THE ERROR MESSAGE AND IF THERE IS A COLUMN LABELED 'ERRORPC' GO TO THAT LOCATION IN THE TEST. THIS IS THE PC OF THE 'ERROR' STATEMENT. THE NUMBER IS THE ERROR MESSAGE NUMBER AND IN THE FRONT OF THE LISTING IS THE 'ERROR MESSAGE POINTER TABLE' WHICH WILL TELL YOU WHAT WORDS WERE TYPED OUT.

IF YOU WANT TO SCOPE THIS ERROR CONDITION, PUT UP SWITCH 09 (LOOP ON ERROR) OR IF YOU WANT TO LOOP ON THE ENTIRE TEST PUT UP SWITCH 14. YOU WILL PROBABLY WANT TO INHIBIT THE ERROR TYPE OUT AT THIS POINT, SWITCH 13 WILL DO THAT.

IF YOU NEED TO DO ACCURATE SCOPING AND NEED A GOOD SYNC POINT THEN MAKE SURE THAT THE MICRO BREAK REGISTER (1777770) HAS '044' IN IT. THIS WILL CAUSE A PULSE ON THE BACK PLANE AT PIN # AE1 (SLOT 10) WHENEVER A 'NOP' IS EXECUTED, AND THAT PULSE WILL MAKE AN EXCELLENT 'EXTERNAL SYNC' SIGNAL FOR YOUR SCOPE. THERE IS A 'NOP' JUST BEFORE EACH INSTRUCTION THAT TESTS A MEMORY MANAGEMENT FUNCTION FOR THE FIRST TIME, SO YOU SHOULD BE ABLE TO SCOPE ON ANY FAILURE THAT THIS PROGRAM CAN DETECT.

ROM STATE DESCRIPTIONS
THIS IS A LIST OF THE ROM OUTPUTS FROM THE MEMORY MANAGEMENT ROMS ON 'SSRA', WITH A SENTENCE OR TWO DESCRIBING THEIR MEANING AS IT RELATES TO MEMORY MANAGEMENT.

ROM OUT01 - ROM OUT03
THESE OUTPUTS ARE SENT TO MULTIPLEXERS AND REGISTERS ON 'SSRA' TO INDICATE A PARTICULAR MACHINE FUNCTION. THE ENCODING OF THESE OUTPUTS IS SHOWN IN A TRUTH TABLE ON 'SSRA'.

ROM OUT04
DESTINATION MODE -- THIS IS USED TO ENABLE RELOCATION IF BIT08 OF MMRO IS SET AND THIS IS THE DESTINATION CYCLE OF THE INSTRUCTION.

ROM OUT05
MFP + MTP -- THIS IS USED TO CLOCK THE PREVIOUS MODE INTO THE 'SPACE' FLIP-FLOPS ON 'SSRB', DURING A MFP + MTP INST.

ROM OUT06
TRAP OR ABORT -- THIS IS USED TO FORCE SETTING OF THE 'KERNEL SPACE' FLIP-FLOP ON 'SSRB' DURING A TRAP OR ABORT SEQUENCE.

ROM OUT07
BUST -- THIS IS USED TO CLOCK MOST OF THE LATCHES IN MEMORY MANAGEMENT SUCH AS: 'SPACE' F/F'S, STATUS REGISTERS,...

ROM OUT08
MFP + MTP -- THIS ASSERTS 'SSRB I SPACEB' TO FORCE I-SPACE ON MFPI + MTP I INSTRUCTIONS IF THE PSW IS NOT (USER MODE/ PREVIOUS USER MODE).

ROM OUT09
INST + INDEX FETCH -- THIS ASSERTS 'SSRB I SPACEA' WHICH FORCES I-SPACE DURING AN INSTRUCTION OR INDEX WORD FETCH.

ROM OUT10
THIS DOES NOT EXIST.

ROM OUT11
INST STARTED IN I-SPACE -- THIS ASSERTS 'SSRB I SPACEB' TO FORCE I-SPACE IF 'SSRB PREV=I' IS SET.

ROM OUT12
CONSOLE -- THIS ASSERTS 'SSRB I SPACEA' IF 'SSRK CNSL I SPACE H' IS TRUE AND YOU EXAMINE OR DEPOSIT.

ROM OUT13
SRCM = 1+2+3+4+5 -- IF THE SOURCE FIELD IS 7 THIS ASSERTS 'SSRB I SPACEB' WHICH FORCES I-SPACE.

ROM OUT14
DSTM = 1+2 -- IF THE DESTINATION FIELD IS 7 THIS ASSERTS 'SSRB I SPACEB' WHICH FORCES I-SPACE.

ROM OUT15
DSTM = 3 -- IF THE DESTINATION FIELD IS 7 THIS ASSERTS 'SSRB I SPACEA' WHICH FORCES I-SPACE DURING THE DESTINATION CYCLE.

ROM OUT16
FLOATING POINT -- IF THE DSTF = 7 AND THE DSTM = 2 THEN THIS ASSERTS 'SSRB I SPACEA' WHICH FORCES I-SPACE ON IMMEDIATE F.P.INST.

NOTE: THE FOLLOWING ROM STATES ARE NOT EXPLICITLY TESTED DUE TO THE FACT THAT I COULDN'T FIGURE OUT A WAY TO TEST THEM UNDER RUN CONDITIONS. THE LOGIC ASSOCIATED WITH THEM HAS BEEN TESTED BY OTHER ROM STATES BUT THESE STATES ARE NOT TESTED.

ROM OUT05
SHR.00, NEG.00, D12.90, D40.30, D12.30

ROM OUT09
FET.06, FET.08, FET.09
THESE NEXT ARE TESTED ON PASSES WITH T-BIT TRAPPING
(PASS 2, 4, ...)
FET.01, FET.02, FET.03

ROM OUT12
EXM.10, EXM.20, DEP.10, DEP.20

ROM OUT13
S45.10

ROM OUT16 FLOATING POINT
FSV.00, FSV.10

9. REVISION HISTORY

REV E0 DOCUMENTATION CHANGES ONLY

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

```
.TITLE CEBBEO 11/70 MEM MGMT
:*COPYRIGHT (C) 1975,1980
:*DIGITAL EQUIPMENT CORP.
:*MAYNARD, MASS. 01754
:*
:*
:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
:*PACKAGE (MAINDEC-11-DZQAC-A3-1).
:*
```

```
.SBTTL OPERATIONAL SWITCH SETTINGS
:*
:*      SWITCH          USE
:-----
:*      15             HALT ON ERROR
:*      14             LOOP ON TEST
:*      13             INHIBIT ERROR TYPEOUTS
:*      12             INHIBIT TRACE TRAP
:*      11             INHIBIT ITERATIONS
:*      10             BELL ON ERROR
:*      9              LOOP ON ERROR
:*      8              LOOP ON TEST IN SWR<6:0>
:*      7              INHIBIT MULTIPLE ERROR TYPEOUTS
```

```
.SBTTL BASIC DEFINITIONS
:*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100          ;;FIRST ADDRESS OF THE STACK
KERSTK= STACK       ;;KERNEL STACK
SUPSTK= STACK-200   ;;SUPERVISOR STACK
USESTK= STACK-300   ;;USER STACK
.EQUIV EMT,ERROR    ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE    ;;BASIC DEFINITION OF SCOPE CALL
PS= 177776          ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
STKLMT= 177774      ;;STACK LIMIT REGISTER
PIRQ= 177772        ;;PROGRAM INTERRUPT REQUEST REGISTER
SWR= 177570         ;;SWITCH REGISTER
DISPLAY=SWR
```

```
.*MISCELLANEOUS DEFINITIONS
HT= 11              ;;CODE FOR HORIZONTAL TAB
LF= 12              ;;CODE LINE FEED
CR= 15              ;;CODE CARRIAGE RETURN
CRLF= 200           ;;CODE FOR CARRIAGE RETURN-LINE FEED
```

```
.*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
.EQUIV R0,R10      ;;GENERAL REGISTER
```

001100
001100
000700
000600

177776

177774
177772
177570
177570

000011
000012
000015
000200

000000
000001
000002
000003
000004
000005
000006
000007

```

57      .EQUIV R1,R11      ;;GENERAL REGISTER
58      .EQUIV R2,R12      ;;GENERAL REGISTER
59      .EQUIV R3,R13      ;;GENERAL REGISTER
60      .EQUIV R4,R14      ;;GENERAL REGISTER
61      .EQUIV R5,R15      ;;GENERAL REGISTER
62      000006      SP=%6
63      .EQUIV SP,KSP      ;;KERNEL STACK POINTER
64      .EQUIV SP,SSP      ;;SUPERVISOR STACK POINTER
65      .EQUIV SP,USP      ;;USER STACK POINTER
66      000007      PC=%7
67
68      ;*PRIORITY LEVEL DEFINITIONS
69      000000      PR0= 0      ;;PRIORITY LEVEL 0
70      000040      PR1= 40      ;;PRIORITY LEVEL 1
71      000100      PR2= 100      ;;PRIORITY LEVEL 2
72      000140      PR3= 140      ;;PRIORITY LEVEL 3
73      000200      PR4= 200      ;;PRIORITY LEVEL 4
74      000240      PR5= 240      ;;PRIORITY LEVEL 5
75      000300      PR6= 300      ;;PRIORITY LEVEL 6
76      000340      PR7= 340      ;;PRIORITY LEVEL 7
77
78      ;*'SWITCH REGISTER' SWITCH DEFINITIONS
79      100000      SW15= 100000
80      040000      SW14= 40000
81      020000      SW13= 20000
82      010000      SW12= 10000
83      004000      SW11= 4000
84      002000      SW10= 2000
85      001000      SW09= 1000
86      000400      SW08= 400
87      000200      SW07= 200
88      000100      SW06= 100
89      000040      SW05= 40
90      000020      SW04= 20
91      000010      SW03= 10
92      000004      SW02= 4
93      000002      SW01= 2
94      000001      SW00= 1
95      .EQUIV SW09,SW9
96      .EQUIV SW08,SW8
97      .EQUIV SW07,SW7
98      .EQUIV SW06,SW6
99      .EQUIV SW05,SW5
100     .EQUIV SW04,SW4
101     .EQUIV SW03,SW3
102     .EQUIV SW02,SW2
103     .EQUIV SW01,SW1
104     .EQUIV SW00,SW0
105
106     ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
107     100000      BIT15= 100000
108     040000      BIT14= 40000
109     020000      BIT13= 20000
110     010000      BIT12= 10000
111     004000      BIT11= 4000
112     002000      BIT10= 2000
    
```

```

113      001000      BIT09= 1000
114      000400      BIT08= 400
115      000200      BIT07= 200
116      000100      BIT06= 100
117      000040      BIT05= 40
118      000020      BIT04= 20
119      000010      BIT03= 10
120      000004      BIT02= 4
121      000002      BIT01= 2
122      000001      BIT00= 1
123      .EQUIV      BIT09,BIT9
124      .EQUIV      BIT08,BIT8
125      .EQUIV      BIT07,BIT7
126      .EQUIV      BIT06,BIT6
127      .EQUIV      BIT05,BIT5
128      .EQUIV      BIT04,BIT4
129      .EQUIV      BIT03,BIT3
130      .EQUIV      BIT02,BIT2
131      .EQUIV      BIT01,BIT1
132      .EQUIV      BIT00,BIT0
    
```

```

134      ;*BASIC 'CPU' TRAP VECTOR ADDRESSES
135      000004      ERRVEC= 4          ;;TIME OUT AND OTHER ERRORS
136      000010      RESVEC= 10         ;;RESERVED AND ILLEGAL INSTRUCTIONS
137      000014      TBITVEC=14         ;;'T' BIT
138      000014      TRTVEC= 14         ;;TRACE TRAP
139      000014      BPTVEC= 14         ;;BREAKPOINT TRAP (BPT)
140      000020      IOTVEC= 20         ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
141      000024      PWRVEC= 24         ;;POWER FAIL
142      000030      EMTVEC= 30         ;;EMULATOR TRAP (EMT) **ERROR**
143      000034      TRAPVEC=34         ;;'TRAP' TRAP
144      000060      TKVEC= 60          ;;TTY KEYBOARD VECTOR
145      000064      TPVEC= 64          ;;TTY PRINTER VECTOR
146      000114      CACHVEC=114        ;;CACHE ERROR INTERRUPT VECTOR
147      000240      PIRQVEC=240        ;;PROGRAM INTERRUPT REQUEST VECTOR
148      000250      MMVEC= 250         ;;MEMORY MANAGEMENT VECTOR
    
```

149
150 .SBTTL CACHE REGISTER DEFINITIONS

```

151
152
153      177740      LOADRS = 177740     ;;LOWER 16 BITS OF ADDRESS THAT CAUSED ERROR
154      177742      HIADRS = 177742    ;;UPPER SIX BITS OF ADDRESS THAT CAUSED ERROR
155      177744      MEMERR = 177744     ;;CACHE ERROR REGISTER
156      177746      CONTRL = 177746    ;;MEMORY CONTROL REGISTER
157      177750      MAINT = 177750     ;;MEMORY MAINTENANCE REGISTER
158      177752      HITMIS = 177752    ;;HIT MISS REGISTER '1' IMPLIES HIT IN CACHE
    
```

159
160
161 .SBTTL CPU REGISTER DEFINITIONS

```

162
163
164      177760      SIZELO = 177760     ;;MEMORY SIZE REGISTER NUMBER TO PUT INTO A PAR
165      177762      SIZEHI = 177762    ;;TO GET TO THE LAST 32 WORDS OF MEMORY
166      177764      SYSTID = 177764    ;;HIGH SIZE REGISTER, RESERVED FOR FUTURE USE
167      177764      SYSTID = 177764    ;;CURRENTLY ALL ZERO
168      177764      SYSTID = 177764    ;;SYSTEM ID REGISTER
    
```

```
169          177766          CPUERR = 177766          ::CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
170                                     ::THE TRAP TO ERRVEC (000004)
171
172
173
174
175          .SBTTL  MEMORY MANAGEMENT DEFINITIONS
176
177
178          ;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES
179
180          177572          MMR0= 177572
181          177574          MMR1= 177574
182          177576          MMR2= 177576
183          172516          MMR3= 172516
184          .EQUIV  MMR0,SRO
185          .EQUIV  MMR1,SR1
186          .EQUIV  MMR2,SR2
187          .EQUIV  MMR3,SR3
188
189          ;*USER 'I' PAGE DESCRIPTOR REGISTERS
190
191          177600          UIPDR0= 177600
192          177602          UIPDR1= 177602
193          177604          UIPDR2= 177604
194          177606          UIPDR3= 177606
195          177610          UIPDR4= 177610
196          177612          UIPDR5= 177612
197          177614          UIPDR6= 177614
198          177616          UIPDR7= 177616
199
200          ;*USER 'D' PAGE DESCRIPTOR REGISTORS
201
202          177620          UDPDR0= 177620
203          177622          UDPDR1= 177622
204          177624          UDPDR2= 177624
205          177626          UDPDR3= 177626
206          177630          UDPDR4= 177630
207          177632          UDPDR5= 177632
208          177634          UDPDR6= 177634
209          177636          UDPDR7= 177636
210
211          ;*USER 'I' PAGE ADDRESS REGISTERS
212
213          177640          UIPAR0= 177640
214          177642          UIPAR1= 177642
215          177644          UIPAR2= 177644
216          177646          UIPAR3= 177646
217          177650          UIPAR4= 177650
218          177652          UIPAR5= 177652
219          177654          UIPAR6= 177654
220          177656          UIPAR7= 177656
221
222          ;*USER 'D' PAGE ADDRESS REGISTERS
223
224          177660          UDPAR0= 177660
```


225	177662	UDPAR1= 177662
226	177664	UDPAR2= 177664
227	177666	UDPAR3= 177666
228	177670	UDPAR4= 177670
229	177672	UDPAR5= 177672
230	177674	UDPAR6= 177674
231	177676	UDPAR7= 177676
232		
233		:*SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS
234		
235	172200	SIPDR0= 172200
236	172202	SIPDR1= 172202
237	172204	SIPDR2= 172204
238	172206	SIPDR3= 172206
239	172210	SIPDR4= 172210
240	172212	SIPDR5= 172212
241	172214	SIPDR6= 172214
242	172216	SIPDR7= 172216
243		
244		:*SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS
245		
246	172220	SDPDR0= 172220
247	172222	SDPDR1= 172222
248	172224	SDPDR2= 172224
249	172226	SDPDR3= 172226
250	172230	SDPDR4= 172230
251	172232	SDPDR5= 172232
252	172234	SDPDR6= 172234
253	172236	SDPDR7= 172236
254		
255		:*SUPERVISOR 'I' PAGE ADDRESS REGISTERS
256		
257	172240	SIPAR0= 172240
258	172242	SIPAR1= 172242
259	172244	SIPAR2= 172244
260	172246	SIPAR3= 172246
261	172250	SIPAR4= 172250
262	172252	SIPAR5= 172252
263	172254	SIPAR6= 172254
264	172256	SIPAR7= 172256
265		
266		:*SUPERVISOR 'D' PAGE ADDRESS REGISTERS
267		
268	172260	SDPAR0= 172260
269	172262	SDPAR1= 172262
270	172264	SDPAR2= 172264
271	172266	SDPAR3= 172266
272	172270	SDPAR4= 172270
273	172272	SDPAR5= 172272
274	172274	SDPAR6= 172274
275	172276	SDPAR7= 172276
276		
277		:*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
278		
279	172300	KIPDR0= 172300
280	172302	KIPDR1= 172302

```
281      172304      KIPDR2= 172304
282      172306      KIPDR3= 172306
283      172310      KIPDR4= 172310
284      172312      KIPDR5= 172312
285      172314      KIPDR6= 172314
286      172316      KIPDR7= 172316
287
288      ;*KERNEL 'D' PAGE DESCRIPTOR REGISTERS
289
290      172320      KDPDR0= 172320
291      172322      KDPDR1= 172322
292      172324      KDPDR2= 172324
293      172326      KDPDR3= 172326
294      172330      KDPDR4= 172330
295      172332      KDPDR5= 172332
296      172334      KDPDR6= 172334
297      172336      KDPDR7= 172336
298
299      ;*KERNEL 'I' PAGE ADDRESS REGISTERS
300
301      172340      KIPAR0= 172340
302      172342      KIPAR1= 172342
303      172344      KIPAR2= 172344
304      172346      KIPAR3= 172346
305      172350      KIPAR4= 172350
306      172352      KIPAR5= 172352
307      172354      KIPAR6= 172354
308      172356      KIPAR7= 172356
309
310      ;*KERNEL 'D' PAGE ADDRESS REGISTERS
311
312      172360      KDPAR0= 172360
313      172362      KDPAR1= 172362
314      172364      KDPAR2= 172364
315      172366      KDPAR3= 172366
316      172370      KDPAR4= 172370
317      172372      KDPAR5= 172372
318      172374      KDPAR6= 172374
319      172376      KDPAR7= 172376
320
321
322
323
324      .SBTTL UNIBUS MAP REGISTER DEFINITIONS
325
326      ;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
327      ;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'
328
329
330
331      170200      MAPL00 = 170200
332      170202      MAPH00 = 170202
333      170204      MAPL01 = 170204
334      170206      MAPH01 = 170206
335      170210      MAPL02 = 170210
336      170212      MAPH02 = 170212
```

337	170214	MAPL03 = 170214
338	170216	MAPH03 = 170216
339	170220	MAPL04 = 170220
340	170222	MAPH04 = 170222
341	170224	MAPL05 = 170224
342	170226	MAPH05 = 170226
343	170230	MAPL06 = 170230
344	170232	MAPH06 = 170232
345	170234	MAPL07 = 170234
346	170236	MAPH07 = 170236
347	170240	MAPL10 = 170240
348	170242	MAPH10 = 170242
349	170244	MAPL11 = 170244
350	170246	MAPH11 = 170246
351	170250	MAPL12 = 170250
352	170252	MAPH12 = 170252
353	170254	MAPL13 = 170254
354	170256	MAPH13 = 170256
355	170260	MAPL14 = 170260
356	170262	MAPH14 = 170262
357	170264	MAPL15 = 170264
358	170266	MAPH15 = 170266
359	170270	MAPL16 = 170270
360	170272	MAPH16 = 170272
361	170274	MAPL17 = 170274
362	170276	MAPH17 = 170276
363	170300	MAPL20 = 170300
364	170302	MAPH20 = 170302
365	170304	MAPL21 = 170304
366	170306	MAPH21 = 170306
367	170310	MAPL22 = 170310
368	170312	MAPH22 = 170312
369	170314	MAPL23 = 170314
370	170316	MAPH23 = 170316
371	170320	MAPL24 = 170320
372	170320	MAPH24 = 170320
373	170324	MAPL25 = 170324
374	170326	MAPH25 = 170326
375	170330	MAPL26 = 170330
376	170332	MAPH26 = 170332
377	170334	MAPL27 = 170334
378	170336	MAPH27 = 170336
379	170340	MAPL30 = 170340
380	170342	MAPH30 = 170342
381	170344	MAPL31 = 170344
382	170346	MAPH31 = 170346
383	170350	MAPL32 = 170350
384	170352	MAPH32 = 170352
385	170354	MAPL33 = 170354
386	170356	MAPH33 = 170356
387	170360	MAPL34 = 170360
388	170362	MAPH34 = 170362
389	170364	MAPL35 = 170364
390	170366	MAPH35 = 170366
391	170370	MAPL36 = 170370
392	170372	MAPH36 = 170372

393	170374	MAPL37 = 170374
394	170376	MAPH37 = 170376
395		.EQUIV MAPL00,MAPL0
396		.EQUIV MAPH00,MAPH0
397		.EQUIV MAPL01,MAPL1
398		.EQUIV MAPH01,MAPH1
399		.EQUIV MAPL02,MAPL2
400		.EQUIV MAPH02,MAPH2
401		.EQUIV MAPL03,MAPL3
402		.EQUIV MAPH03,MAPH3
403		.EQUIV MAPL04,MAPL4
404		.EQUIV MAPH04,MAPH4
405		.EQUIV MAPL05,MAPL5
406		.EQUIV MAPH05,MAPH5
407		.EQUIV MAPL06,MAPL6
408		.EQUIV MAPH06,MAPH6
409		.EQUIV MAPL07,MAPL7
410		.EQUIV MAPH07,MAPH7

;DEFINITIONS

417	100000	VSPE=BIT15
418	040000	IVSS=BIT14
419	020000	VSIU=BIT13
420	010000	VCIP=BIT12
421	004000	DMMA=BIT11
422	002000	FVPE=BIT10
423	001000	UCB=BIT9
424	000400	FCAC=BIT8
425	000040	S1=BIT5
426	000020	S0=BIT4
427	000010	M1=BIT3
428	000004	M0=BIT2
429	000002	DUT=BIT1
430	000001	DT=BIT0

432	100000	BYP=BIT15
-----	--------	-----------

434	000054	S1MOM1=BIT5+BIT3+BIT2
435	000034	S0MOM1=BIT4+BIT3+BIT2
436	000014	MOM1=BIT3+BIT2

438	177746	CONTRL=177746
439	177752	HITMIS=177752
440	177744	MSER=177744

::*****

.SBTTL TRAP CATCHER

446	000000	. = 0
-----	--------	-------

;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS

448

```
449 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
450
451 .SBTTL STARTING ADDRESS(ES)
452      .=200
453
454 000200 000137 040000      JMP      @#STRT1      ;; JUMP TO STARTING ADDRESS OF PROGRAM
455                                ;; RUN ENTIRE PROGRAM (ALL TESTS)
456 000204 000137 040010      JMP      STRT2        ;; STARTING AT ENTRY POINT 2
457                                ;; MEMORY MANAGEMENT STATUS REGISTERS
458 000210 000137 040020      JMP      STRT3        ;; STARTING AT ENTRY POINT 3
459                                ;; PAGE ADDRESS AND DESCRIPTOR REGISTERS
460 000214 000137 040030      JMP      STRT4        ;; STARTING AT ENTRY POINT 4
461                                ;; RELOCATION AND ADDER TESTS
462 000220 000137 040040      JMP      STRT5        ;; STARTING AT ENTRY POINT 5
463                                ;; MEMORY MANAGEMENT TRAP AND ABORT LOGIC
464 000224 000137 040050      JMP      STRT6        ;; STARTING AT ENTRY POINT 6
465                                ;; D-SPACE ENABLING
466 000230 000137 040060      JMP      STRT7        ;; STARTING AT ENTRY POINT 7
467                                ;; A & W BIT LOGIC & DUAL MAPPING
468 000234 000137 040070      JMP      STRT8        ;; STARTING AT ENTRY POINT 8
469                                ;; MFP & MTP LOGIC TESTS
470 ;:*****
471
472 .SBTTL      ACT11 HOOKS
473
474 ;*THE FOLLOWING LOCATIONS ARE SETUP TO BE USED WITH ACT11
475 ;*
476 ;*LOCATION 46 WILL CONTAIN THE ADDRESS OF THE LOGICAL
477 ;*END OF THE PROGRAM.
478 ;*LOCATION 52 IS USED TO SPECIFY PROGRAM OPERATING REQUIREMENTS
479 ;*AND/OR RESTRICTIONS. THIS IS ACCOMPLISHED BY SETTING VARIOUS BITS
480 ;*TO A ONE OR A ZERO. THE BITS USED AND THERE MEANING ARE:
481 ;*
482 ;*      BIT 15=1 PROGRAM SHOULD BE POWER FAILED WHILE RUNNING
483 ;*          =0 NO POWER FAIL DESIRED
484 ;*
485 ;*      BIT 14=1 PROGRAM RUN TIME IS MEMORY SIZE DEPENDENT
486 ;*          =0 RUN TIME IS NOT MEMORY SIZE DEPENDENT
487 ;*
488 ;*      BITS 13-0 MUST BE ZERO'S
489 ;*
490      000240      $SVPC=.      ;; SAVE LOCATION COUNTER
491      000046      .=46        ;; SET LOCATION COUNTER
492 000046 025412      .WORD  SENDAD      ;; SET LOC.46 TO ADDRESS SENDAD
493      000052      .=52        ;; SET LOCATION COUNTER
494 000052 000000      .WORD  0        ;; SET LOC.52 TO ZERO
495      000240      .=$SVPC      ;; RESTORE LOCATION COUNTER
```

```

496 ;:*****
497
498 .SBTTL COMMON TAGS
499
500 ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
501 ;*USED IN THE PROGRAM.
502
503         001100             .=1100
504
505 001100 SCMTAG:           ;; START OF COMMON TAGS
506 001100 000000 $PASS: .WORD 0           ;; CONTAINS PASS COUNT
507 001102 000 $STNM: .BYTE 0           ;; CONTAINS THE TEST NUMBER
508 001103 000 $ERFLG: .BYTE 0          ;; CONTAINS ERROR FLAG
509 001104 000000 $STSTNM: .WORD 0      ;; TEST NUMBER STORAGE
510 001106 000000 $ICNT: .WORD 0        ;; CONTAINS SUBTEST ITERATION COUNT
511 001110 000000 $LPADR: .WORD 0       ;; CONTAINS SCOPE LOOP
512 001112 000000 $LPERR: .WORD 0      ;; CONTAINS SCOPE RETURN FOR ERRORS
513 001114 000000 $L_TTL: .WORD 0      ;; CONTAINS TOTAL ERRORS DETECTED
514 001116 000 $ITEMB: .BYTE 0         ;; CONTAINS ITEM CONTROL BYTE
515 001117 001 $ERMAX: .BYTE 1         ;; CONTAINS MAX. ERRORS PER TEST
516 001120 000000 $ERRPC: .WORD 0      ;; CONTAINS PC OF LAST ERROR INSTRUCTION
517 001122 000000 $GDADR: .WORD 0      ;; CONTAINS OF 'GOOD' DATA
518 001124 000000 $BDADR: .WORD 0      ;; CONTAINS OF 'BAD' DATA
519 001126 000000 $GDDAT: .WORD 0      ;; CONTAINS 'GOOD' DATA
520 001130 000000 $BDDAT: .WORD 0      ;; CONTAINS 'BAD' DATA
521 001132 000000 000000 000000 $RESV: .WORD 0,0,0 ;; RESERVED--NOT TO BE USED
522 001140 177560 $TKS: 177560         ;; TTY KBD STATUS
523 001142 177562 $TKB: 177562         ;; TTY KBD BUFFER
524 001144 177564 $TPS: 177564         ;; TTY PRINTER STATUS REG.
525 001146 177566 $TPB: 177566         ;; TTY PRINTER BUFFER REG.
526 001150 000 $NULL: .BYTE 0          ;; CONTAINS NULL CHARACTER FOR FILLS
527 001151 002 $FILLS: .BYTE 2         ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
528 001152 012 $FILLC: .BYTE 12       ;; INSERT FILL CHARS. AFTER A 'LINE FEED'
529 001153 000 $TPFLG: .BYTE 0        ;; 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
530 001154 000000 $REGAD: .WORD 0     ;; CONTAINS THE FROM
531 ;; WHICH ($REGO) WAS OBTAINED
532 001156 000000 $REG0: .WORD 0       ;; CONTAINS ((SREGAD)+0)
533 001160 000000 $REG1: .WORD 0       ;; CONTAINS ((SREGAD)+2)
534 001162 000000 $REG2: .WORD 0       ;; CONTAINS ((SREGAD)+4)
535 001164 000000 $REG3: .WORD 0       ;; CONTAINS ((SREGAD)+6)
536 001166 000000 $REG4: .WORD 0       ;; CONTAINS ((SREGAD)+10)
537 001170 000000 $REG5: .WORD 0       ;; CONTAINS ((SREGAD)+12)
538 001172 000000 $TMP0: .WORD 0       ;; USER DEFINED
539 001174 000000 $TMP1: .WORD 0       ;; USER DEFINED
540 001176 000000 $TMP2: .WORD 0       ;; USER DEFINED
541 001200 000000 $TMP3: .WORD 0       ;; USER DEFINED
542 001202 000000 $TMP4: .WORD 0       ;; USER DEFINED
543 001204 000000 $TMP5: .WORD 0       ;; USER DEFINED
544 001206 000000 $TIMES: 0           ;; MAX. NUMBER OF ITERATIONS
545 001210 000000 $ESCAPE: 0          ;; ESCAPE ON ERROR
546 001212 177607 000377 $BELL: .ASCIZ <207><377><377> ;; CODE FOR BELL
547 001216 077 $QUES: .ASCII /?/        ;; QUESTION MARK
548 001217 015 $CRLF: .ASCII <15>     ;; CARRIAGE RETURN
549 001220 000012 $LF: .ASCIZ <12>         ;; LINE FEED
550 001222 000000 $FSTTST: .WORD 0    ;; HOLDS THE INDEX TO
551 ;; THE STARTING ADDRESS TABLE
    
```

552	001224	000000	CPUEXP: .WORD	0	:HOLDS EXPECTED CPU ERROR CONDITION
553	001226	000000	MPEXP: .WORD		:HOLDS EXPECTED MEMORY MANAGEMENT ABORT CONDITION
554	001230	000000	PADRSL: .WORD	0	:HOLDS THE LOWER 16 BITS OF A 22-BIT
555					:ADDRESS GENERATED FOR TYPE OUT
556	001232	000000	PADRSH: .WORD	0	:HOLDS THE UPPER 6 BITS OF A 22-BIT
557					:ADDRESS GENERATED FOR TYPE OUT
558	001234	000000	PLOADR: .WORD	0	:HOLDS CACHE LO ADDR REG
559	001236	000000	PHIADR: .WORD	0	:HOLDS CACHE HI ADDR REG
560	001240	000000	PPARER: .WORD	0	:HOLDS PARITY ERROR REG
561	001242	000000	PCONTR: .WORD	0	:HOLDS CACHE CONTROL REG
562	001244	000000	PMAINT: .WORD	0	:HOLDS CACHE MAINTENANCE REG
563	001246	000000	PHITMI: .WORD	0	:HOLDS CACHE HIT MISS REG
564	001250	000000	PMR0: .WORD	0	:HOLDS MEMORY MANAGEMENT REGISTER 0
565	001252	000000	PMR1: .WORD	0	:HOLDS MEMORY MANAGEMENT REGISTER 1
566	001254	000000	PMR2: .WORD	0	:HOLDS MEMORY MANAGEMENT REGISTER 2
567	001256	000000	PMR3: .WORD	0	:HOLDS MEMORY MANAGEMENT REGISTER 3
568	001260	000000	PCPUER: .WORD	0	:HOLDS CPU ERROR REGISTER.
569	001262	000000	BADPC: .WORD	0	:HOLDS PC AT ABORT OR TRAP TIME.
570	001264	000000	TESTNO: .WORD	0	:HOLDS TEST NUMBER OF LAST ERROR.
571	001266	000000	DATAOR: .WORD	0	:HOLDS LOGICAL OR OF BAD DATA
572	001270	000000	DATAND: .WORD	0	:HOLDS LOGICAL AND OF BAD DATA
573	001272	000000	PATTOR: .WORD	0	:HOLDS LOGICAL OR OF PATTERN LOADED
574	001274	000000	PATAND: .WORD	0	:HOLDS LOGICAL AND OF PATTERN LOADED
575	001276	000000	ADDROR: .WORD	0	:HOLDS LOGICAL OR OF ADDRESS
576	001300	000000	ADRAND: .WORD	0	:HOLDS LOGICAL AND OF ADDRESS
577	001302	000000	ERRCNT: .WORD	0	:HOLDS NUMBER OF ERRORS ON TEST
578	001304	000000	HOLFLG: .WORD	0	:HOLDS NUMBER OF CONSECUTIVE TIME OUTS
579					:OCCURRING IN A HOLE IN MEMORY
580	001306	000000	OLDPC: .WORD	0	:HOLDS THE RETURN ADDRESS
581					:IN CASE OF A LOOP ON ERROR
582	001310	000000	OLDPS: .WORD	0	:HOLDS THE OLD PROCESSOR STATUS
583					:IN CASE OF A LOOP ON ERROR
584	001312	000340	OLDPSW: .WORD	000340	:HOLDS THE OLD PSW SO THAT THE T-BIT
585					:CAN BE RESTORED PROPERLY
586	001314	000000	RETRY: .WORD	0	:FLAG TO DECIDE IF ONE PARITY HAS
587					:HAS BEEN ATTEMPTED
588	001316	000000	NXTTST: .WORD	0	:HOLDS ADDRESS OF THE NEXT TEST
589					
590			:: THIS AREA STARTS THE DATA TABLE WHERE ANY TABLE REFERENCE WILL		
591			:: REFER TO. ALL DATA WORDS WILL BE LOADED HERE IN ONE SPOT SO THAT		
592			:: IT WILL BE EASIER FOR YOU TO FIND OUT WHAT THAT WORD IS.		
593					
594	001320	020000	READON: .WORD	20000	:READ ONLY BIT IN MMR0
595					
596					
597	001322		PARTAB:		:THIS IS THE TABLE OF THE FIRST
598					:PAR OR PDR OF EACH GROUP. THEY
599					:WILL BE USED FOR A DUAL ADDRESSING
600					:TEST BETWEEN GROUPS.
601	001322	172200	.WORD	172200	:SIPDR0
602	001324	172240	.WORD	172240	:SIPAR0
603	001326	172300	.WORD	172300	:KIPDR0
604	001330	172340	.WORD	172340	:KIPAR0
605	001332	177600	.WORD	177600	:UIPDR0
606	001334	177640	.WORD	177640	:UIPAR0
607	001336	040644	STRTAB: .WORD	TST1	:STARTING ADDRESS OF TEST ONE

608	001340	041612	.WORD	ENTPT2	:ADDRESS OF ENTRY POINT 2
609	001342	042770	.WORD	ENTPT3	:ADDRESS OF ENTRY POINT 3
610	001344	047500	.WORD	ENTPT4	:ADDRESS OF ENTRY POINT 4
611	001346	052626	.WORD	ENTPT5	:ADDRESS OF ENTRY POINT 5
612	001350	060000	.WORD	ENTPT6	:ADDRESS OF ENTRY POINT 6
613	001352	062470	.WORD	ENTPT7	:ADDRESS OF ENTRY POINT 7
614	001354	070170	.WORD	ENTPT8	:ADDRESS OF ENTRY POINT 8
615	001356	001000	ENMMTR: .WORD	BIT9	:ENABLE MEMORY MANAGEMENT TRAPS BIT
616	001360	000	KB11E: .BYTE	0	:KB-11E WITHOUT MP CACHE FLAG
617	001361	000	KB11EM: .BYTE	0	:KB-11EM WITH MP CACHE FLAG
618	001362	000	KB11CM: .BYTE	0	:MODIFIED PROCESSOR FLAG (11/70 WITH MP MODS)
619	001363	000	CISP: .BYTE	0	:CISP OPTION FLAG (NOT PRESENTLY NEEDED)
620					
621					
622		000007			:OPCODE FOR MFPT INSTRUCTION (AVAILABLE ON KB11-E AND KB11-EM ONLY)
					MFPT=7


```

623      ;:*****
624
625      .SBTTL  ERROR POINTER TABLE
626
627      ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
628      ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
629      ;*LOCATION SITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
630      ;*NOTE1:      IF SITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
631      ;*NOTE2:      EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
632
633      ;*      EM      ;;POINTS TO THE ERROR MESSAGE
634      ;*      DH      ;;POINTS TO THE DATA HEADER
635      ;*      DT      ;;POINTS TO THE DATA
636      ;*      DF      ;;POINTS TO THE DATA FORMAT
637
638
639      001364      $ERRTB:
640
641
642      ;ITEM 1
643      001364      003447      EM1      ;NOT THE CORRECT CPU ABORT CONDITION THRU 'ERRVEC' (004)
644      001366      017003      DH1      ;EXPECTD RECEIVD TESTNO PC AT ABORT
645      001370      023532      DT1      ;CPUEXP,PCPUER,TESTNO,BADPC,0
646      001372      024650      DF1      ;0,0,0,0
647
648      ;ITEM 2
649      001374      003537      EM2      ;UNEXPECTED CPU TRAP OR ABORT THRU 'ERRVEC' (004)
650      001376      017013      DH2      ;RECEIVD TESTNO PC AT ABORT
651      001400      023534      DT2      ;PCPUER,TESTNO,BADPC,0
652      001402      024651      DF2      ;0,0,0
653
654      ;ITEM 3
655      001404      003620      EM3      ;UNEXPECTED CACHE PARITY ERROR THRU 'CACHVEC' (114)
656      001406      017047      DH3      ;PARITY ADDRESS CONTROL MAINTEN
657      ;RECEIVD REFERENC'D REGISTR REGISTR TESTNO PC AT ABORT
658      001410      023544      DT3      ;PPARER,PLOADR,PCONTR,PMAINT,TESTNO,BADPC,0
659      001412      024654      DF3      ;0,2,0,0,0,0
660
661      ;ITEM 4
662      001414      003736      EM4      ;UNEXPECTED MAIN MEMORY PARITY ERROR THRU 'CACHVEC' (114)
663      001416      017047      DH3      ;PARITY ADDRESS CONTROL MAINTEN
664      ;RECEIVD REFERENC'D REGISTR REGISTR TESTNO PC AT ABORT
665      001420      023544      DT3      ;PPARER,PLOADR,PCONTR,PMAINT,TESTNO,BADPC,0
666      001422      024654      DF3      ;0,2,0,0,0,0
667
668      ;ITEM 5
669      001424      004061      EM5      ;UNEXPECTED MEMORY MANAGEMENT TRAP OR ABORT
670      001426      017177      DH5      ;ERROR AUTOI/D VIRTUAL
671      ;REGISTR REGISTR ADDRESS TESTNO PC AT ABORT
672      001430      023562      DT5      ;PMR0,PMR1,PMR2,TESTNO,BADPC,0
673      001432      024662      DF5      ;0,0,0,0,0
674
675      ;ITEM 6
676      001434      004134      EM6      ;MEMORY MANAGEMENT TRAP OR ABORT HAD INCORRECT CONDITION
677      001436      017303      DH6      ;EXPECTD ERROR AUTOI/D VIRTUAL
678      ;CONDITN REGISTR REGISTR ADDRESS TESTNO PC AT ABORT
    
```

679	001440	023576	DT6	:MMEXP,MMR0,MMR1,MMR2,TESTNO,BADPC,0
680	001442	024667	DF6	:0,0,0,0,0,0
681				
682			:ITEM 7	
683	001444	004224	EM7	:MEMORY MANAGEMENT REGISTER 0 WILL NOT CLEAR
684	001446	017427	DH7	:(MMR0) TESTNO ERRORPC
685	001450	023614	DT7	:\$REG1,TESTNO,\$ERRPC,0
686	001452	024675	DF7	:0,0,0
687				
688			:ITEM 10	
689	001454	004300	EM10	:CAN'T SET 171000 INTO MMR0
690	001456	017427	DH7	:(MMR0) TESTNO ERRORPC
691	001460	023614	DT7	:\$REG1,TESTNO,\$ERRPC,0
692	001462	024675	DF7	:0,0,0
693				
694			:ITEM 11	
695	001464	004331	EM11	:GOT THE WRONG DATA BACK FROM MMR0
696	001466	017457	DH11	:LOADED RECEIVD TESTNO ERRORPC
697	001470	023624	DT11	:\$REG2,\$REG1,TESTNO,\$ERRPC,0
698	001472	024700	DF11	:0,0,0,0
699				
700			:ITEM 12	
701	001474	004373	EM12	:MEMORY MANAGEMENT REGISTER 1 WILL NOT CLEAR
702	001476	017527	DH12	:(MMR1) TESTNO ERRORPC
703	001500	023636	DT12	:\$REG2,TESTNO,\$ERRPC,0
704	001502	024704	DF12	:0,0,0
705				
706			:ITEM 13	
707	001504	004447	EM13	:MMR1 DID NOT TRACK PROPERLY
708	001506	017517	DH13	:EXPECTD (MMR1) TESTNO ERRORPC
709	001510	023646	DT13	:\$REG3,\$REG2,TESTNO,\$ERRPC,0
710	001512	024707	DF13	:0,0,0,0
711				
712			:ITEM 14	
713	001514	004503	EM14	:MMR2 DID NOT TRACK PROPERLY
714	001516	017557	DH14	:EXPECTD (MMR2) TESTNO ERRORPC
715	001520	023646	DT13	:\$REG3,\$REG2,TESTNO,\$ERRPC
716	001522	024707	DF13	:0,0,0,0
717				
718			:ITEM 15	
719	001524	004537	EM15	:MEMORY MANAGEMENT REGISTER 3 WILL NOT CLEAR
720	001526	017627	DH15	:(MMR3) TESTNO ERRORPC
721	001530	023636	DT12	:\$REG,TESTNO,\$ERRPC,0
722	001532	024704	DF12	:0,0,0
723				
724			:ITEM 16	
725	001534	004613	EM16	:MMR3 IS HOLDING THE WRONG DATA
726	001536	017617	DH16	:LOADED (MMR3) TESTNO ERRORPC
727	001540	023660	DT16	:\$REG1,\$REG2,TESTNO,\$ERRPC,0
728	001542	024713	DF16	:0,0,0,0
729				
730			:ITEM 17	
731	001544	004646	EM17	:SUMMARY OF PAR/PDR REFERENCE TIMEOUTS
732	001546	017657	DH17	:ADDROR ADDRAND TESTNO #ERRORS
733	001550	023672	DT17	:ADDROR,ADDRAND,TESTNO,ERRCNT,0
734	001552	024722	DF17	:0,0,0,1

Line No.	Code	Address	Register	Description
735				
736			:ITEM 20	
737	001554	004714	EM20	: FOLLOWING PAR/PDR WILL NOT ZERO
738	001556	017717	DH20	: ADDRESS DATA TESTNO ERRORPC
739	001560	023704	DT20	: \$REG0,\$REG2,TESTNO,\$ERRPC,0
740	001562	024726	DF20	: 0,0,0,0
741				
742			:ITEM 21	
743	001564	004755	EM21	: SUMMARY OF DUAL ADDRESSING ERRORS
744	001566	017757	DH21	: ADDROR ADDRAND ADDROR ADDRAND
745				: LOADED LOADED ENABLED ENABLED TESTNO #ERRORS
746	001570	023716	DT21	: ADDROR,ADRAND,DATAOR,DATAAND,TESTNO,ERRCNT,0
747	001572	024732	DF21	: 0,0,0,0,0,1
748				
749			:ITEM 22	
750	001574	005017	EM22	: SUMMARY OF COUNT PATTERN FAILURES
751	001576	020077	DH22	: ADDROR ADDRAND PATROR PATRAND DATAOR DATAAND TESTNO #ERRORS
752	001600	023734	DT22	: ADDROR,ADRAND,PATOR,PATAND,DATAOR,DATAAND,TESTNO,ERRCNT,0
753	001602	024740	DF22	: 0,0,0,0,0,0,1
754				
755			:ITEM 23	
756	001604	005061	EM23	: ERROR IN BYTE ADDRESSING OF PAR/PDR
757	001606	020176	DH23	: ADDRESS EXPECTD RECEIVD TESTNO ERRORPC
758	001610	023756	DT23	: \$REG0,\$REG2,\$REG1,TESTNO,\$ERRPC,0
759	001612	024750	DF23	: 0,0,0,0,0
760				
761			:ITEM 24	
762	001614	005125	EM24	: ONE OF THE REGISTERS TIMED OUT
763	001616	020246	DH24	: REGADDR TESTNO ERRORPC
764	001620	023772	DT24	: \$REG0,TESTNO,\$ERRPC,0
765	001622	024755	DF24	: 0,0,0
766				
767			:ITEM 25	
768	001624	005164	EM25	: LOW BYTE OF LOW SIZE REGISTER IS NOT ALL ONES
769	001626	020276	DH25	: REGADDR DATA TESTNO ERRORPC
770	001630	024002	DT25	: \$REG0,\$REG1,TESTNO,\$ERRPC,0
771	001632	024760	DF25	: 0,0,0,0
772				
773			:ITEM 26	
774	001634	005242	EM26	: COULD WRITE ONE OF THE SIZE REGISTERS
775	001636	020276	DH25	: REGADDR DATA TESTNO ERRORPC
776	001640	024002	DT25	: \$REG0,\$REG2,TESTNO,\$ERRPC,0
777	001642	024760	DF25	: 0,0,0,0
778				
779			:ITEM 27	
780	001644	005310	EM27	: HIGH SIZE REGISTER IS NOT ZERO
781	001646	020276	DH25	: REGADDR DATA TESTNO ERRORPC
782	001650	024002	DT25	: \$REG0,\$REG1,TESTNO,\$ERRPC,0
783	001652	024760	DF25	: 0,0,0,0
784				
785			:ITEM 30	
786	001654	005347	EM30	: CPU ERROR REGISTER NOT ZERO AFTER LOADING NEGATIVE ONE.
787	001656	020276	DH25	: REGADDR DATA TESTNO ERRORPC
788	001660	024002	DT25	: \$REG0,\$REG1,TESTNO,\$ERRPC
789	001662	024760	DF25	: 0,0,0,0
790				

791			: ITEM 31	
792	001664	005436	EM31	: LOWER BYTE OF P.S.W. NOT CORRECT
793	001666	020336	DH31	: REGADDR DATAREC DATAEXP TESTNO ERRORPC
794	001670	024014	DT31	: \$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0
795	001672	024764	DF31	: 0,0,0,0,0
796				
797			: ITEM 32	
798	001674	005477	EM32	: MICRO BREAK REG LOADED INCORECTLY
799	001676	020336	DH31	: REGADDR DATAREC DATAEXP TESTNO ERRORPC
800	001700	024014	DT31	: \$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0
801	001702	024764	DF31	: 0,0,0,0,0
802				
803			: ITEM 33	
804	001704	005711	EM33	: DIDN'T LOAD PROGRAM INTERRUPT REQUEST REGISTER
805	001706	020336	DH31	: REGADDR DATAREC DATAEXP TESTNO ERRORPC
806	001710	024014	DT31	: \$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0
807	001712	024764	DF31	: 0,0,0,0,0
808				
809			: IJWM 34	
810	001714	005770	EM34	: LOADED MORE THAN UPPER BYTE OF STACK LIMIT REGISTER
811	001716	020336	DH31	: REGADDR DATAREC DATAEXP TESTNO ERRORPC
812	001720	024014	DT31	: \$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0
813	001722	024764	DF31	: 0,0,0,0,0
814				
815			: ITEM 35	
816	001724	006054	EM35	: KERNEL STACK POINTER NOT 1100 AFTER LOADING SSP AND USP
817	001726	020406	DH35	: KSP TESTNO ERRORPC
818	001730	024030	DT35	: \$REG0,TESTNO,\$ERRPC,0
819	001732	024771	DF35	: 0,0,0
820				
821			: ITEM 36	
822	001734	006142	EM36	: DUAL ADDRESSING BETWEEN PAR/PDR GROUPS
823	001736	020436	DH36	: INDEX INDEX PAR/PDR
824				: EXPECTD RECEIVD ADDRREAD TESTNO ERRORPC
825	001740	024040	DT36	: \$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0
826	001742	024774	DF36	: 0,0,0,0,0
827				
828			: ITEM 37	
829	001744	006211	EM37	: BAD RELOCATION, ON STORING DATA 18-BIT MAPPING
830	001746	020536	DH37	: ADDRESS GDDATA BADDATA TESTNO ERRORPC
831	001750	024054	DT37	: \$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0
832	001752	025001	DF37	: 3,0,0,0,0
833				
834			: ITEM 40	
835	001754	006270	EM40	: 18-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM
836	001756	020606	DH40	: STARTADR FINISHADR TESTNO ERRORPC
837	001760	024070	DT40	: \$TMP1,\$TMP2,TESTNO,\$ERRPC,0
838	001762	025006	DF40	: 4,4,0,0
839				
840			: ITEM 41	
841	001764	006351	EM41	: 18-BIT MAPPING POSSIBLE HOLE AT THE TOP OF MEMORY
842	001766	020652	DH41	: STARTADR TESTNO ERRORPC
843	001770	024102	DT41	: \$TMP1,TESTNO,\$ERRPC,0
844	001772	025012	DF41	: 4,0,0
845				
846			: ITEM 42	

847	001774	006433	EM42	: FAULTY CARRY PROPAGATION 18-BIT MAPPING.
848	001776	020704	DH42	: PATTERN DATA ADDRESS
849				: LOADED FETCHED INTENDED TESTNO ERRORPC
850	002000	024112	DT42	: \$REG2,\$REG3,\$REG0,TESTNO,\$ERRPC,0
851	002002	025015	DF42	: 0,0,3,0,0
852				
853			: ITEM 43	
854	002004	006504	EM43	: NO TRAP THRU ERRVEC, AT 18-BIT ADDRESS 760000
855	002006	021006	DH43	: TESTNO ERRORPC
856	002010	024126	DT43	: TESTNO,\$ERRPC,0
857	002012	025022	DF43	: 0,0
858				
859			: ITEM 44	
860	002014	006562	EM44	: DIDN'T GET WRAP AROUND TO ADDRESS ZERO
861	002016	021026	DH44	: DATA TESTNO ERRORPC
862	002020	024134	DT44	: \$REG1,TESTNO,\$ERRPC,0
863	002022	025024	DF44	: 0,0,0
864				
865			: ITEM 45	
866	002024	006631	EM45	: NO TRAP THRU ERRVEC, ON NON-EXISTANT ADDRESS
867	002026	021056	DH45	: NON-EXADDR TESTNO ERRORPC
868	002030	024144	DT45	: \$REG0,TESTNO,\$ERRPC,0
869	002032	025027	DF45	: 3,0,0
870				
871			: ITEM 46	
872	002034	006706	EM46	: BAD RELOCATION, CARRY PROPAGATION 22-BIT MAPPING
873	002036	020704	DH42	: PATTERN DATA ADDRESS
874				: LOADED FETCHED INTENDED TESTNO ERRORPC
875	002040	024112	DT42	: \$REG2,\$REG3,\$REG0,TESTNO,\$ERRPC
876	002042	025015	DF42	: 0,0,3,0,0
877				
878			: ITEM 47	
879	002044	006767	EM47	: DID NOT GET UNIBUS ADDRESS
880	002046	020704	DH42	: PATTERN DATA ADDRESS
881				: LOADED FETCHED INTENDED TESTNO ERRORPC
882	002050	024112	DT42	: \$REG2,\$REG3,\$REG0,TESTNO,\$ERRPC,0
883	002052	025015	DF42	: 0,0,3,0,0
884				
885			: ITEM 50	
886	002054	007022	EM50	: COMPARE CIRCUIT FOR TOP OF MEMORY BAD
887	002056	021111	DH50	: KIPAR4 SIZELO TESTNO ERRORPC
888	002060	024154	DT50	: KIPAR4,SIZELO,TESTNO,\$ERRPC,0
889	002062	025032	DF50	: 4,4,0,0
890				
891			: ITEM 51	
892	002064	007070	EM51	: PAGE LENGTH ABORT AT WRONG VIRTUAL ADDRESS
893	002066	021155	DH51	: PGLNFD VABLKNO TESTNO ERRORPC
894	002070	024166	DT51	: \$REG1,\$REG3,TESTNO,\$ERRPC,0
895	002072	025036	DF51	: 0,0,0,0
896				
897			: ITEM 52	
898	002074	007143	EM52	: NO PAGE LENGTH ABORT, WHEN CONDITION CORRECT
899	002076	021155	DH51	: PGLNFD VABLKNO TESTNO ERRORPC
900	002100	024166	DT51	: \$REG1,\$REG3,TESTNO,\$ERRPC,0
901	002102	025036	DF51	: 0,0,0,0
902				

903			:ITEM 53	
904	002104	007220	EM53	:DID NOT ABORT ON NON-RESIDENT PAGE KIPDR5
905	002106	021215	DH53	:TESTNO ERRORPC
906	002110	024200	DT53	:TESTNO,\$ERRPC,0
907	002112	025042	DF53	:0,0
908				
909			:ITEM 54	
910	002114	007272	EM54	:DID NOT ABORT ON READ ONLY PAGE KIPDR4
911	002116	021215	DH53	:TESTNO ERRORPC
912	002120	024200	DT53	:TESTNO,\$ERRPC,0
913	002122	025042	DF53	:0,0
914				
915			:ITEM 55	
916	002124	007341	EM55	:INCORRECT READ FROM PAGE KIPDR4 BIT09 (MMRO) CLEAR
917	002126	021235	DH55	:EXPDATA RECDATA TESTNO ERRORPC
918	002130	024206	DT55	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
919	002132	025044	DF55	:0,0,0,0
920				
921			:ITEM 56	
922	002134	007424	EM56	:NO M.M. TRAP WHEN BIT09 (MMRO) SET PAGE KIPDR4
923	002136	021275	DH56	:(MMRO) KIPDR4 TESTNO ERRORPC
924	002140	024220	DT56	:MMRO,KIPDR4,TESTNO,\$ERRPC,0
925	002142	025050	DF56	:0,0,0,0
926				
927			:ITEM 57	
928	002144	007503	EM57	:INCORRECT READ FROM PAGE KIPDR4, BIT09 (MMRO) SET
929	002146	021235	DH55	:EXPDATA RECDATA TESTNO ERRORPC
930	002150	024206	DT55	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
931	002152	025044	DF55	:0,0,0,0
932				
933			:ITEM 60	
934	002154	007565	EM60	:INCORRECT WRITE TO PAGE KIPDR4, BIT09 (MMRO) SET
935	002156	021215	DH53	:TESTNO ERRORPC
936	002160	024200	DT53	:TESTNO,\$ERRPC,0
937	002162	025042	DF53	:0,0
938				
939			:ITEM 61	
940	002164	007646	EM61	:NO M.M. TRAP WHEN BIT09 (MMRO) SET PAGE KIPDR7
941	002166	021275	DH56	:(MMRO) KIPDR4 TESTNO ERRORPC
942	002170	024220	DT56	:MMRO,KIPDR4,TESTNO,\$ERRPC,0
943	002172	025050	DF56	:0,0,0,0
944				
945			:ITEM 62	
946	002174	007725	EM62	:INCORRECT READ FROM PAGE KIPDR7, BIT09 (MMRO) SET
947	002176	021235	DH55	:EXPDATA RECDATA TESTNO ERRORPC
948	002200	024206	DT55	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
949	002202	025044	DF55	:0,0,0,0
950				
951			:ITEM 63	
952	002204	010007	EM63	:TRAP WHEN NO RELOCATION
953	002206	021215	DH53	:TESTNO ERRORPC
954	002210	024200	DT53	:TESTNO,\$ERRPC,0

Line	Code	Address	Register	Description
955	002212	025042	DF53	:0,0
956				
957			:ITEM 64	
958	002214	010037	EM64	:TOOK TWO VECTORS WITH TRAP AND ABORT ON SAME INSTRUCTION
959				:EXPECTING '1074' FOR KERNEL STACK POINTER
960	002216	021335	DH64	:RECEIVED TESTNO ERRORPC
961	002220	024232	DT64	:STMP0,TESTNO,\$ERRPC,0
962	002222	025054	DF64	:0,0,0
963				
964			:ITEM 65	
965	002224	010202	EM65	:MEMORY MANAGEMENT ABORT CONDITION WRONG
966	002226	021365	DH65	:EXPCOND ABRTCOND TESTNO ERRORPC
967	002230	024242	DT65	:MMEXP,MMR0,TESTNO,\$ERRPC,0
968	002232	025057	DF65	:0,0,0,0
969				
970			:ITEM 66	
971	002234	010252	EM66	:BIT 12 OF MMR0 WAS NOT SET WHEN A.C.F. SATISFIED
972	002236	021425	DH66	:(MMR0) TESTNO ERRORPC
973	002240	024254	DT66	:MMR0,TESTNO,\$ERRPC,0
974	002242	025063	DF66	:0,0,0
975				
976			:ITEM 67	
977	002244	010333	EM67	:NO TRAP WHEN INSTRUCTION CLEARING BIT09 (MMR0) CAUSES TRAP COND
978	002246	021215	DH53	:TESTNO ERRORPC
979	002250	024200	DT53	:TESTNO,\$ERRPC,0
980	002252	025042	DF53	:0,0
981				
982			:ITEM 70	
983	002254	010434	EM70	:ERROR DURING M.M. ABORT IN TRAP SEQUENCE
984				:EXPECTED:
985	002256	021455	DH70	:XXX017 040241 173366 000004
986				:PROSTAT (MMR0) (MMR1) (MMR2) TESTNO ERRORPC
987				:RECEIVED:
988	002260	024264	DT70	:\$REG1,MMR0,MMR1,MMR2,TESTNO,\$ERRPC,0
989	002262	025066	DF70	:0,0,0,0,0,0
990				
991			:ITEM 71	
992	002264	010517	EM71	:INSTRUCTION FETCH DID NOT ABORT IN ILLEGAL MODE (10)
993	002266	021215	DH53	:TESTNO ERRORPC
994	002270	024200	DT53	:TESTNO,\$ERRPC,0
995	002272	025042	DF53	:0,0
996				
997			:ITEM 72	
998	002274	010604	EM72	:ERROR CONDITION NOT CORRECT IN ILLEGAL MODE (10)
999	002276	021606	DH72	:EXPSTAT (MMR0) TESTNO ERRORPC
1000	002300	024302	DT72	:\$REG1,MMR0,TESTNO,\$ERRPC,0
1001	002302	025074	DF72	:0,0,0,0
1002				
1003			:ITEM 73	
1004	002304	010665	EM73	:AT LEAST ONE M.M. STATUS REGISTERS WAS CLOCKED AFTER BEING LOCK
1005	002306	021646	DH73	:ORIGINAL DATA NEW DATA
1006				:(MMR0) (MMR1) (MMR2) (MMR0) (MMR1) (MMR2) TESTNO ERRORPC
1007	002310	024314	DT73	:MMR0,MMR1,MMR2,STMP0,STMP1,STMP2,TESTNO,\$ERRPC,0
1008	002312	025100	DF73	:0,0,0,0,0,0,0,0
1009				
1010			:ITEM 74	

1011	002314	010767	EM74	:DID NOT CHANGE MAPPING TO SUPERVISOR MODE
1012	002316	021215	DH53	:TESTNO ERRORPC
1013	002320	024200	DT53	:TESTNO,\$ERRPC,0
1014	002327	025042	DF53	:0,0
1015				
1016			:ITEM 75	
1017	002324	011041	EM75	:ABORT CONDITION INCORRECT EXPECTING 100051
1018	002326	021777	DH75	:RECEIVD (MMR1) (MMR2) TESTNO ERRORPC
1019	002330	024336	DT75	:PMR0,PMR1,PMR2,TESTNO,\$ERRPC,0
1020	002332	025110	DF75	:0,0,0,0,0
1021				
1022			:ITEM 76	
1023	002334	011114	EM76	:DID NOT CHANGE MAPPING TO USER MODE
1024	002336	021215	DH53	:TESTNO ERRORPC
1025	002340	024200	DT53	:TESTNO,\$ERRPC,0
1026	002342	025042	DF53	:0,0
1027				
1028			:ITEM 77	
1029	002344	011160	EM77	:ABORT CONDITION INCORRECT EXPECTING 100153
1030	002346	021777	DH75	:RECEIVD (MMR1) (MMR2) TESTNO ERRORPC
1031	002350	024336	DT75	:PMR0,PMR1,PMR2,TESTNO,\$ERRPC,0
1032	002352	025110	DF75	:0,0,0,0,0
1033				
1034			:ITEM 100	
1035	002354	011233	EM100	:MMR2 LOCKED UP THE WRONG VIRTUAL ADDRESS
1036	002356	022047	DH100	:VIRTADR (MMR2) TESTNO ERRORPC
1037	002360	024352	DT100	:\$REG1,PMR2,TESTNO,\$ERRPC,0
1038	002362	025115	DF100	:0,0,0,0
1039				
1040			:ITEM 101	
1041	002364	011304	EM101	:MMR0 LOCKED UP THE WRONG PAGE NUMBER
1042	002366	022107	DH101	:EXPECTD (MMR0) TESTNO ERRORPC
1043	002370	024364	DT101	:\$REG2,PMR0,TESTNO,\$ERRPC,0
1044	002372	025121	DF101	:0,0,0,0
1045				
1046			:ITEM 102	
1047	002374	011351	EM102	:W-BIT NOT SET ON WRITE TO PAGE 4
1048	002376	022147	DH102	:KIPDR4 TESTNO ERRORPC
1049	002400	024376	DT102	:\$TMP0,TESTNO,\$ERRPC,0
1050	002402	025125	DF102	:0,0,0
1051				
1052			:ITEM 103	
1053	002404	011412	EM103	:W-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4
1054	002406	022147	DH102	:KIPDR4 TESTNO ERRORPC
1055	002410	024376	DT102	:\$TMP0,TESTNO,\$ERRPC,0
1056	002412	025125	DF102	:0,0,0
1057				
1058			:ITEM 104	
1059	002414	011473	EM104	:W-BIT DID NOT CLEAR ON WRITE TO KIPDR4
1060	002416	022147	DH102	:KIPDR4 TESTNO ERRORPC
1061	002420	024376	DT102	:\$TMP0,TESTNO,\$ERRPC,0
1062	002422	025125	DF102	:0,0,0
1063				
1064			:ITEM 105	
1065	002424	011542	EM105	:A-BIT DID NOT SET ON READ TO PAGE 4 A.C.F.=4
1066	002426	022147	DH102	:KIPDR4 TESTNO ERRORPC

1067	002430	024376	DT102	:STMP0,TESTNO,\$ERRPC,0
1068	002432	025125	DF102	:0,0,0
1069				
1070			:ITEM 106	
1071	002434	011617	EM106	:A-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4
1072	002436	022147	DH102	:KIPDR4 TESTNO ERRORPC
1073	002440	024376	DT102	:STMP0,TESTNO,\$ERRPC,0
1074	002442	025125	DF102	:0,0,0
1075				
1076			:ITEM 107	
1077	002444	011700	EM107	:A-BIT DID NOT CLEAR ON WRITE TO KIPAR4
1078	002446	022147	DH102	:KIPDR4 TESTNO ERRORPC
1079	002450	024376	DT102	:STMP0,TESTNO,\$ERRPC,0
1080	002452	025125	DF102	:0,0,0
1081				
1082			:ITEM 110	
1083	002454	011747	EM110	:W-BIT DID NOT SET ON WRITE TO I/O PAGE
1084	002456	022177	DH110	:KIPDR7 TESTNO ERRORPC
1085	002460	024376	DT102	:STMP0,TESTNO,\$ERRPC,0
1086	002462	025125	DF102	:0,0,0
1087				
1088			:ITEM 111	
1089	002464	012016	EM111	:W-BIT DID NOT REMAIN SET AFTER INTERNAL REGISTER WRITE
1090	002466	022177	DH110	:KIPDR7 TESTNO ERRORPC
1091	002470	024376	DT102	:STMP0,TESTNO,\$ERRPC,0
1092	002472	025125	DF102	:0,0,0
1093				
1094			:ITEM 112	
1095	002474	012105	EM112	:DUAL MAPPING BETWEEN PAGES
1096	002476	022227	DH112	:GDPAGE BDPAGE TESTNO ERRORPC
1097	002500	024406	DT112	:\$REG3,\$REG1,TESTNO,\$ERRPC,0
1098	002502	025130	DF112	:0,0,0,0
1099				
1100			:ITEM 113	
1101	002504	012140	EM113	:NO PAGE HAD BOTH ITS A & W BITS SET
1102	002506	022267	DH113	:TSTPAGE CONTENT TESTNO ERRORPC
1103	002510	024420	DT113	:\$REG3,\$REG0,TESTNO,\$ERRPC,0
1104	002512	025134	DF113	:0,0,0,0
1105				
1106			:ITEM 114	
1107	002514	012204	EM114	:DID NOT PICK UP CORRECT STACK POINTER
1108	002516	022327	DH114	:EXPECTD RECEIVD TESTNO ERRORPC
1109	002520	024432	DT114	:\$REG2,\$REG1,TESTNO,\$ERRPC,0
1110	002522	025140	DF114	:0,0,0,0
1111				
1112			:ITEM 115	
1113	002524	012252	EM115	:CURRENT MODE STACK NOT PUSHED IN MFP INSTRUCTION
1114	002526	021215	DH53	:TESTNO ERRORPC
1115	002530	024200	DT53	:TESTNO,\$ERRPC,0
1116	002532	025042	DF53	:0,0
1117				
1118			:ITEM 116	
1119	002534	012333	EM116	:WRONG DATA FETCHED BY MFP INSTRUCTION
1120	002536	022367	DH116	:EXPECTD RECEIVD TESTNO ERRORPC
1121	002540	024444	DT116	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
1122	002542	025144	DF116	:0,0,0,0

1123				
1124			:ITEM 117	
1125	002544	012401	EM117	:TRIED TO REFERENCE NON-RESIDENT PAGE
1126	002546	022427	DH117	:(MMR0) (MMR1) (MMR2) TESTNO ERRORPC
1127	002550	024456	DT117	:PMR0,PMR1,PMR2,TESTNO,\$ERRPC,0
1128	002552	025150	DF117	:0,0,0,0,0
1129				
1130			:ITEM 120	
1131	002554	012446	EM120	:STACK POINTER NOT CHANGED BY MTP INSTRUCTION
1132	002556	022477	DH120	:STKPTR TESTNO ERRORPC
1133	002560	024472	DT120	:\$REG1,TESTNO,\$ERRPC,0
1134	002562	025155	DF120	:0,0,0
1135				
1136			:ITEM 121	
1137	002564	012523	EM121	:INCORRECT STORE BY MTP INSTRUCTION
1138	002566	022527	DH121	:GDDATA STORED TESTNO ERRORPC
1139	002570	024444	DT116	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
1140	002572	025144	DF116	:0,0,0,0
1141				
1142			:ITEM 122	
1143	002574	012566	EM122	:ABORTED THRU RIGHT VECTOR BUT PICKED UP WRONG PSW
1144	002576	022567	DH122	:(PSW) TESTNO ERRORPC EXPECTING XXX340
1145	002600	024502	DT122	:\$REG0,TESTNO,\$ERRPC,0
1146	002602	025160	DF122	:0,0,0
1147				
1148			:ITEM 123	
1149	002604	012650	EM123	:ABORTED THRU SUPERVISOR SPACE, SUPERVISOR PSW IS XXX140
1150	002606	022640	DH123	:(PSW) TESTNO ERRORPC
1151	002610	024502	DT122	:\$REG0,TESTNO,\$ERRPC,0
1152	002612	025160	DF122	:0,0,0
1153				
1154			:ITEM 124	
1155	002614	012740	EM124	:ABORTED THRU USER SPACE, USER PSW IS XXX000
1156	002616	022640	DH123	:(PSW) TESTNO ERRORPC
1157	002620	024502	DT122	:\$REG0,TESTNO,\$ERRPC,0
1158	002622	025160	DF122	:0,0,0
1159				
1160			:ITEM 125	
1161	002624	013014	EM125	:22-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM
1162	002626	020606	DH40	:STARTADR FINISHADR TESTNO ERRORPC
1163	002630	024070	DT40	:\$TMP1,\$TMP2,TESTNO,\$ERRPC,0
1164	002632	025006	DF40	:4,4,0,0
1165				
1166			:ITEM 126	
1167	002634	013075	EM126	:22-BIT MAPPING POSSIBLE HOLE AT THE TOP OF MEMORY
1168	002636	020652	DH41	:STARTADR TESTNO ERRORPC
1169	002640	024102	DT41	:\$TMP1,TESTNO,\$ERRPC,0
1170	002642	025012	DF41	:4,0,0
1171				
1172			:ITEM 127	
1173	002644	013157	EM127	:DID NOT ABORT REFERENCE TO A MEMORY MANAGEMENT REGISTER
1174	002646	022670	DH127	:TESTNO ERRORPC
1175	002650	024512	DT127	:TESTNO,\$ERRPC,0
1176	002652	025163	DF127	:0,0
1177				
1178			:ITEM 130	

1179	002654	013247	EM130	:ABORT IN KERNEL D-SPACE PICKED UP VECTOR FROM I-SPACE
1180	002656	022567	DH122	:(PSW) TESTNO ERRORPC EXPECTING XXX340
1181	002660	024502	DT122	:\$REG0,TESTNO,\$ERRPC,0
1182	002662	025160	DF122	:0,0,0
1183				
1184			:ITEM 131	
1185	002664	013335	EM131	:M.M. ABORT IN KERNEL D-SPACE HAD WRONG CONDITION
1186	002666	022707	DH131	:(MMR0) (MMR1) (MMR2) TESTNO ERRORPC EXPECTING 020031
1187	002670	024520	DT131	:\$MMR0,\$MMR1,\$MMR2,TESTNO,\$ERRPC,0
1188	002672	025165	DF131	:0,0,0,0,0
1189				
1190			:ITEM 132	
1191	002674	013416	EM132	:D SPACE ENABLE CIRCUITRY HAS FAILED
1192	002676	017177	DH5	:ERROR AUTOI/D VIRTUAL
1193				:REGISTR REGISTR ADDRESS TESTNO PC AT ABORT
1194	002700	023562	DT5	:\$MMR0,\$MMR1,\$MMR2,TESTNO,BADPC,0
1195	002702	024662	DF5	:0,0,0,0,0
1196			:ITEM 133	
1197	002704	013462	EM133	:BYP BIT IN KIPDR COULD NOT BE CLEARED
1198	002706	023000	DH133	: PC KIPDR (KIPDR)
1199	002710	024534	DT133	:\$ERRPC,\$REG0,\$REG1,0
1200	002712	024651	DF2	:0,0,0
1201			:ITEM 134	
1202	002714	013530	EM134	:BYP BIT IN KIPDR COULD NOT BE SET
1203	002716	023000	DH133	
1204	002720	024534	DT133	
1205	002722	024651	DF2	
1206				
1207			:ITEM 135	
1208	002724	013572	EM135	:TEST DATA COULD NOT BE MADE HIT
1209	002726	023027	DH135	: PC CCR PARADR PAR PDR TST-DATA-ADR
1210	002730	024544	DT135	:\$ERRPC,\$REG0,\$REG1,\$REG2,\$REG3,\$REG4,0
1211	002732	024667	DF6	
1212				
1213			:ITEM 136	
1214	002734	013632	EM136	:TEST DATA REFERENCE NOT A MISS
1215				:CACHED DATA WAS NOT FORCED A MISS ON VIRTUAL PAGE BYPASS
1216	002736	023027	DH135	
1217	002740	024544	DT135	
1218	002742	024667	DF6	
1219				
1220			:ITEM 137	
1221	002744	013763	EM137	:TEST DATA REFERENCE NOT A MISS
1222				:CACHED DATA WAS NOT INVALIDATED ON VIRTUAL BYPASS
1223	002746	023027	DH135	
1224	002750	024544	DT135	
1225	002752	024667	DF6	
1226			:ITEM 140	
1227	002754	014112	EM140	:BYP BIT IN SIPDR COULD NOT BE CLEARED
1228	002756	023116	DH140	: PC SIPDR (SIPDR)
1229	002760	024534	DT133	
1230	002762	024651	DF2	
1231				
1232			:ITEM 141	
1233	002764	014160	EM141	:BYP IN SIPDR COULD NOT BE SET
1234	002766	023116	DH140	

1235	002770	024534	DT133	
1236	002772	024651	DF2	
1237				
1238			: ITEM 142	
1239	002774	014222	EM142	:BYP BIT IN UIPDR COULD NOT BE CLEARED
1240	002776	023145	DH142	: PC UIPDR (UIPDR)
1241	003000	024534	DT133	
1242	003002	024651	DF2	
1243				
1244			: ITEM 143	
1245	003004	014270	EM143	:BYP BIT IN UIPDR COULD NOT BE SET
1246	003006	023145	DH142	
1247	003010	024534	DT133	
1248	003012	024651	DF2	
1249				
1250			: ITEM 144	
1251	003014	005606	EM32M	:MICRO BREAK REG LOADED INCORECTLY. 16 BITS WRITABLE
1252	003016	020336	DH31	:REGADDR DATAREC DATAEXP TESTNO ERRORPC
1253	003020	024014	DT31	:SREG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0
1254	003022	024764	DF31	:0,0,0,0,0
1255				
1256			: ITEM 145	
1257	003024	014332	EM145	:DPAR READ BACK INCORRECTLY
1258	003026	023174	DH145	:ADDR EXP REC TEST ERR
1259	003030	024562	DT145	:\$TMP0,\$TMP1,\$TMP2,TESTNO,ERRORPC,0
1260	003032	024764	DF31	:0,0,0,0,0
1261				
1262			: ITEM 146	
1263	003034	014461	EM146	:NO KT ABORT
1264	003036	023243	DH146	
1265	003040	024576	DT146	
1266	003042	025163	DF127	
1267				
1268			: ITEM 147	
1269	003044	014637	EM147	:PS<08> FAILED TO SET
1270	003046	023243	DH146	
1271	003050	024576	DT146	
1272	003052	025163	DF127	
1273				
1274			: ITEM 150	
1275	003054	014671	EM150	:KT ABORT TRAPPED TO 4
1276	003056	023243	DH146	
1277	003060	024576	DT146	
1278	003062	025163	DF127	
1279				
1280			: ITEM 151	
1281	003064	014775	EM151	:KT ABORT TRAPPED TO 10
1282	003066	023243	DH146	
1283	003070	024576	DT146	
1284	003072	025163	DF127	
1285				
1286			: ITEM 152	
1287	003074	015044	EM152	:KT ABORT TRAPPED TO 240
1288	003076	023243	DH146	
1289	003100	024576	DT146	
1290	003102	025163	DF127	

1291				
1292			:ITEM 153	
1293	003104	015110	EM153	;KT TRAP DID NOT WORK
1294	003106	023243	DH146	
1295	003110	024576	DT146	
1296	003112	025163	DF127	
1297				
1298			:ITEM 154	
1299	003114	000000	0	;DELETED
1300	003116	000000	0	
1301	003120	000000	0	
1302	003122	000000	0	
1303				
1304			:ITEM 155	
1305	003124	015217	EM155	;NO KT TRAP ON SOURCE OPERAND
1306	003126	023243	DH146	
1307	003130	024576	DT146	
1308	003132	025163	DF127	
1309				
1310			:ITEM 156	
1311	003134	015337	EM156	;NO ABORT ON NEXM
1312	003136	023243	DH146	
1313	003140	024576	DT146	
1314	003142	025163	DF127	
1315				
1316			:ITEM 157	
1317	003144	000000	0	;DELETED
1318	003146	000000	0	
1319	003150	000000	0	
1320	003152	000000	0	
1321				
1322			:ITEM 160	
1323	003154	015472	EM160	;NEXM BIT DID NOT SET IN CPUERR REG
1324	003156	023267	DH147	
1325	003160	024604	DT147	
1326	003162	025130	DF112	
1327				
1328			:ITEM 161	
1329	003164	015600	EM161	;NEXM BIT DID NOT CLEAR IN CPUERR REG
1330	003166	023243	DH146	
1331	003170	024576	DT146	
1332	003172	025163	DF127	
1333				
1334			:ITEM 162	
1335	003174	015651	EM162	;KT ABORT ON NEXM (KB11-B/C)
1336	003176	023243	DH146	
1337	003200	024576	DT146	
1338	003202	025163	DF127	
1339				
1340			:ITEM 163	
1341	003204	015730	EM163	;KT ABORT ON SL RED
1342	003206	023243	DH146	
1343	003210	024576	DT146	
1344	003212	025163	DF127	
1345				
1346			:ITEM 164	

1347	003214	016005	EM164	:KT ABORT ON ODD ADDRESS
1348	003216	023243	DH146	
1349	003220	024576	DT146	
1350	003222	025163	DF127	
1351				
1352			:ITEM 165	
1353	003224	016070	EM165	:KT ABORT FAILED TO OVER-RIDE NEXM (KB11-E/EM)
1354	003226	023243	DH146	
1355	003230	024576	DT146	
1356	003232	025163	DF127	
1357				
1358			:ITEM 166	
1359	003234	016154	EM166	:TMCE CACHE BEND DID NOT GO
1360	003236	023243	DH146	
1361	003240	024576	DT146	
1362	003242	025163	DF127	
1363				
1364			:ITEM 167	
1365	003244	016231	EM167	:
1366	003246	023342	DH150	
1367	003250	000000	0	
1368	003252	000000	0	
1369				
1370			:ITEM 170	
1371	003254	016425	EM170	:BEN 6 FAILED ON PS RESTORE
1372	003256	023243	DH146	
1373	003260	024576	DT146	
1374	003262	025163	DF127	
1375				
1376			:ITEM 171	
1377	003264	016521	EM171	:GOING TO NEXT TEST
1378	003266	000000	0	
1379	003270	000000	0	
1380	003272	000000	0	
1381	003274		ER200:	:STARTING ADDRESS FOR ITEM 201-377
1382				
1383			:ITEM 201	
1384	003274	016546	EM201	:THE FOLLOWING ARE PAR/PDR REFERENCE TIMEOUTS
1385	003276	023370	DH201	:ADDRESS TESTNO
1386				
1387			:ITEM 301	
1388	003300	024616	DT201	:SREG0,,TESTNO,0
1389	003302	025172	DF201	:0,0
1390				
1391			:ITEM 202	
1392	003304	016623	EM202	:THE FOLLOWING ARE DUAL ADDRESSING ERRORS FOR PAR'S/PDR'S
1393	003306	023407	DH202	:ADDRESS ADDRESS
1394				:LOADED JUST READ TESTNO
1395				
1396			:ITEM 302	
1397	003310	024624	DT202	:SREG0,SREG1,TESTNO,0
1398	003312	025174	DF202	:0,0,0
1399				
1400			:ITEM 203	
1401	003314	016714	EM203	:THE FOLLOWING ARE COUNT PATTERN ERRORS FOR PAR'S/PDR'S
1402	003316	023457	DH203	:ADDRESS DATA RED PATTERN COUNT TESTNO

```

1403
1404
1405 003320 024634      :ITEM 303
1406 003322 025177      DT203      :$REG0,$REG2,$REG4,$REG1,TESTNO,0
                                DF203      :0,0,0,0,0
1407
1408
1409      :MESSAGES
1410
1411 003324 041600 052520 052440 MSG1: .ASCIZ<CRLF>  'CPU UNDER TEST FOUND TO BE A '
1412 003332 042116 051105 052040
1413 003340 051505 020124 047506
1414 003346 047125 020104 047524
1415 003354 041040 020105 020101
1416 003362      000
1417 003363      113 030502 026461 MSG2: .ASCIZ  'KB11-EM'<CRLF>
1418 003370 046505 000200
1419 003374 041113 030461 041055 MSG3: .ASCIZ  'KB11-B/C'<CRLF>
1420 003402 041457 000200
1421 003406 041113 030461 041455 MSG4: .ASCIZ  'KB11-CM          '<CRLF>
1422 003414 020115 020040 020040
1423 003422 020040 020040 020040
1424 003430 020040 020040 100040
1425 003436      000
1426 003437      113 030502 026461 MSG5: .ASCIZ  'KB11-E'<CRLF>
1427 003444 100105      000
1428
1429      .SBTTL  ERROR TABLE MESSAGES AND DATA POINTERS
1430 003447      116 052117 052040 EM1: .ASCIZ  ?NOT THE CORRECT CPU ABORT CONDITION THRU 'ERRVEC' (004)?
1431 003454 042510 041440 051117
1432 003462 042522 052103 041440
1433 003470 052520 040440 047502
1434 003476 052122 041440 047117
1435 003504 044504 044524 047117
1436 003512 052040 051110 020125
1437 003520 042447 051122 042526
1438 003526 023503 024040 030060
1439 003534 024464      000
1440 003537      125 042516 050130 EM2: .ASCIZ  ?UNEXPECTED CPU TRAP OR ABORT THRU 'ERRVEC' (004)?
1441 003544 041505 042524 020104
1442 003552 050103 020125 051124
1443 003560 050101 047440 020122
1444 003566 041101 051117 020124
1445 003574 044124 052522 023440
1446 003602 051105 053122 041505
1447 003610 020047 030050 032060
1448 003616 000051
1449 003620 047125 054105 042520 EM3: .ASCII  ?UNEXPECTED CACHE PARITY ERROR THRU 'CACHVEC' (114)?<CRLF>
1450 003626 052103 042105 041440
1451 003634 041501 042510 050040
1452 003642 051101 052111 020131
1453 003650 051105 047522 020122
1454 003656 044124 052522 023440
1455 003664 040503 044103 042526
1456 003672 023503 024040 030461
1457 003700 024464      200
1458 003703      127 046111 020114      .ASCIZ  ?WILL RETRY THIS TEST ONCE.?
    
```

1459	003710	042522	051124	020131	
1460	003716	044124	051511	052040	
1461	003724	051505	020124	047117	
1462	003732	042503	000056		
1463	003736	047125	054105	042520	EM4: .ASCII ?UNEXPECTED MAIN MEMORY PARITY ERROR THRU 'CACHVEC' (114)?<CRLF>
1464	003744	052103	042105	046440	
1465	003752	044501	020116	042515	
1466	003760	047515	054522	050040	
1467	003766	051101	052111	020131	
1468	003774	051105	047522	020122	
1469	004002	044124	052522	023440	
1470	004010	040503	044103	042526	
1471	004016	023503	024040	030461	
1472	004024	024464	200		
1473	004027	127	046111	020114	.ASCIZ ?WILL RETRY THIS TEST ONCE?
1474	004034	042522	051124	020131	
1475	004042	044124	051511	052040	
1476	004050	051505	020124	047117	
1477	004056	042503	000		
1478					
1479	004061	125	042516	050130	EM5: .ASCIZ ?UNEXPECTED MEMORY MANAGEMENT TRAP OR ABORT?
1480	004066	041505	042524	020104	
1481	004074	042515	047515	054522	
1482	004102	046440	047101	043501	
1483	004110	046505	047105	020124	
1484	004116	051124	050101	047440	
1485	004124	020122	041101	051117	
1486	004132	000124			
1487	004134	042515	047515	054522	EM6: .ASCIZ ?MEMORY MANAGEMENT TRAP OR ABORT HAD INCORRECT CONDITION?
1488	004142	046440	047101	043501	
1489	004150	046505	047105	020124	
1490	004156	051124	050101	047440	
1491	004164	020122	041101	051117	
1492	004172	020124	040510	020104	
1493	004200	047111	047503	051122	
1494	004206	041505	020124	047503	
1495	004214	042116	052111	047511	
1496	004222	000116			
1497	004224	042515	047515	054522	EM7: .ASCIZ ?MEMORY MANAGEMENT REGISTER 0 WILL NOT CLEAR?
1498	004232	046440	047101	043501	
1499	004240	046505	047105	020124	
1500	004246	042522	044507	052123	
1501	004254	051105	030040	053440	
1502	004262	046111	020114	047516	
1503	004270	020124	046103	040505	
1504	004276	000122			
1505	004300	040503	023516	020124	EM10: .ASCIZ ?CAN'T SET 171000 IN MMR0?
1506	004306	042523	020124	033461	
1507	004314	030061	030060	044440	
1508	004322	020116	046515	030122	
1509	004330	000			
1510	004331	107	052117	052040	EM11: .ASCIZ ?GOT THE WRONG DATA BACK FROM MMR0?
1511	004336	042510	053440	047522	
1512	004344	043516	042040	052101	
1513	004352	020101	040502	045503	
1514	004360	043040	047522	020115	

1515	004366	046515	030122	000	
1516	004373	115	046505	051117	EM12: .ASCIZ ?MEMORY MANAGEMENT REGISTER 1 WILL NOT CLEAR?
1517	004400	020131	040515	040516	
1518	004406	042507	042515	052116	
1519	004414	051040	043505	051511	
1520	004422	042524	020122	020061	
1521	004430	044527	046114	047040	
1522	004436	052117	041440	042514	
1523	004444	051101	000		
1524	004447	115	051115	020061	EM13: .ASCIZ ?MMR1 DID NOT TRACK PROPERLY?
1525	004454	044504	020104	047516	
1526	004462	020124	051124	041501	
1527	004470	020113	051120	050117	
1528	004476	051105	054514	000	
1529	004503	115	051115	020062	EM14: .ASCIZ ?MMR2 DID NOT TRACK PROPERLY?
1530	004510	044504	020104	047516	
1531	004516	020124	051124	041501	
1532	004524	020113	051120	050117	
1533	004532	051105	054514	000	
1534	004537	115	046505	051117	EM15: .ASCIZ ?MEMORY MANAGEMENT REGISTER 3 WILL NOT CLEAR?
1535	004544	020131	040515	040516	
1536	004552	042507	042515	052116	
1537	004560	051040	043505	051511	
1538	004566	042524	020122	020063	
1539	004574	044527	046114	047040	
1540	004602	052117	041440	042514	
1541	004610	051101	000		
1542	004613	115	051115	020063	EM16: .ASCIZ ?MMR3 IS HOLDING WRONG DATA?
1543	004620	051511	044040	046117	
1544	004626	044504	043516	053440	
1545	004634	047522	043516	042040	
1546	004642	052101	000101		
1547	004646	052523	046515	051101	EM17: .ASCIZ ?SUMMARY OF PAR/PDR REFERENCE TIMEOUTS?
1548	004654	020131	043117	050040	
1549	004662	051101	050057	051104	
1550	004670	051040	043105	051105	
1551	004676	047105	042503	052040	
1552	004704	046511	047505	052125	
1553	004712	000123			
1554	004714	047506	046114	053517	EM20: .ASCIZ ?FOLLOWING PAR/PDR WILL NOT CLEAR?
1555	004722	047111	020107	040520	
1556	004730	027522	042120	020122	
1557	004736	044527	046114	047040	
1558	004744	052117	041440	042514	
1559	004752	051101	000		
1560	004755	123	046525	040515	EM21: .ASCIZ ?SUMMARY OF DUAL ADDRESSING ERRORS?
1561	004762	054522	047440	020106	
1562	004770	052504	046101	040440	
1563	004776	042104	042522	051523	
1564	005004	047111	020107	051105	
1565	005012	047522	051522	000	
1566	005017	123	046525	040515	EM22: .ASCIZ ?SUMMARY OF COUNT PATTERN FAILURES?
1567	005024	054522	047440	020106	
1568	005032	047503	047125	020124	
1569	005040	040520	052124	051105	
1570	005046	020116	040506	046111	

1571	005054	051125	051505	000	
1572	005061	105	051122	051117	EM23: .ASCIZ ?ERROR IN BYTE ADDRESSING OF PAR/PDR?
1573	005066	044440	020116	054502	
1574	005074	042524	040440	042104	
1575	005102	042522	051523	047111	
1576	005110	020107	043117	050040	
1577	005116	051101	050057	051104	
1578	005124	000			
1579	005125	117	042516	047440	EM24: .ASCIZ ?ONE OF THE REGISTERS TIMED OUT?
1580	005132	020106	044124	020105	
1581	005140	042522	044507	052123	
1582	005146	051105	020123	044524	
1583	005154	042515	020104	052517	
1584	005162	000124			
1585	005164	047514	020127	054502	EM25: .ASCIZ ?LOW BYTE OF LOW SIZE REGISTER IS NOT ALL ONES?
1586	005172	042524	047440	020106	
1587	005200	047514	020127	044523	
1588	005206	042532	051040	043505	
1589	005214	051511	042524	020122	
1590	005222	051511	047040	052117	
1591	005230	040440	046114	047440	
1592	005236	042516	000123		
1593	005242	047503	046125	020104	EM26: .ASCIZ ?COULD WRITE ONE OF THE SIZE REGISTERS?
1594	005250	051127	052111	020105	
1595	005256	047117	020105	043117	
1596	005264	052040	042510	051440	
1597	005272	055111	020105	042522	
1598	005300	044507	052123	051105	
1599	005306	000123			
1600	005310	044510	044107	051440	EM27: .ASCIZ ?HIGH SIZE REGISTER IS NOT ZERO?
1601	005316	055111	020105	042522	
1602	005324	044507	052123	051105	
1603	005332	044440	020123	047516	
1604	005340	020124	042532	047522	
1605	005346	000			
1606	005347	103	052520	042440	EM30: .ASCIZ ?CPU ERROR REGISTER NOT ZERO AFTER LOADING NEGATIVE ONE?
1607	005354	051122	051117	051040	
1608	005362	043505	051511	042524	
1609	005370	020122	047516	020124	
1610	005376	042532	047522	040440	
1611	005404	052106	051105	046040	
1612	005412	040517	044504	043516	
1613	005420	047040	043505	052101	
1614	005426	053111	020105	047117	
1615	005434	000105			
1616	005436	047514	042527	020122	EM31: .ASCIZ ?LOWER BYTE OF P.S.W. NOT CORRECT?
1617	005444	054502	042524	047440	
1618	005452	020106	027120	027123	
1619	005460	027127	047040	052117	
1620	005466	041440	051117	042522	
1621	005474	052103	000		
1622	005477	115	041511	030122	EM32: .ASCIZ ?MICRO BREAK REG LOADED INCORRECTLY. ONLY LOWER BYTE SHOULD BE WRITABLE?
1623	005504	041040	042522	045501	
1624	005512	051040	043505	046040	
1625	005520	040517	042504	020104	
1626	005526	047111	047503	051122	

1627	005534	041505	046124	027131
1628	005542	047440	046116	020131
1629	005550	047514	042527	020122
1630	005556	054502	042524	051440
1631	005564	047510	046125	020104
1632	005572	042502	053440	044522
1633	005600	040524	046102	000105
1634	005606	044515	051103	020060
1635	005614	051102	040505	020113
1636	005622	042522	020107	047514
1637	005630	042101	042105	044440
1638	005636	041516	051117	042522
1639	005644	052103	054514	020056
1640	005652	046101	020114	033061
1641	005660	041040	052111	020123
1642	005666	044123	052517	042114
1643	005674	041040	020105	051127
1644	005702	052111	041101	042514
1645	005710	000		
1646	005711	104	042111	023516
1647	005716	020124	047514	042101
1648	005724	050040	047522	051107
1649	005732	046501	044440	052116
1650	005740	051105	052522	052120
1651	005746	051040	050505	042525
1652	005754	052123	051040	043505
1653	005762	051511	042524	000122
1654	005770	047514	042101	042105
1655	005776	046440	051117	020105
1656	006004	044124	047101	052440
1657	006012	050120	051105	041040
1658	006020	052131	020105	043117
1659	006026	051440	040524	045503
1660	006034	046040	046511	052111
1661	006042	051040	043505	051511
1662	006050	042524	000122	
1663	006054	042513	047122	046105
1664	006062	051440	040524	045503
1665	006070	050040	044517	052116
1666	006076	051105	047040	052117
1667	006104	030440	030061	020060
1668	006112	043101	042524	020122
1669	006120	047514	042101	047111
1670	006126	020107	051523	020120
1671	006134	020046	051525	000120
1672	006142	052504	046101	040440
1673	006150	042104	042522	051523
1674	006156	047111	020107	042502
1675	006164	053524	042505	020116
1676	006172	040520	027522	042120
1677	006200	020122	051107	052517
1678	006206	051520	000	
1679	006211	102	042101	051040
1680	006216	046105	041517	052101
1681	006224	047511	026116	047440
1682	006232	020116	052123	051117

EM32M: .ASCIZ ?MICRO BREAK REG LOADED INCORRECTLY. ALL 16 BITS SHOULD BE WRITABLE?

EM33: .ASCIZ ?DIDN'T LOAD PROGRAM INTERRUPT REQUEST REGISTER?

EM34: .ASCIZ ?LOADED MORE THAN UPPER BYTE OF STACK LIMIT REGISTER?

EM35: .ASCIZ ?KERNEL STACK POINTER NOT 1100 AFTER LOADING SSP & USP?

EM36: .ASCIZ ?DUAL ADDRESSING BETWEEN PAR/PDR GROUPS?

EM37: .ASCIZ ?BAD RELOCATION, ON STORING DATA 18-BIT MAPPING?

1683	006240	047111	020107	040504	
1684	006246	040524	030440	026470	
1685	006254	044502	020124	040515	
1686	006262	050120	047111	000107	
1687	006270	034061	041055	052111	EM40: .ASCIZ ?18-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM?
1688	006276	046440	050101	044520	
1689	006304	043516	050040	051517	
1690	006312	044523	046102	020105	
1691	006320	047510	042514	044440	
1692	006326	020116	040515	047111	
1693	006334	046440	046505	051117	
1694	006342	020131	051106	046517	
1695	006350	000			
1696	006351	061	026470	044502	EM41: .ASCIZ ?18-BIT MAPPING POSSIBLE HOLE AT THE TOP OF MEMORY?
1697	006356	020124	040515	050120	
1698	006364	047111	020107	047520	
1699	006372	051523	041111	042514	
1700	006400	044040	046117	020105	
1701	006406	052101	052040	042510	
1702	006414	052040	050117	047440	
1703	006422	020106	042515	047515	
1704	006430	054522	000		
1705	006433	106	052501	052114	EM42: .ASCIZ ?FAULTY CARRY PROPAGATION 18-BIT MAPPING.?
1706	006440	020131	040503	051122	
1707	006446	020131	051120	050117	
1708	006454	043501	052101	047511	
1709	006462	020116	034061	041055	
1710	006470	052111	046440	050101	
1711	006476	044520	043516	000056	
1712	006504	047516	052040	040522	EM43: .ASCIZ ?NO TRAP THRU ERRVEC, AT 18-BIT ADDRESS 760000?
1713	006512	020120	044124	052522	
1714	006520	042440	051122	042526	
1715	006526	026103	040440	020124	
1716	006534	034061	041055	052111	
1717	006542	040440	042104	042522	
1718	006550	051523	033440	030066	
1719	006556	030060	000060		
1720	006562	044504	047104	052047	EM44: .ASCIZ ?DIDN'T GET WRAP AROUND TO ADDRESS ZERO?
1721	006570	043440	052105	053440	
1722	006576	040522	020120	051101	
1723	006604	052517	042116	052040	
1724	006612	020117	042101	051104	
1725	006620	051505	020123	042532	
1726	006626	047522	000		
1727	006631	116	020117	051124	EM45: .ASCIZ ?NO TRAP THRU ERRVEC, ON NON-EXISTANT ADDRESS?
1728	006636	050101	052040	051110	
1729	006644	020125	051105	053122	
1730	006652	041505	020054	047117	
1731	006660	047040	047117	042455	
1732	006666	044530	052123	047101	
1733	006674	020124	042101	051104	
1734	006702	051505	000123		
1735	006706	040502	020104	042522	EM46: .ASCIZ ?BAD RELOCATION, CARRY PROPAGATION 22-BIT MAPPING?
1736	006714	047514	040503	044524	
1737	006722	047117	020054	040503	
1738	006730	051122	020131	051120	

1739	006736	050117	043501	052101		
1740	006744	047511	020116	031062		
1741	006752	041055	052111	046440		
1742	006760	050101	044520	043516		
1743	006766	000				
1744	006767	104	042111	047040	EM47:	.ASCIZ ?DID NOT GET UNIBUS ADDRESS?
1745	006774	052117	043440	052105		
1746	007002	052440	044516	052502		
1747	007010	020123	042101	051104		
1748	007016	051505	000123			
1749	007022	047503	050115	051101	EM50:	.ASCIZ ?COMPARE CIRCUIT FOR TOP OF MEMORY BAD?
1750	007030	020105	044503	041522		
1751	007036	044525	020124	047506		
1752	007044	020122	047524	020120		
1753	007052	043117	046440	046505		
1754	007060	051117	020131	040502		
1755	007066	000104				
1756	007070	040520	042507	046040	EM51:	.ASCIZ ?PAGE LENGTH ABORT AT WRONG VIRTUAL ADDRESS?
1757	007076	047105	052107	020110		
1758	007104	041101	051117	020124		
1759	007112	052101	053440	047522		
1760	007120	043516	053040	051111		
1761	007126	052524	046101	040440		
1762	007134	042104	042522	051523		
1763	007142	000				
1764	007143	116	020117	040520	EM52:	.ASCIZ ?NO PAGE LENGTH ABORT, WHEN CONDITION CORRECT?
1765	007150	042507	046040	047105		
1766	007156	052107	020110	041101		
1767	007164	051117	026124	053440		
1768	007172	042510	020116	047503		
1769	007200	042116	052111	047511		
1770	007206	020116	047503	051122		
1771	007214	041505	000124			
1772	007220	044504	020104	047516	EM53:	.ASCIZ ?DID NOT ABORT ON NON-RESIDENT PAGE KIPDR5?
1773	007226	020124	041101	051117		
1774	007234	020124	047117	047040		
1775	007242	047117	051055	051505		
1776	007250	042111	047105	020124		
1777	007256	040520	042507	045440		
1778	007264	050111	051104	000065		
1779	007272	044504	020104	047516	EM54:	.ASCIZ ?DID NOT ABORT ON READ ONLY PAGE KIPDR4?
1780	007300	020124	041101	051117		
1781	007306	020124	047117	051040		
1782	007314	040505	020104	047117		
1783	007322	054514	050040	043501		
1784	007330	020105	044513	042120		
1785	007336	032122	000			
1786	007341	111	041516	051117	EM55:	.ASCIZ ?INCORRECT READ FROM PAGE KIPDR4 BIT09 (MMRO) CLEAR?
1787	007346	042522	052103	051040		
1788	007354	040505	020104	051106		
1789	007362	046517	050040	043501		
1790	007370	020105	044513	042120		
1791	007376	032122	041040	052111		
1792	007404	034460	024040	046515		
1793	007412	030122	020051	046103		
1794	007420	040505	000122			

1795	007424	047516	046440	046456	EM56: .ASCIZ ?NO M.M. TRAP WHEN BIT09 (MMRO) SET PAGE KIPDR4?
1796	007432	020056	051124	050101	
1797	007440	053440	042510	020116	
1798	007446	044502	030124	020071	
1799	007454	046450	051115	024460	
1800	007462	051440	052105	050040	
1801	007470	043501	020105	044513	
1802	007476	042120	032122	000	
1803	007503	111	041516	051117	EM57: .ASCIZ ?INCORRECT READ FROM PAGE KIPDR4, BIT09 (MMRO) SET?
1804	007510	042522	052103	051040	
1805	007516	040505	020104	051106	
1806	007524	046517	050040	043501	
1807	007532	020105	044513	042120	
1808	007540	032122	020054	044502	
1809	007546	030124	020071	046450	
1810	007554	051115	024460	051440	
1811	007562	052105	000		
1812	007565	111	041516	051117	EM60: .ASCIZ ?INCORRECT WRITE TO PAGE KIPDR4, BIT09 (MMRO) SET?
1813	007572	042522	052103	053440	
1814	007600	044522	042524	052040	
1815	007606	020117	040520	042507	
1816	007614	045440	050111	051104	
1817	007622	026064	041040	052111	
1818	007630	034460	024040	046515	
1819	007636	030122	020051	042523	
1820	007644	000124			
1821	007646	047516	046440	046456	EM61: .ASCIZ ?NO M.M. TRAP WHEN BIT09 (MMRO) SET PAGE KIPDR7?
1822	007654	020056	051124	050101	
1823	007662	053440	042510	020116	
1824	007670	044502	030124	020071	
1825	007676	046450	051115	024460	
1826	007704	051440	052105	050040	
1827	007712	043501	020105	044513	
1828	007720	042120	033522	000	
1829	007725	111	041516	051117	EM62: .ASCIZ ?INCORRECT READ FROM PAGE KIPDR7, BIT09 (MMRO) SET?
1830	007732	042522	052103	051040	
1831	007740	040505	020104	051106	
1832	007746	046517	050040	043501	
1833	007754	020105	044513	042120	
1834	007762	033522	020054	044502	
1835	007770	030124	020071	046450	
1836	007776	051115	024460	051440	
1837	010004	052105	000		
1838	010007	124	040522	020120	EM63: .ASCIZ ?TRAP WHEN NO RELOCATION?
1839	010014	044127	047105	047040	
1840	010022	020117	042522	047514	
1841	010030	040503	044524	047117	
1842	010036	000			
1843	010037	124	047517	020113	EM64: .ASCII ?TOOK TWO VECTORS WITH TRAP AND ABORT ON SAME INSTRUCTION?<CRLF>
1844	010044	053524	020117	042526	
1845	010052	052103	051117	020123	
1846	010060	044527	044124	052040	
1847	010066	040522	020120	047101	
1848	010074	020104	041101	051117	
1849	010102	020124	047117	051440	
1850	010110	046501	020105	047111	

1851	010116	052123	052522	052103	
1852	010124	047511	100116		
1853	010130	054105	042520	052103	.ASCIZ ?EXPECTING '1074' FOR KERNEL STACK POINTER?
1854	010136	047111	020107	030442	
1855	010144	033460	021064	043040	
1856	010152	051117	045440	051105	
1857	010160	042516	020114	052123	
1858	010166	041501	020113	047520	
1859	010174	047111	042524	000122	
1860	010202	042515	047515	054522	EM65: .ASCIZ ?MEMORY MANAGEMENT ABORT CONDITION WRONG?
1861	010210	046440	047101	043501	
1862	010216	046505	047105	020124	
1863	010224	041101	051117	020124	
1864	010232	047503	042116	052111	
1865	010240	047511	020116	051127	
1866	010246	047117	000107		
1867	010252	044502	020124	031061	EM66: .ASCIZ ?BIT 12 OF MMRO WAS NOT SET WHEN A.C.F. SATISFIED?
1868	010260	047440	020106	046515	
1869	010266	030122	053440	051501	
1870	010274	047040	052117	051440	
1871	010302	052105	053440	042510	
1872	010310	020116	027101	027103	
1873	010316	027106	051440	052101	
1874	010324	051511	044506	042105	
1875	010332	000			
1876	010333	116	020117	051124	EM67: .ASCIZ ?NO TRAP WHEN INSTRUCTION CLEARING BIT09 (MMRO) CAUSES TRAP COND.?
1877	010340	050101	053440	042510	
1878	010346	020116	047111	052123	
1879	010354	052522	052103	047511	
1880	010362	020116	046103	040505	
1881	010370	044522	043516	041040	
1882	010376	052111	034460	024040	
1883	010404	046515	030122	020051	
1884	010412	040503	051525	051505	
1885	010420	052040	040522	020120	
1886	010426	047503	042116	000056	
1887	010434	051105	047522	020122	EM70: .ASCII ?ERROR DURING M.M. ABORT IN TRAP SEQUENCE?<CRLF>
1888	010442	052504	044522	043516	
1889	010450	046440	046456	020056	
1890	010456	041101	051117	020124	
1891	010464	047111	052040	040522	
1892	010472	020120	042523	052521	
1893	010500	047105	042503	200	
1894	010505	105	050130	041505	.ASCIZ ?EXPECTED:?
1895	010512	042524	035104	000	
1896	010517	111	051516	051124	EM71: .ASCIZ ?INSTRUCTION FETCH DID NOT ABORT IN ILLEGAL MODE (10)?
1897	010524	041525	044524	047117	
1898	010532	043040	052105	044103	
1899	010540	042040	042111	047040	
1900	010546	052117	040440	047502	
1901	010554	052122	044440	020116	
1902	010562	046111	042514	040507	
1903	010570	020114	047515	042504	
1904	010576	024040	030061	000051	
1905	010604	051105	047522	020122	EM72: .ASCIZ ?ERROR CONDITION NOT CORRECT IN ILLEGAL MODE (10)?
1906	010612	047503	042116	052111	

Line	Address	Code	Module	Module	Message
1907	010620	047511	020116	047516	
1908	010626	020124	047503	051122	
1909	010634	041505	020124	047111	
1910	010642	044440	046114	043505	
1911	010650	046101	046440	042117	
1912	010656	020105	030450	024460	
1913	010664	000			
1914	010665	101	020124	042514	EM73: .ASCIZ ?AT LEAST ONE M.M. STATUS REGISTERS WAS CLOKED AFTER BEING LOCKED?
1915	010672	051501	020124	047117	
1916	010700	020105	027115	027115	
1917	010706	051440	040524	052524	
1918	010714	020123	042522	044507	
1919	010722	052123	051105	020123	
1920	010730	040527	020123	046103	
1921	010736	041517	042513	020104	
1922	010744	043101	042524	020122	
1923	010752	042502	047111	020107	
1924	010760	047514	045503	042105	
1925	010766	000			
1926	010767	104	042111	047040	EM74: .ASCIZ ?DID NOT CHANGE MAPPING TO SUPERVISOR MODE?
1927	010774	052117	041440	040510	
1928	011002	043516	020105	040515	
1929	011010	050120	047111	020107	
1930	011016	047524	051440	050125	
1931	011024	051105	044526	047523	
1932	011032	020122	047515	042504	
1933	011040	000			
1934	011041	101	047502	052122	EM75: .ASCIZ ?ABORT CONDITION INCORRECT EXPECTING 100051?
1935	011046	041440	047117	044504	
1936	011054	044524	047117	044440	
1937	011062	041516	051117	042522	
1938	011070	052103	042440	050130	
1939	011076	041505	044524	043516	
1940	011104	030440	030060	032460	
1941	011112	000061			
1942	011114	044504	020104	047516	EM76: .ASCIZ ?DID NOT CHANGE MAPPING TO USER MODE?
1943	011122	020124	044103	047101	
1944	011130	042507	046440	050101	
1945	011136	044520	043516	052040	
1946	011144	020117	051525	051105	
1947	011152	046440	042117	000105	
1948	011160	041101	051117	020124	EM77: .ASCIZ ?ABORT CONDITION INCORRECT EXPECTING 100153?
1949	011166	047503	042116	052111	
1950	011174	047511	020116	047111	
1951	011202	047503	051122	041505	
1952	011210	020124	054105	042520	
1953	011216	052103	047111	020107	
1954	011224	030061	030460	031465	
1955	011232	000			
1956	011233	115	051115	020062	EM100: .ASCIZ ?MMR2 LOCKED UP THE WRONG VIRTUAL ADDRESS?
1957	011240	047514	045503	042105	
1958	011246	052440	020120	044124	
1959	011254	020105	051127	047117	
1960	011262	020107	044526	052122	
1961	011270	040525	020114	042101	
1962	011276	051104	051505	000123	

1963	011304	046515	030122	046040	EM101: .ASCIZ ?MMRO LOCKED UP THE WRONG PAGE NUMBER?
1964	011312	041517	042513	020104	
1965	011320	050125	052040	042510	
1966	011326	053440	047522	043516	
1967	011334	050040	043501	020105	
1968	011342	052516	041115	051105	
1969	011350	000			
1970	011351	127	041055	052111	EM102: .ASCIZ ?W-BIT NOT SET ON WRITE TO PAGE 4?
1971	011356	047040	052117	051440	
1972	011364	052105	047440	020116	
1973	011372	051127	052111	020105	
1974	011400	047524	050040	043501	
1975	011406	020105	000064		
1976	011412	026527	044502	020124	EM103: .ASCIZ ?W-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4?
1977	011420	044504	020104	047516	
1978	011426	020124	042522	040515	
1979	011434	047111	051440	052105	
1980	011442	047440	020116	027115	
1981	011450	027115	040440	047502	
1982	011456	052122	040440	020124	
1983	011464	040520	042507	032040	
1984	011472	000			
1985	011473	127	041055	052111	EM104: .ASCIZ ?W-BIT DID NOT CLEAR ON WRITE TO KIPDR4?
1986	011500	042040	042111	047040	
1987	011506	052117	041440	042514	
1988	011514	051101	047440	020116	
1989	011522	051127	052111	020105	
1990	011530	047524	045440	050111	
1991	011536	051104	000064		
1992	011542	026501	044502	020124	EM105: .ASCIZ ?A-BIT DID NOT SET ON READ TO PAGE 4 A.C.F.=4?
1993	011550	044504	020104	047516	
1994	011556	020124	042523	020124	
1995	011564	047117	051040	040505	
1996	011572	020104	047524	050040	
1997	011600	043501	020105	020064	
1998	011606	027101	027103	027106	
1999	011614	032075	000		
2000	011617	101	041055	052111	EM106: .ASCIZ ?A-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4?
2001	011624	042040	042111	047040	
2002	011632	052117	051040	046505	
2003	011640	044501	020116	042523	
2004	011646	020124	047117	046440	
2005	011654	046456	020056	041101	
2006	011662	051117	020124	052101	
2007	011670	050040	043501	020105	
2008	011676	000064			
2009	011700	026501	044502	020124	EM107: .ASCIZ ?A-BIT DID NOT CLEAR ON WRITE TO KIPAR4?
2010	011706	044504	020104	047516	
2011	011714	020124	046103	040505	
2012	011722	020122	047117	053440	
2013	011730	044522	042524	052040	
2014	011736	020117	044513	040520	
2015	011744	032122	000		
2016	011747	127	041055	052111	EM110: .ASCIZ ?W-BIT DID NOT SET ON WRITE TO I/O PAGE?
2017	011754	042040	042111	047040	
2018	011762	052117	051440	052105	

2019	011770	047440	020116	051127	
2020	011776	052111	020105	047524	
2021	012004	044440	047457	050040	
2022	012012	043501	000105		
2023	012016	026527	044502	020124	EM111: .ASCIZ ?W-BIT DID NOT REMAIN SET AFTER INTERNAL REGISTER WRITE?
2024	012024	044504	020104	047516	
2025	012032	020124	042522	040515	
2026	012040	047111	051440	052105	
2027	012046	040440	052106	051105	
2028	012054	044440	052116	051105	
2029	012062	040516	020114	042522	
2030	012070	044507	052123	051105	
2031	012076	053440	044522	042524	
2032	012104	000			
2033	012105	104	040525	020114	EM112: .ASCIZ ?DUAL MAPPING BETWEEN PAGES?
2034	012112	040515	050120	047111	
2035	012120	020107	042502	053524	
2036	012126	042505	020116	040520	
2037	012134	042507	000123		
2038	012140	047516	050040	043501	EM113: .ASCIZ ?NO PAGE HAD BOTH ITS A & W BITS SET?
2039	012146	020105	040510	020104	
2040	012154	047502	044124	044440	
2041	012162	051524	040440	023040	
2042	012170	053440	041040	052111	
2043	012176	020123	042523	000124	
2044	012204	044504	020104	047516	EM114: .ASCIZ ?DID NOT PICK UP CORRECT STACK POINTER?
2045	012212	020124	044520	045503	
2046	012220	052440	020120	047503	
2047	012226	051122	041505	020124	
2048	012234	052123	041501	020113	
2049	012242	047520	047111	042524	
2050	012250	000122			
2051	012252	052503	051122	047105	EM115: .ASCIZ ?CURRENT MODE STACK NOT PUSHED IN MFP INSTRUCTION?
2052	012260	020124	047515	042504	
2053	012266	051440	040524	045503	
2054	012274	047040	052117	050040	
2055	012302	051525	042510	020104	
2056	012310	047111	046440	050106	
2057	012316	044440	051516	051124	
2058	012324	041525	044524	047117	
2059	012332	000			
2060	012333	127	047522	043516	EM116: .ASCIZ ?WRONG DATA FETCHED BY MFP INSTRUCTION?
2061	012340	042040	052101	020101	
2062	012346	042506	041524	042510	
2063	012354	020104	054502	046440	
2064	012362	050106	044440	051516	
2065	012370	051124	041525	044524	
2066	012376	047117	000		
2067	012401	124	044522	042105	EM117: .ASCIZ ?TRIED TO REFERENCE NON-RESIDENT PAGE?
2068	012406	052040	020117	042522	
2069	012414	042506	042522	041516	
2070	012422	020105	047516	026516	
2071	012430	042522	044523	042504	
2072	012436	052116	050040	043501	
2073	012444	000105			
2074	012446	052123	041501	020113	EM120: .ASCIZ ?STACK POINTER NOT CHANGED BY MFP INSTRUCTION?

2075	012454	047520	047111	042524	
2076	012462	020122	047516	020124	
2077	012470	044103	047101	042507	
2078	012476	020104	054502	046440	
2079	012504	050124	044440	051516	
2080	012512	051124	041525	044524	
2081	012520	047117	000		
2082	012523	111	041516	051117	EM121: .ASCIZ ?INCORRECT STORE BY MTP INSTRUCTION?
2083	012530	042522	052103	051440	
2084	012536	047524	042522	041040	
2085	012544	020131	052115	020120	
2086	012552	047111	052123	052522	
2087	012560	052103	047511	000116	
2088	012566	041101	051117	042524	EM122: .ASCIZ ?ABORTED THRU RIGHT VECTOR BUT PICKED UP WRONG PSW?
2089	012574	020104	044124	052522	
2090	012602	051040	043511	052110	
2091	012610	053040	041505	047524	
2092	012616	020122	052502	020124	
2093	012624	044520	045503	042105	
2094	012632	052440	020120	051127	
2095	012640	047117	020107	051520	
2096	012646	000127			
2097	012650	041101	051117	042524	EM123: .ASCIZ ?ABORTED THRU SUPERVISOR SPACE, SUPERVISOR PSW IS XXX340?
2098	012656	020104	044124	052522	
2099	012664	051440	050125	051105	
2100	012672	044526	047523	020122	
2101	012700	050123	041501	026105	
2102	012706	051440	050125	051105	
2103	012714	044526	047523	020122	
2104	012722	051520	020127	051511	
2105	012730	054040	054130	032063	
2106	012736	000060			
2107	012740	041101	051117	042524	EM124: .ASCIZ ?ABORTED THRU USER SPACE, USER PSW IS XXX000?
2108	012746	020104	044124	052522	
2109	012754	052440	042523	020122	
2110	012762	050123	041501	026105	
2111	012770	052440	042523	020122	
2112	012776	051520	020127	051511	
2113	013004	054040	054130	030060	
2114	013012	000060			
2115	013014	031062	041055	052111	EM125: .ASCIZ ?22-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM?
2116	013022	046440	050101	044520	
2117	013030	043516	050040	051517	
2118	013036	044523	046102	020105	
2119	013044	047510	042514	044440	
2120	013052	020116	040515	047111	
2121	013060	046440	046505	051117	
2122	013066	020131	051106	046517	
2123	013074	000			
2124	013075	062	026462	044502	EM126: .ASCIZ ?22-BIT MAPPING POSSIBLE HOLE AT THE TOP OF MEMORY?
2125	013102	020124	040515	050120	
2126	013110	047111	020107	047520	
2127	013116	051523	041111	042514	
2128	013124	044040	046117	020105	
2129	013132	052101	052040	042510	
2130	013140	052040	050117	047440	

Line	Code	Address	Value	Message
2131	013146	020106	042515	047515
2132	013154	054522	000	
2133	013157	104	042111	047040
2134	013164	052117	040440	047502
2135	013172	052122	051040	043105
2136	013200	051105	047105	042503
2137	013206	052040	020117	020101
2138	013214	042515	047515	054522
2139	013222	046440	046501	043501
2140	013230	046505	047105	020124
2141	013236	042522	044507	052123
2142	013244	051105	000	
2143	013247	101	047502	052122
2144	013254	044440	020116	042513
2145	013262	047122	046101	042040
2146	013270	051455	040520	042503
2147	013276	050040	041511	042513
2148	013304	020104	050125	053040
2149	013312	041505	047524	020122
2150	013320	051106	046517	044440
2151	013326	051455	040520	042503
2152	013334	000		
2153				
2154	013335	115	046456	020056
2155	013342	041101	051117	020124
2156	013350	047111	045440	051105
2157	013356	042516	020114	026504
2158	013364	050123	041501	020105
2159	013372	040510	020104	051127
2160	013400	047117	020107	047503
2161	013406	042116	052111	047511
2162	013414	000116		
2163	013416	026504	050123	041501
2164	013424	020105	047105	041101
2165	013432	042514	041440	051111
2166	013440	052503	052111	054522
2167	013446	044040	051501	043040
2168	013454	044501	042514	000104
2169	013462	054502	020120	044502
2170	013470	020124	047111	045440
2171	013476	050111	051104	041440
2172	013504	052517	042114	047040
2173	013512	052117	041040	020105
2174	013520	046103	040505	042522
2175	013526	000104		
2176	013530	054502	020120	044502
2177	013536	020124	047111	045440
2178	013544	050111	051104	041440
2179	013552	052517	042114	047040
2180	013560	052117	041040	020105
2181	013566	042523	000124	
2182	013572	042524	052123	042055
2183	013600	052101	020101	047503
2184	013606	046125	020104	047516
2185	013614	020124	042502	046440
2186	013622	042101	020105	044510

EM127: .ASCIZ ?DID NOT ABORT REFERENCE TO A MEMORY MANAGEMENT REGISTER?

EM130: .ASCIZ ?ABORT IN KERNAL D-SPACE PICKED UP VECTOR FROM I-SPACE?

EM131: .ASCIZ ?M.M. ABORT IN KERNEL D-SPACE HAD WRONG CONDITION?

EM132: .ASCIZ ?D-SPACE ENABLE CIRCUITRY HAS FAILED?

EM133: .ASCIZ ?BYP BIT IN KIPDR COULD NOT BE CLEARED?

EM134: .ASCIZ ?BYP BIT IN KIPDR COULD NOT BE SET?

EM135: .ASCIZ /TEST-DATA COULD NOT BE MADE HIT/

2187	013650	000124			
2188	013632	042524	052123	042055	EM136: .ASCII /TEST-DATA REFERENCE NOT A MISS/
2189	013640	052101	020101	042522	
2190	013646	042506	042522	041516	
2191	013654	020105	047516	020124	
2192	013662	020101	044515	051523	
2193	013670	005015	040503	044103	.ASCIZ <15><12>/CACHED DATA WAS NOT FORCED A MISS ON VIRTUAL PAGE BYPASS/
2194	013676	042105	042040	052101	
2195	013704	020101	040527	020123	
2196	013712	047516	020124	047506	
2197	013720	041522	042105	040440	
2198	013726	046440	051511	020123	
2199	013734	047117	053040	051111	
2200	013742	052524	046101	050040	
2201	013750	043501	020105	054502	
2202	013756	040520	051523	000	
2203	013763	124	051505	020124	EM137: .ASCII /TEST DATA REFERENCE NOT A MISS/
2204	013770	040504	040524	051040	
2205	013776	043105	051105	047105	
2206	014004	042503	047040	052117	
2207	014012	040440	046440	051511	
2208	014020	123			
2209	014021	015	041412	041501	.ASCIZ <15><12>/CACHED DATA WAS NOT INVALIDATED ON VIRTUAL PAGE BYPASS/
2210	014026	042510	020104	040504	
2211	014034	040524	053440	051501	
2212	014042	047040	052117	044440	
2213	014050	053116	046101	042111	
2214	014056	052101	042105	047440	
2215	014064	020116	044526	052122	
2216	014072	040525	020114	040520	
2217	014100	042507	041040	050131	
2218	014106	051501	000123		
2219	014112	054502	020120	044502	EM140: .ASCIZ ?BYP BIT IN SIPDR COULD NOT BE CLEARED?
2220	014120	020124	047111	051440	
2221	014126	050111	051104	041440	
2222	014134	052517	042114	047040	
2223	014142	052117	041040	020105	
2224	014150	046103	040505	042522	
2225	014156	000104			
2226	014160	054502	020120	044502	EM141: .ASCIZ ?BYP BIT IN SIPDR COULD NOT BE SET?
2227	014166	020124	047111	051440	
2228	014174	050111	051104	041440	
2229	014202	052517	042114	047040	
2230	014210	052117	041040	020105	
2231	014216	042523	000124		
2232	014222	054502	020120	044502	EM142: .ASCIZ ?BYP BIT IN UIPDR COULD NOT BE CLEARED?
2233	014230	020124	047111	052440	
2234	014236	050111	051104	041440	
2235	014244	052517	042114	047040	
2236	014252	052117	041040	020105	
2237	014260	046103	040505	042522	
2238	014266	000104			
2239	014270	054502	020120	044502	EM143: .ASCIZ ?BYP BIT IN UIPDR COULD NOT BE SET?
2240	014276	020124	047111	052440	
2241	014304	050111	051104	041440	
2242	014312	052517	042114	047040	

2243	014320	052117	041040	020105	
2244	014326	042523	000124		
2245	014332	050104	051101	051040	EM145: .ASCIZ /DPAR READ BACK INCORRECTLY/
2246	014340	040505	020104	040502	
2247	014346	045503	044440	041516	
2248	014354	051117	042522	052103	
2249	014362	054514	000		
2250	014365	012	042015	040520	ECO: .ASCIZ <12><15>/DPAR READ BACK PROBLEM CORRECTED BY ECO NO. M8140-00002/<12><15>
2251	014372	020122	042522	042101	
2252	014400	041040	041501	020113	
2253	014406	051120	041117	042514	
2254	014414	020115	047503	051122	
2255	014422	041505	042524	020104	
2256	014430	054502	042440	047503	
2257	014436	047040	027117	046440	
2258	014444	030470	030064	030055	
2259	014452	030060	031060	006412	
2260	014460	000			
2261	014461	015	051412	051123	EM146: .ASCII <CR><LF>/SSRC KT ABORT FLG DOESN'T GET TO TMCC E34(10) OR/<CR><LF>
2262	014466	020103	052113	040440	
2263	014474	047502	052122	043040	
2264	014502	043514	042040	042517	
2265	014510	047123	052047	043440	
2266	014516	052105	052040	020117	
2267	014524	046524	041503	042440	
2268	014532	032063	030450	024460	
2269	014540	047440	006522	012	
2270	014545	105	032063	030450	.ASCIZ /E34(10) BAD OR TMCC AERF(1) DOESN'T GO HIGH ON A KT ABORT/
2271	014552	024460	041040	042101	
2272	014560	047440	020122	046524	
2273	014566	041503	040440	051105	
2274	014574	024106	024461	042040	
2275	014602	042517	047123	052047	
2276	014610	043440	020117	044510	
2277	014616	044107	047440	020116	
2278	014624	020101	052113	040440	
2279	014632	047502	052122	000	
2280	014637	015	050012	053523	EM147: .ASCIZ<CR><LF>/PSW BIT 8 FAILED TO SET/
2281	014644	041040	052111	034040	
2282	014652	043040	044501	042514	
2283	014660	020104	047524	051440	
2284	014666	052105	000		
2285	014671	015	052012	041515	EM150: .ASCII<CR><LF>/TMCB SEGT DOESN'T GO LOW ON A KT ABORT OR/<CR><LF>
2286	014676	020102	042523	052107	
2287	014704	042040	042517	047123	
2288	014712	052047	043440	020117	
2289	014720	047514	020127	047117	
2290	014726	040440	045440	020124	
2291	014734	041101	051117	020124	
2292	014742	051117	005015		
2293	014746	052111	042040	042517	.ASCIZ /IT DOESN'T GET TO DAPE/
2294	014754	047123	052047	043440	
2295	014762	052105	052040	020117	
2296	014770	040504	042520	000	
2297	014775	015	042012	050101	EM151: .ASCIZ<CR><LF>/DAPE TV05*07 DOESN'T GO HIGH ON SEGT/
2298	015002	020105	053124	032460	

2299	015010	030052	020067	047504	
2300	015016	051505	023516	020124	
2301	015024	047507	044040	043511	
2302	015032	020110	047117	051440	
2303	015040	043505	000124		
2304	015044	005015	040504	042520	EM152: .ASCIZ<CR><LF>/DAPE TV03 DOESN'T GO HIGH ON SEGT/
2305	015052	052040	030126	020063	
2306	015060	047504	051505	023516	
2307	015066	020124	047507	044040	
2308	015074	043511	020110	047117	
2309	015102	051440	043505	000124	
2310	015110	005015	046524	040503	EM153: .ASCII<CR><LF>/TMCA SEG+CON+PAR DOESN'T GO LOW OR IT DOES/<CR><LF>
2311	015116	051440	043505	041453	
2312	015124	047117	050053	051101	
2313	015132	042040	042517	047123	
2314	015140	052047	043440	020117	
2315	015146	047514	020127	051117	
2316	015154	044440	020124	047504	
2317	015162	051505	005015		
2318	015166	047516	020124	042507	.ASCIZ /NOT GET THRU TMCB E70(2)/
2319	015174	020124	044124	052522	
2320	015202	052040	041515	020102	
2321	015210	033505	024060	024462	
2322	015216	000			
2323	015217	015	052012	041515	EM155: .ASCII<CR><LF>/TMCA SEGTF DOESN'T GET TO E44 OR TMCE PAUSES H/<CR><LF>
2324	015224	020101	042523	052107	
2325	015232	020106	047504	051505	
2326	015240	023516	020124	042507	
2327	015246	020124	047524	042440	
2328	015254	032064	047440	020122	
2329	015262	046524	042503	050040	
2330	015270	052501	042523	020123	
2331	015276	006510	012		
2332	015301	104	042517	047123	.ASCIZ /DOESN'T GET TO E44 OR E44 BAD/
2333	015306	052047	043440	052105	
2334	015314	052040	020117	032105	
2335	015322	020064	051117	042440	
2336	015330	032064	041040	042101	
2337	015336	000			
2338	015337	015	052012	041515	EM156: .ASCII<CR><LF>/TMCC NEXM DOESN'T GO LOW OR IT DOESN'T GET THRU E34/<CR><LF>
2339	015344	020103	042516	046530	
2340	015352	042040	042517	047123	
2341	015360	052047	043440	020117	
2342	015366	047514	020127	051117	
2343	015374	044440	020124	047504	
2344	015402	051505	023516	020124	
2345	015410	042507	020124	044124	
2346	015416	052522	042440	032063	
2347	015424	005015			
2348	015426	051117	044440	020124	.ASCIZ /OR IT DOESN'T GET TO E14 OR E40 BAD/
2349	015434	047504	051505	023516	
2350	015442	020124	042507	020124	
2351	015450	047524	042440	032061	
2352	015456	047440	020122	032105	
2353	015464	020060	040502	000104	
2354	015472	005015	042516	046530	EM160: .ASCII<CR><LF>/NEXM BIT DIDN'T SET IN CPU ERROR REG/<CR><LF>

2355	015500	041040	052111	042040	
2356	015506	042111	023516	020124	
2357	015514	042523	020124	047111	
2358	015522	041440	052520	042440	
2359	015530	051122	051117	051040	
2360	015536	043505	005015		
2361	015542	051117	052040	041515	.ASCIZ /OR TMCC ABORT DOESN'T GO HIGH/
2362	015550	020103	041101	051117	
2363	015556	020124	047504	051505	
2364	015564	023516	020124	047507	
2365	015572	044040	043511	000110	
2366	015600	005015	042516	046530	EM161: .ASCIZ <CR><LF>/NEXM BIT DIDN'T CLEAR IN CPU ERROR REG/
2367	015606	041040	052111	042040	
2368	015614	042111	023516	020124	
2369	015622	046103	040505	020122	
2370	015630	047111	041440	052520	
2371	015636	042440	051122	051117	
2372	015644	051040	043505	000	
2373	015651	015	052012	041515	EM162: .ASCIZ<CR><LF>/TMCE KT BEND DOESN'T GO LOW ON TMCC NEXM LOW/
2374	015656	020105	052113	041040	
2375	015664	047105	020104	047504	
2376	015672	051505	023516	020124	
2377	015700	047507	046040	053517	
2378	015706	047440	020116	046524	
2379	015714	041503	047040	054105	
2380	015722	020115	047514	000127	
2381	015730	005015	046524	042503	EM163: .ASCIZ<CR><LF>/TMCE KT BEND DOESN'T GO LOW ON TMCD SL RED/
2382	015736	045440	020124	042502	
2383	015744	042116	042040	042517	
2384	015752	047123	052047	043440	
2385	015760	020117	047514	020127	
2386	015766	047117	052040	041515	
2387	015774	020104	046123	051040	
2388	016002	042105	000		
2389	016005	015	052012	041515	EM164: .ASCIZ<CR><LF>/TMCE KT BEND DOESN'T GO LOW ON TMCC ODD ADRS ERR/
2390	016012	020105	052113	041040	
2391	016020	047105	020104	047504	
2392	016026	051505	023516	020124	
2393	016034	047507	046040	053517	
2394	016042	047440	020116	046524	
2395	016050	041503	047440	042104	
2396	016056	040440	051104	020123	
2397	016064	051105	000122		
2398	016070	005015	052113	040440	EM165: .ASCIZ<CR><LF>?KT ABORT FAILED TO OVER-RIDE NEXM TRAP (KB11-E/EM?)
2399	016076	047502	052122	043040	
2400	016104	044501	042514	020104	
2401	016112	047524	047440	042526	
2402	016120	026522	044522	042504	
2403	016126	047040	054105	020115	
2404	016134	051124	050101	024040	
2405	016142	041113	030461	042455	
2406	016150	042457	000115		
2407	016154	005015	046524	042503	EM166: .ASCIZ<CR><LF>/TMCE CACHE BEND DIDN'T GO HIGH ON KT ABORT/
2408	016162	041440	041501	042510	
2409	016170	041040	047105	020104	
2410	016176	044504	047104	052047	

2411	016204	043440	020117	044510
2412	016212	044107	047440	020116
2413	016220	052113	040440	047502
2414	016226	052122	000	
2415				
2416	016231	015	043012	046117
2417	016236	047514	044527	043516
2418	016244	044440	020123	020101
2419	016252	044514	052123	047440
2420	016260	020106	044124	020105
2421	016266	052123	041501	020113
2422	016274	044514	044515	020124
2423	016302	042522	006507	012
2424	016307	046	051440	020120
2425	016314	040526	052514	051505
2426	016322	052040	040510	020124
2427	016330	040503	051525	042105
2428	016336	040440	020116	051105
2429	016344	047522	027122	020040
2430	016352	044124	054505	040440
2431	016360	042522	005015	
2432	016364	051107	052517	042520
2433	016372	020104	041501	047503
2434	016400	042122	047111	020107
2435	016406	047524	042440	051122
2436	016414	051117	052040	050131
2437	016422	051505	000	
2438				
2439	016425	015	051412	051123
2440	016432	020101	051520	051040
2441	016440	051505	047524	042522
2442	016446	030450	020051	047504
2443	016454	051505	023516	020124
2444	016462	042507	020124	047524
2445	016470	051040	041501	020113
2446	016476	033105	006463	012
2447	016503	117	020122	033105
2448	016510	024063	024465	041040
2449	016516	042101	000	
2450	016521	015	043412	044517
2451	016526	043516	052040	020117
2452	016534	042516	052130	052040
2453	016542	051505	000124	
2454	016546	044124	020105	047506
2455	016554	046114	053517	047111
2456	016562	020107	051101	020105
2457	016570	040520	027522	042120
2458	016576	020122	042522	042506
2459	016604	042522	041516	020105
2460	016612	044524	042515	052517
2461	016620	051524	000	
2462	016623	124	042510	043040
2463	016630	046117	047514	044527
2464	016636	043516	040440	042522
2465	016644	042040	040525	020114
2466	016652	042101	051104	051505

EM167: .ASCII<CR><LF>/FOLLOWING IS A LIST OF THE STACK LIMIT REG/<CR><LF>

.ASCII/B SP VALUES THAT CAUSED AN ERROR. THEY ARE/<CR><LF>

.ASCIZ /GROUPED ACCORDING TO ERROR TYPES/

EM170: .ASCII<CR><LF>/SSRA PS RESTORE(1) DOESN'T GET TO RACK E63/<CR><LF>

.ASCIZ /OR E63(5) BAD/

EM171: .ASCIZ<CR><LF> /GOING TO NEXT TEST/

EM201: .ASCIZ ?THE FOLLOWING ARE PAR/PDR REFERENCE TIMEOUTS?

EM202: .ASCIZ ?THE FOLLOWING ARE DUAL ADDRESSING ERRORS FOR PAR'S/PDR'S?

2467	016660	044523	043516	042440
2468	016666	051122	051117	020123
2469	016674	047506	020122	040520
2470	016702	023522	027523	042120
2471	016710	023522	000123	
2472	016714	044124	020105	047506
2473	016722	046114	053517	047111
2474	016730	020107	051101	020105
2475	016736	047503	047125	020124
2476	016744	040520	052124	051105
2477	016752	020116	051105	047522
2478	016760	051522	043040	051117
2479	016766	050040	051101	051447
2480	016774	050057	051104	051447
2481	017002	000		
2482				
2483				
2484	017003	105	050130	041505
2485	017010	042124	040	
2486	017013	122	041505	044505
2487	017020	042126	052040	051505
2488	017026	047124	020117	050040
2489	017034	020103	052101	040440
2490	017042	047502	052122	000
2491	017047	120	051101	052111
2492	017054	020131	040440	042104
2493	017062	042522	051523	020040
2494	017070	041440	047117	051124
2495	017076	046117	046440	044501
2496	017104	052116	047105	200
2497	017111	103	047117	044504
2498	017116	047124	051040	043105
2499	017124	051105	047105	042103
2500	017132	051040	043505	051511
2501	017140	051124	051040	043505
2502	017146	051511	051124	052040
2503	017154	051505	047124	020117
2504	017162	050040	020103	052101
2505	017170	040440	047502	052122
2506	017176	000		
2507	017177	105	051122	051117
2508	017204	020040	040440	052125
2509	017212	044517	042057	053040
2510	017220	051111	052524	046101
2511	017226	200		
2512	017227	122	043505	051511
2513	017234	051124	051040	043505
2514	017242	051511	051124	040440
2515	017250	042104	042522	051523
2516	017256	052040	051505	047124
2517	017264	020117	050040	020103
2518	017272	052101	040440	047502
2519	017300	052122	000	
2520	017303	105	050130	041505
2521	017310	042124	042440	051122
2522	017316	051117	020040	040440

EM203: .ASCIZ ?THE FOLLOWING ARE COUNT PATTERN ERRORS FOR PAR'S/PDR'S?

DH1: .ASCII ?EXPECTD ?

DH2: .ASCIZ ?RECEIVD TESTNO PC AT ABORT?

DH3: .ASCII ?PARITY ADDRESS CONTROL MAINTEN?<CRLF>

.ASCIZ ?CONDITN REFERENC D REGISTR REGISTR TESTNO PC AT ABORT?

DH5: .ASCII ?ERROR AUTOI/D VIRTUAL?<CRLF>

.ASCIZ ?REGISTR REGISTR ADDRESS TESTNO PC AT ABORT?

DH6: .ASCII ?EXPECTD ERROR AUTOI/D VIRTUAL?<CRLF>

2523	017324	052125	044517	042057					
2524	017332	053040	051111	052524					
2525	017340	046101	200						
2526	017343	103	047117	044504		.ASCIZ	?CONDITN	REGISTR	REGISTR ADDRESS TESTNO PC AT ABORT?
2527	017350	047124	051040	043505					
2528	017356	051511	051124	051040					
2529	017364	043505	051511	051124					
2530	017372	040440	042104	042522					
2531	017400	051523	052040	051505					
2532	017406	047124	020117	050040					
2533	017414	020103	052101	040440					
2534	017422	047502	052122	000					
2535	017427	050	046515	030122	DH7:	.ASCIZ	?(MMR0)	TESTNO	ERRORPC?
2536	017434	020051	052040	051505					
2537	017442	047124	020117	042440					
2538	017450	051122	051117	041520					
2539	017456	000							
2540	017457	114	040517	042504	DH11:	.ASCIZ	?LOADED	RECEIVD	TESTNO ERRORPC?
2541	017464	020104	051040	041505					
2542	017472	044505	042126	052040					
2543	017500	051505	047124	020117					
2544	017506	042440	051122	051117					
2545	017514	041520	000						
2546	017517	105	050130	041505	DH13:	.ASCII	?EXPECTD	?	
2547	017524	042124	040						
2548	017527	050	046515	030522	DH12:	.ASCIZ	?(MMR1)	TESTNO	ERRORPC?
2549	017534	020051	052040	051505					
2550	017542	047124	020117	042440					
2551	017550	051122	051117	041520					
2552	017556	000							
2553	017557	105	050130	041505	DH14:	.ASCIZ	?EXPECTD	(MMR2)	TESTNO ERRORPC?
2554	017564	042124	024040	046515					
2555	017572	031122	020051	052040					
2556	017600	051505	047124	020117					
2557	017606	042440	051122	051117					
2558	017614	041520	000						
2559	017617	114	040517	042504	DH16:	.ASCII	?LOADED	?	
2560	017624	020104	040						
2561	017627	050	046515	031522	DH15:	.ASCIZ	?(MMR3)	TESTNO	ERRORPC?
2562	017634	020051	052040	051505					
2563	017642	047124	020117	042440					
2564	017650	051122	051117	041520					
2565	017656	000							
2566	017657	101	042104	047522	DH17:	.ASCIZ	?ADDROR	ADDRAND	TESTNO #ERRORS?
2567	017664	020122	040440	042104					
2568	017672	040522	042116	052040					
2569	017700	051505	047124	020117					
2570	017706	021440	051105	047522					
2571	017714	051522	000						
2572	017717	101	042104	042522	DH20:	.ASCIZ	?ADDRESS	DATA	TESTNO ERRORPC?
2573	017724	051523	042040	052101					
2574	017732	020101	020040	052040					
2575	017740	051505	047124	020117					
2576	017746	042440	051122	051117					
2577	017754	041520	000						
2578	017757	101	042104	047522	DH21:	.ASCII	?ADDROR	ADDRAND	ADDROR ADDRAND?<CRLF>

2579	017764	020122	040440	042104						
2580	017772	040522	042116	040440						
2581	020000	042104	047522	020122						
2582	020006	040440	042104	040522						
2583	020014	042116	200							
2584	020017	114	040517	042504	.ASCIZ	?LOADED	LOADED	ENABLED	ENABLED	TESTNO #ERRORS?
2585	020024	020104	046040	040517						
2586	020032	042504	020104	042440						
2587	020040	040516	046102	042105						
2588	020046	042440	040516	046102						
2589	020054	042105	052040	051505						
2590	020062	047124	020117	021440						
2591	020070	051105	047522	051522						
2592	020076	000								
2593	020077	101	042104	047522	DH22:	.ASCIZ	?ADDROR	ADDRAND	PATRNOR	PATRNAD DATAOR DATAND TESTNO #ERRORS?
2594	020104	020122	040440	042104						
2595	020112	040522	042116	050040						
2596	020120	052101	047122	051117						
2597	020126	050040	052101	047122						
2598	020134	042101	042040	052101						
2599	020142	047501	020122	042040						
2600	020150	052101	047101	020104						
2601	020156	042524	052123	047516						
2602	020164	020040	042443	051122						
2603	020172	051117	000123							
2604	020176	042101	051104	051505	DH23:	.ASCIZ	?ADDRESS	EXPECTD	RECEIVD	TESTNO ERRORPC?
2605	020204	020123	054105	042520						
2606	020212	052103	020104	042522						
2607	020220	042503	053111	020104						
2608	020226	042524	052123	047516						
2609	020234	020040	051105	047522						
2610	020242	050122	000103							
2611	020246	042522	040507	042104	DH24:	.ASCIZ	?REGADDR	TESTNO	ERRORPC?	
2612	020254	020122	042524	052123						
2613	020262	047516	020040	051105						
2614	020270	047522	050122	000103						
2615	020276	042522	040507	042104	DH25:	.ASCIZ	?REGADDR	DATA	TESTNO	ERRORPC?
2616	020304	020122	040504	040524						
2617	020312	020040	020040	042524						
2618	020320	052123	047516	020040						
2619	020326	051105	047522	050122						
2620	020334	000103								
2621	020336	042522	040507	042104	DH31:	.ASCIZ	?REGADDR	DATAREC	DATAEXP	TESTNO ERRORPC?
2622	020344	020122	040504	040524						
2623	020352	042522	020103	040504						
2624	020360	040524	054105	020120						
2625	020366	042524	052123	047516						
2626	020374	020040	051105	047522						
2627	020402	050122	000103							
2628	020406	051513	020120	020040	DH35:	.ASCIZ	?KSP	TESTNO	ERRORPC?	
2629	020414	020040	042524	052123						
2630	020422	047516	020040	051105						
2631	020430	047522	050122	000103						
2632	020436	047111	042504	020130	DH36:	.ASCII	?INDEX	INDEX	PAR/PDR?	<CRLF>
2633	020444	020040	047111	042504						
2634	020452	020130	020040	040520						

2635	020460	027522	042120	100122					
2636	020466	054105	042520	052103		.ASCIZ	?EXPECTD	RECEIVD	ADRREAD TESTNO ERRORPC?
2637	020474	020104	042522	042503					
2638	020502	053111	020104	042101					
2639	020510	051122	040505	020104					
2640	020516	042524	052123	047516					
2641	020524	020040	051105	047522					
2642	020532	050122	000103						
2643	020536	042101	051104	051505	DH37:	.ASCIZ	?ADDRESS	GDDATA	BADDATA TESTNO ERRORPC?
2644	020544	020123	042107	040504					
2645	020552	040524	020040	040502					
2646	020560	042104	052101	020101					
2647	020566	042524	052123	047516					
2648	020574	020040	051105	047522					
2649	020602	050122	000103						
2650	020606	052123	051101	040524	DH40:	.ASCIZ	?STARTADR	FINISHADR	TESTNO ERRORPC?
2651	020614	051104	020040	044506					
2652	020622	044516	044123	042101					
2653	020630	020122	042524	052123					
2654	020636	047516	020040	051105					
2655	020644	047522	050122	000103					
2656	020652	052123	051101	040524	DH41:	.ASCIZ	?STARTADR	TESTNO	ERRORPC?
2657	020660	051104	020040	042524					
2658	020666	052123	047516	020040					
2659	020674	051105	047522	050122					
2660	020702	000103							
2661	020704	040520	052124	051105	DH42:	.ASCII	?PATTERN	DATA	ADDRESS?<CRLF>
2662	020712	020116	040504	040524					
2663	020720	020040	020040	042101					
2664	020726	051104	051505	100123					
2665	020734	047514	042101	042105		.ASCIZ	?LOADED	FETCHED	INTENDED TESTNO ERRORPC?
2666	020742	020040	042506	041524					
2667	020750	042510	020104	047111					
2668	020756	042524	042116	042105					
2669	020764	020040	042524	052123					
2670	020772	047516	020040	051105					
2671	021000	047522	050122	000103					
2672	021006	042524	052123	047516	DH43:	.ASCIZ	?TESTNO	ERRORPC?	
2673	021014	020040	051105	047522					
2674	021022	050122	000103						
2675	021026	040504	040524	020040	DH44:	.ASCIZ	?DATA	TESTNO	ERRORPC?
2676	021034	020040	042524	052123					
2677	021042	047516	020040	051105					
2678	021050	047522	050122	000103					
2679	021056	047516	026516	054105	DH45:	.ASCIZ	?NON-EXADDR	TESTNO	ERRORPC?
2680	021064	042101	051104	052040					
2681	021072	051505	047124	020117					
2682	021100	042440	051122	051117					
2683	021106	041520	000						
2684	021111	113	050111	051101	DH50:	.ASCIZ	?KIPAR4	SIZELO	TESTNO ERRORPC?
2685	021116	020064	020040	051440					
2686	021124	055111	046105	020117					
2687	021132	020040	052040	051505					
2688	021140	047124	020117	042440					
2689	021146	051122	051117	041520					
2690	021154	000							

2691	021155	120	046107	047105	DH51:	.ASCIZ	?PGLENFD	VABLKNO	TESTNO	ERRORPC?
2692	021162	042106	053040	041101						
2693	021170	045514	047516	052040						
2694	021176	051505	047124	020117						
2695	021204	042440	051122	051117						
2696	021212	041520	000							
2697	021215	124	051505	047124	DH53:	.ASCIZ	?TESTNO	ERRORPC?		
2698	021222	020117	042440	051122						
2699	021230	051117	041520	000						
2700	021235	105	050130	040504	DH55:	.ASCIZ	?EXPDATA	RECDATA	TESTNO	ERRORPC?
2701	021242	040524	051040	041505						
2702	021250	040504	040524	052040						
2703	021256	051505	047124	020117						
2704	021264	042440	051122	051117						
2705	021272	041520	000							
2706	021275	050	046515	030122	DH56:	.ASCIZ	?(MMRO)	KIPDR4	TESTNO	ERRORPC?
2707	021302	020051	045440	050111						
2708	021310	051104	020064	052040						
2709	021316	051505	047124	020117						
2710	021324	042440	051122	051117						
2711	021332	041520	000							
2712	021335	122	041505	044505	DH64:	.ASCIZ	?RECEIVD	TESTNO	ERRORPC?	
2713	021342	042126	052040	051505						
2714	021350	047124	020117	042440						
2715	021356	051122	051117	041520						
2716	021364	000								
2717	021365	105	050130	047503	DH65:	.ASCIZ	?EXPCOND	ABRTCND	TESTNO	ERRORPC?
2718	021372	042116	040440	051102						
2719	021400	041524	042116	052040						
2720	021406	051505	047124	020117						
2721	021414	042440	051122	051117						
2722	021422	041520	000							
2723	021425	050	046515	030122	DH66:	.ASCIZ	?(MMRO)	TESTNO	ERRORPC?	
2724	021432	020051	052040	051505						
2725	021440	047124	020117	042440						
2726	021446	051122	051117	041520						
2727	021454	000								
2728	021455	130	054130	030460	DH70:	.ASCII	?XXX017	040241	173366	000004?<CRLF>
2729	021462	020067	030040	030064						
2730	021470	032062	020061	030440						
2731	021476	031467	033063	020066						
2732	021504	030040	030060	030060						
2733	021512	100064								
2734	021514	051120	051517	040524		.ASCII	?PROSTAT	(MMRO)	(MMR1)	(MMR2) TESTNO ERRORPC?<CRLF>
2735	021522	020124	046450	051115						
2736	021530	024460	020040	046450						
2737	021536	051115	024461	020040						
2738	021544	046450	051115	024462						
2739	021552	020040	042524	052123						
2740	021560	047516	020040	051105						
2741	021566	047522	050122	100103						
2742	021574	042522	042503	053111		.ASCIZ	?RECEIVED:?			
2743	021502	042105	000072							
2744	021606	054105	051520	040524	DH72:	.ASCIZ	?EXPSTAT	(MMRO)	TESTNO	ERRORPC?
2745	021614	020124	046450	051115						
2746	021622	024460	020040	042524						

2803	022302	042524	052116	052040						
2804	022310	051505	047124	020117						
2805	022316	042440	051122	051117						
2806	022324	041520	000							
2807	022327	105	050130	041505	DH114:	.ASCIZ	?EXPECTD RECEIVD TESTNO	ERRORPC?		
2808	022334	042124	051040	041505						
2809	022342	044505	042126	052040						
2810	022350	051505	047124	020117						
2811	022356	042440	051122	051117						
2812	022364	041520	000							
2813	022367	105	050130	041505	DH116:	.ASCIZ	?EXPECTD RECEIVD TESTNO	ERRORPC?		
2814	022374	042124	051040	041505						
2815	022402	044505	042126	052040						
2816	022410	051505	047124	020117						
2817	022416	042440	051122	051117						
2818	022424	041520	000							
2819	022427	050	046515	030122	DH117:	.ASCIZ	?(MMR0) (MMR1) (MMR2)	TESTNO	ERRORPC?	
2820	022434	020051	024040	046515						
2821	022442	030522	020051	024040						
2822	022450	046515	031122	020051						
2823	022456	052040	051505	047124						
2824	022464	020117	042440	051122						
2825	022472	051117	041520	000						
2826	022477	123	045524	052120	DH120:	.ASCIZ	?STKPTR	TESTNO	ERRORPC?	
2827	022504	020122	052040	051505						
2828	022512	047124	020117	042440						
2829	022520	051122	051117	041520						
2830	022526	000								
2831	022527	107	042104	052101	DH121:	.ASCIZ	?GDDATA	STORED	TESTNO	ERRORPC?
2832	022534	020101	051440	047524						
2833	022542	042522	020104	052040						
2834	022550	051505	047124	020117						
2835	022556	042440	051122	051117						
2836	022564	041520	000							
2837	022567	050	051520	024527	DH122:	.ASCIZ	?(PSW)	TESTNO	ERRORPC	EXPECTING XXX340?
2838	022574	020040	052040	051505						
2839	022602	047124	020117	042440						
2840	022610	051122	051117	041520						
2841	022616	042440	050130	041505						
2842	022624	044524	043516	054040						
2843	022632	054130	032063	000060						
2844	022640	050050	053523	020051	DH123:	.ASCIZ	?(PSW)	TESTNO	ERRORPC?	
2845	022646	020040	042524	052123						
2846	022654	047516	020040	051105						
2847	022662	047522	050122	000103						
2848	022670	042524	052123	047516	DH127:	.ASCIZ	?TESTNO	ERRORPC?		
2849	022676	042440	051122	051117						
2850	022704	041520	000							
2851	022707	050	046515	030122	DH131:	.ASCIZ	?(MMR0) (MMR1) (MMR2)	TESTNO	ERRORPC	EXPECTING 020031?
2852	022714	020051	024040	046515						
2853	022722	030522	020051	024040						
2854	022730	046515	031122	020051						
2855	022736	052040	051505	047124						
2856	022744	020117	042440	051122						
2857	022752	051117	041520	042440						
2858	022760	050130	041505	044524						

2859	022766	043516	030040	030062					
2860	022774	031460	000061						
2861	023000	020040	041520	020040	DH133:	.ASCIZ	? PC	KIPDR	(KIPDR)?
2862	023006	020040	044513	042120					
2863	023014	020122	024040	044513					
2864	023022	042120	024522	000					
2865	023027	040	050040	020103	DH135:	.ASCIZ	/ PC	CCR PAR-ADR	(PAR) (PDR) TST-DATA-ADRS(VA)/
2866	023034	020040	041440	051103					
2867	023042	020040	040520	026522					
2868	023050	042101	020122	020040					
2869	023056	050050	051101	020051					
2870	023064	020040	042120	024522					
2871	023072	020040	051524	026524					
2872	023100	040504	040524	040455					
2873	023106	051104	024123	040526					
2874	023114	000051							
2875	023116	020040	041520	020040	DH140:	.ASCIZ	? PC	SIPDR	(SIPDR)?
2876	023124	020040	044523	042120					
2877	023132	020122	024040	044523					
2878	023140	042120	024522	000					
2879	023145	040	050040	020103	DH142:	.ASCIZ	? PC	UIPDR	(UIPDR)?
2880	023152	020040	052440	050111					
2881	023160	051104	020040	052450					
2882	023166	050111	051104	000051					
2883	023174	042101	051104	051505	DH145:	.ASCIZ	/ADDRESS EXPECT RECEIVE TESTNO	ERRORPC/	
2884	023202	020123	054105	042520					
2885	023210	052103	051040	041505					
2886	023216	044505	042526	052040					
2887	023224	051505	047124	020117					
2888	023232	042440	051122	051117					
2889	023240	041520	000						
2890	023243	105	051122	051117	DH146:	.ASCIZ	/ERRORPC TEST NUMBER/		
2891	023250	041520	052040	051505					
2892	023256	020124	052516	041115					
2893	023264	051105	000						
2894	023267	105	051122	051117	DH147:	.ASCII	/ERRORPC	CPUERR REG	TST NUM/<CR><LF>
2895	023274	041520	041411	052520					
2896	023302	051105	020122	042522					
2897	023310	004507	051524	020124					
2898	023316	052516	006515	012					
2899	023323	011	054105	042520		.ASCIZ	/	EXPECT ACTUAL/	
2900	023330	052103	040411	052103					
2901	023336	040525	000114						
2902	023342	051105	047522	050122	DH150:	.ASCIZ	/ERRORPC TEST NUMBER/<CR><LF>		
2903	023350	020103	042524	052123					
2904	023356	047040	046525	042502					
2905	023364	006522	000012						
2906	023370	042101	051104	051505	DH201:	.ASCIZ	?ADDRESS TESTNO?		
2907	023376	020123	042524	052123					
2908	023404	047516	000						
2909	023407	101	042104	042522	DH202:	.ASCII	?ADDRESS ADDRESS?<CRLF>		
2910	023414	051523	040440	042104					
2911	023422	042522	051523	200					
2912	023427	114	040517	042504		.ASCIZ	?LOADED JUSTREAD TESTNO?		
2913	023434	020104	045040	051525					
2914	023442	051124	040505	020104					

2915	023450	042524	052123	047516					
2916	023456	000							
2917	023457	101	042104	042522	DH203:	.ASCIZ	?ADDRESS DATARED PATTERN COUNT	TESTNO?	
2918	023464	051523	042040	052101					
2919	023472	051101	042105	050040					
2920	023500	052101	042524	047122					
2921	023506	041440	052517	052116					
2922	023514	020040	052040	051505					
2923	023522	047124	000117						
2924	023526	020040	000		TWOSP:	.ASCIZ	/ /		
2925		023532				.EVEN			
2926									
2927									
2928	023532	001224			DT1:	.WORD	CPUEXP		
2929	023534	001260	001264	001262	DT2:	.WORD	PCPUER,TESTNO,BADPC,0		
2930	023542	000000							
2931	023544	001240	001234	001242	DT3:	.WORD	PPARER,PLOADR,PCONTR,PMAINT,TESTNO,BADPC,0		
2932	023552	001244	001264	001262					
2933	023560	000000							
2934	023562	001250	001252	001254	DT5:	.WORD	PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0		
2935	023570	001264	001262	000000					
2936	023576	001226	001250	001252	DT6:	.WORD	MMEXP,PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0		
2937	023604	001254	001264	001262					
2938	023612	000000							
2939	023614	001160	001264	001120	DT7:	.WORD	\$REG1,TESTNO,\$ERRPC,0		
2940	023622	000000							
2941	023624	001162	001160	001264	DT11:	.WORD	\$REG2,\$REG1,TESTNO,\$ERRPC,0		
2942	023632	001120	000000						
2943	023636	001162	001264	001120	DT12:	.WORD	\$REG2,TESTNO,\$ERRPC,0		
2944	023644	000000							
2945	023646	001164	001162	001264	DT13:	.WORD	\$REG3,\$REG2,TESTNO,\$ERRPC,0		
2946	023654	001120	000000						
2947	023660	001160			DT16:	.WORD	\$REG1		
2948	023662	001162	001264	001120	DT15:	.WORD	\$REG2,TESTNO,\$ERRPC,0		
2949	023670	000000							
2950	023672	001276	001300	001264	DT17:	.WORD	ADDROR,ADRAND,TESTNO,ERRCNT,0		
2951	023700	001302	000000						
2952	023704	001156	001162	001264	DT20:	.WORD	\$REG0,\$REG2,TESTNO,\$ERRPC,0		
2953	023712	001120	000000						
2954	023716	001276	001300	001266	DT21:	.WORD	ADDROR,ADRAND,DATAOR,DATAND,TESTNO,ERRCNT,0		
2955	023724	001270	001264	001302					
2956	023732	000000							
2957	023734	001276	001300	001272	DT22:	.WORD	ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,TESTNO,ERRCNT,0		
2958	023742	001274	001266	001270					
2959	023750	001264	001302	000000					
2960	023756	001156	001162	001160	DT23:	.WORD	\$REG0,\$REG2,\$REG1,TESTNO,\$ERRPC,0		
2961	023764	001264	001120	000000					
2962	023772	001156	001264	001120	DT24:	.WORD	\$REG0,TESTNO,\$ERRPC,0		
2963	024000	000000							
2964	024002	001156	001160	001264	DT25:	.WORD	\$REG0,\$REG1,TESTNO,\$ERRPC,0		
2965	024010	001120	000000						
2966	024014	001156	001160	001162	DT31:	.WORD	\$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0		
2967	024022	001264	001120	000000					
2968	024030	001156	001264	001120	DT35:	.WORD	\$REG0,TESTNO,\$ERRPC,0		
2969	024036	000000							
2970	024040	001156	001160	001162	DT36:	.WORD	\$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0		

2971	024046	001264	001120	000000			
2972	024054	001156	001160	001162	DT37:	.WORD	\$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0
2973	024062	001264	001120	000000			
2974	024070	001174	001176	001264	DT40:	.WORD	\$TMP1,\$TMP2,TESTNO,\$ERRPC,0
2975	024076	001120	000000				
2976	024102	001174	001264	001120	DT41:	.WORD	\$TMP1,TESTNO,\$ERRPC,0
2977	024110	000000					
2978	024112	001162	001164	001156	DT42:	.WORD	\$REG2,\$REG3,\$REG0,TESTNO,\$ERRPC,0
2979	024120	001264	001120	000000			
2980	024126	001264	001120	000000	DT43:	.WORD	TESTNO,\$ERRPC,0
2981	024134	001160	001264	001120	DT44:	.WORD	\$REG1,TESTNO,\$ERRPC,0
2982	024142	000000					
2983	024144	001156	001264	001120	DT45:	.WORD	\$REG0,TESTNO,\$ERRPC,0
2984	024152	000000					
2985	024154	172350	177760	001264	DT50:	.WORD	KIPAR4,SIZELO,TESTNO,\$ERRPC,0
2986	024162	001120	000000				
2987	024166	001160	001164	001264	DT51:	.WORD	\$REG1,\$REG3,TESTNO,\$ERRPC,0
2988	024174	001120	000000				
2989	024200	001264	001120	000000	DT53:	.WORD	TESTNO,\$ERRPC,0
2990	024206	001156	001160	001264	DT55:	.WORD	\$REG0,\$REG1,TESTNO,\$ERRPC,0
2991	024214	001120	000000				
2992	024220	177572	172310	001264	DT56:	.WORD	MMR0,KIPDR4,TESTNO,\$ERRPC,0
2993	024226	001120	000000				
2994	024232	001172	001264	001120	DT64:	.WORD	\$TMP0,TESTNO,\$ERRPC,0
2995	024240	000000					
2996	024242	001226	001250	001264	DT65:	.WORD	MMEXP,MMR0,TESTNO,\$ERRPC,0
2997	024250	001120	000000				
2998	024254	001250	001264	001120	DT66:	.WORD	MMR0,TESTNO,\$ERRPC,0
2999	024262	000000					
3000	024264	001160	001250	001252	DT70:	.WORD	\$REG1,MMR0,MMR1,MMR2,TESTNO,\$ERRPC,0
3001	024272	001254	001264	001120			
3002	024300	000000					
3003	024302	001160	001250	001264	DT72:	.WORD	\$REG1,MMR0,TESTNO,\$ERRPC,0
3004	024310	001120	000000				
3005	024314	001250	001252	001254	DT73:	.WORD	MMR0,MMR1,MMR2,\$TMP0,\$TMP1,\$TMP2,TESTNO,\$ERRPC,0
3006	024322	001172	001174	001176			
3007	024330	001264	001120	000000			
3008	024336	001250	001252	001254	DT75:	.WORD	MMR0,MMR1,MMR2,TESTNO,\$ERRPC,0
3009	024344	001264	001120	000000			
3010	024352	001160	001254	001264	DT100:	.WORD	\$REG1,MMR2,TESTNO,\$ERRPC,0
3011	024360	001120	000000				
3012	024364	001162	001250	001264	DT101:	.WORD	\$REG2,MMR0,TESTNO,\$ERRPC,0
3013	024372	001120	000000				
3014	024376	001172	001264	001120	DT102:	.WORD	\$TMP0,TESTNO,\$ERRPC,0
3015	024404	000000					
3016	024406	001164	001160	001264	DT112:	.WORD	\$REG3,\$REG1,TESTNO,\$ERRPC,0
3017	024414	001120	000000				
3018	024420	001164	001156	001264	DT113:	.WORD	\$REG3,\$REG0,TESTNO,\$ERRPC,0
3019	024426	001120	000000				
3020	024432	001162	001160	001264	DT114:	.WORD	\$REG2,\$REG1,TESTNO,\$ERRPC,0
3021	024440	025760	000000				
3022	024444	001156	001160	001264	DT116:	.WORD	\$REG0,\$REG1,TESTNO,\$ERRPC,0
3023	024452	001120	000000				
3024	024456	001250	001252	001254	DT117:	.WORD	MMR0,MMR1,MMR2,TESTNO,\$ERRPC,0
3025	024464	001264	001120	000000			
3026	024472	001160	001264	001120	DT120:	.WORD	\$REG1,TESTNO,\$ERRPC,0

3027	024500	000000								
3028	024502	001156	001264	001120	DT122:	.WORD	\$REG0,TESTNO,SERRPC,0			
3029	024510	000000								
3030	024512	001264	001120	000000	DT127:	.WORD	TESTNO,SERRPC,0			
3031	024520	001250	001252	001254	DT131:	.WORD	PMPR0,PMPR1,PMPR2,TESTNO,SERRPC,0			
3032	024526	001264	001120	000000						
3033	024534	001120	001156	001160	DT133:	.WORD	SERRPC,\$REG0,\$REG1,0			
3034	024542	000000								
3035	024544	001120	001156	001160	DT135:	.WORD	SERRPC,\$REG0,\$REG1,\$REG2,\$REG3,\$REG4,0			
3036	024552	001162	001164	001166						
3037	024560	000000								
3038	024562	001172	001174	001176	DT145:	.WORD	\$TMP0,\$TMP1,\$TMP2,TESTNO,SERRPC,0			
3039	024570	001264	001120	000000						
3040	024576	001120	001264	000000	DT146:	.WORD	SERRPC,TESTNO,0			
3041	024604	001120	001172	001174	DT147:	.WORD	SERRPC,\$TMP0,\$TMP1,TESTNO,0			
3042	024612	001264	000000							
3043	024616	001156	001264	000000	DT201:	.WORD	\$REG0,TESTNO,0			
3044	024624	001156	001160	001264	DT202:	.WORD	\$REG0,\$REG1,TESTNO,0			
3045	024632	000000								
3046	024634	001156	001162	001166	DT203:	.WORD	\$REG0,\$REG2,\$REG4,\$REG1,TESTNO,0			
3047	024642	001160	001264	000000						
3048										
3049										
3050	024650	000			DF1:	.BYTE	0			
3051	024651	000	000	000	DF2:	.BYTE	0,0,0			
3052	024654	000	002	000	DF3:	.BYTE	0,2,0,0,0,0			
3053	024657	000	000	000						
3054	024662	000	000	000	DF5:	.BYTE	0,0,0,0,0			
3055	024665	000	000							
3056	024667	000	000	000	DF6:	.BYTE	0,0,0,0,0,0			
3057	024672	000	000	000						
3058	024675	000	000	000	DF7:	.BYTE	0,0,0			
3059	024700	000	000	000	DF11:	.BYTE	0,0,0,0			
3060	024703	000								
3061	024704	000	000	000	DF12:	.BYTE	0,0,0			
3062	024707	000	000	000	DF13:	.BYTE	0,0,0,0			
3063	024712	000								
3064	024713	000	000	000	DF16:	.BYTE	0,0,0			
3065	024716	000	000	000	DF15:	.BYTE	0,0,0,0			
3066	024721	000								
3067	024722	000	000	000	DF17:	.BYTE	0,0,0,1			
3068	024725	001								
3069	024726	000	000	000	DF20:	.BYTE	0,0,0,0			
3070	024731	000								
3071	024732	000	000	000	DF21:	.BYTE	0,0,0,0,0,1			
3072	024735	000	000	001						
3073	024740	000	000	000	DF22:	.BYTE	0,0,0,0,0,0,0,1			
3074	024743	000	000	000						
3075	024746	000	001							
3076	024750	000	000	000	DF23:	.BYTE	0,0,0,0,0			
3077	024753	000	000							
3078	024755	000	000	000	DF24:	.BYTE	0,0,0			
3079	024750	000	000	000	DF25:	.BYTE	0,0,0,0			
3080	024763	000								
3081	024764	000	000	000	DF31:	.BYTE	0,0,0,0,0			
3082	024767	000	000							

3083	024771	000	000	000	DF35:	.BYTE	0,0,0
3084	024774	000	000	000	DF36:	.BYTE	0,0,0,0,0
3085	024777	000	000				
3086	025001	003	000	000	DF37:	.BYTE	3,0,0,0,0
3087	025004	000	000				
3088	025006	004	004	000	DF40:	.BYTE	4,4,0,0
3089	025011	000					
3090	025012	004	000	000	DF41:	.BYTE	4,0,0
3091	025015	000	000	003	DF42:	.BYTE	0,0,3,0,0
3092	025020	000	000				
3093	025022	000	000		DF43:	.BYTE	0,0
3094	025024	000	000	000	DF44:	.BYTE	0,0,0
3095	025027	003	000	000	DF45:	.BYTE	3,0,0
3096	025031	004	004	000	DF50:	.BYTE	4,4,0,0
3097	025034	000					
3098	025036	000	000	000	DF51:	.BYTE	0,0,0,0
3099	025041	000					
3100	025042	000	000		DF53:	.BYTE	0,0
3101	025044	000	000	000	DF55:	.BYTE	0,0,0,0
3102	025047	000					
3103	025050	000	000	000	DF56:	.BYTE	0,0,0,0
3104	025053	000					
3105	025054	000	000	000	DF64:	.BYTE	0,0,0
3106	025057	000	000	000	DF65:	.BYTE	0,0,0,0
3107	025062	000					
3108	025063	000	000	000	DF66:	.BYTE	0,0,0
3109	025066	000	000	000	DF70:	.BYTE	0,0,0,0,0,0
3110	025071	000	000	000			
3111	025074	000	000	000	DF72:	.BYTE	0,0,0,0
3112	025077	000					
3113	025100	000	000	000	DF73:	.BYTE	0,0,0,0,0,0,0,0
3114	025103	000	000	000			
3115	025106	000	000				
3116	025110	000	000	000	DF75:	.BYTE	0,0,0,0,0
3117	025113	000	000				
3118	025115	000	000	000	DF100:	.BYTE	0,0,0,0
3119	025120	000					
3120	025121	000	000	000	DF101:	.BYTE	0,0,0,0
3121	025124	000					
3122	025125	000	000	000	DF102:	.BYTE	0,0,0
3123	025130	000	000	000	DF112:	.BYTE	0,0,0,0
3124	025133	000					
3125	025134	000	000	000	DF113:	.BYTE	0,0,0,0
3126	025137	000					
3127	025140	000	000	000	DF114:	.BYTE	0,0,0,0
3128	025143	000					
3129	025144	000	000	000	DF116:	.BYTE	0,0,0,0
3130	025147	000					
3131	025150	000	000	000	DF117:	.BYTE	0,0,0,0,0
3132	025153	000	000				
3133	025155	000	000	000	DF120:	.BYTE	0,0,0
3134	025160	000	000	000	DF122:	.BYTE	0,0,0
3135	025163	000	000		DF127:	.BYTE	0,0
3136	025165	000	000	000	DF131:	.BYTE	0,0,0,0,0
3137	025170	000	000				
3138							

```

3139
3140
3141 025172 000 000 DF201: .BYTE 0,0
3142 025174 000 000 000 DF202: .BYTE 0,0,0
3143 025177 000 000 000 DF203: .BYTE 0,0,0,0,0
3144 025202 000 000
3145 .EVEN
3146
3147
3148 ;:*****
3149
3150 .SBTTL END OF PASS ROUTINE
3151
3152 ;*INCREMENT THE PASS NUMBER ($PASS)
3153 ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
3154 ;*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYYY'
3155 ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
3156 ;*IF SW12=1 INHIBIT TRACE TRAP
3157 ;*IF THERES A MONITOR GO TO IT
3158 ;*IF THERE ISN'T JUMP TO LOOP
3159
3160 025204 SEOP:
3161 025204 000240 NOP
3162 025206 005037 001102 CLR $TSTNM ;:ZERO THE TEST NUMBER
3163 025212 005037 001206 CLR $TIMES ;:ZERO THE NUMBER OF ITERATIONS
3164 025216 005237 001100 INC $PASS ;:INCREMENT THE PASS NUMBER
3165 025222 042737 100000 001100 BIC #100000,$PASS ;:DON'T ALLOW A NEG. NUMBER
3166 025230 005327 DEC (PC)+ ;:LOOP?
3167 025232 000001 SEOPCT: .WORD 1
3168 025234 003072 BGT $DOAGN ;:YES
3169 025236 012737 MOV (PC)+,@(PC)+ ;:RESTORE COUNTER
3170 025240 000001 SENDCT: .WORD 1
3171 025242 025232 SEOPCT
3172 025244 104400 025252 TYPE ,65$ ;:TYPE ASCIZ STRING
3173 025250 000407 BR ,64$ ;:GET OVER THE ASCIZ
3174 ;:65$: .ASCIZ <12><15>/END PASS #/
3175 025270 ;:64$:
3176 025270 013746 001100 MOV $PASS,-(SP) ;:SAVE $PASS FOR TYPEOUT
3177 ;:TYPE PASS NUMBER
3178 025274 104410 TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
3179 025276 104400 025304 TYPE ,67$ ;:TYPE ASCIZ STRING
3180 025302 000421 BR ,66$ ;:GET OVER THE ASCIZ
3181 ;:67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
3182 ;:66$:
3183 025346 013746 001114 MOV $ERTTL,-(SP) ;:SAVE $ERTTL FOR TYPEOUT
3184 ;:TOTAL NUMBER OF ERRORS
3185 025352 104410 TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
3186 025354 104400 001217 TYPE ,SCLRF ;:TYPE CARRIAGE RETURN, LINE FEED
3187 025360 005037 001114 CLR $ERTTL ;:CLEAR ERROR TOTAL
3188 025364 013700 000042 $GET42: MOV @#42,R0 ;:GET MONITOR ADDRESS
3189 025370 001414 BEQ $DOAGN ;:BRANCH IF NO MONITOR
3190 025372 005046 CLR -(SP) ;:INSURE THE 'T' BIT IS CLEAR
3191 025374 012746 025402 MOV #SCLR.T,-(SP) ;:SETUP FOR AN RTI OR RTT
3192 025400 000427 BR $RTRN ;:GO DO AN RTI OR RTT TO LOAD THE PSW
3193 ;:WITH A CLEARED 'T' BIT
3194 025402 $CLR.T:
    
```

```

3195 025402 013700 000042      MOV      @#42,R0      ;;INSURE R0 CONTAINS THE MONITORS
3196 025406 001405              BEQ      $DOAGN      ;;RETURN ADDRESS
3197 025410 000005              RESET             ;;CLEAR THE WORLD
3198 025412 004710      SENDAD: JSR      PC,(R0)  ;;GO TO MONITOR
3199 025414 000240              NOP              ;;SAVE ROOM
3200 025416 000240              NOP              ;;FOR
3201 025420 000240              NOP              ;;ACT11
3202 025422              SDOAGN:
3203 025422 013746 177776      MOV      @#PS,-(SP)  ;;PUT THE PS ON THE STACK AND
3204 025426 042716 000020      BIC      #20,(SP)   ;;CLEAR THE 'T' BIT
3205 025432 032737 010000 177570      BIT      #BIT12,@#SWR  ;;RUN WITH TRACE TRAP?
3206 025440 001005              BNE      1$        ;;BR IF NO
3207 025442 005137 025466      COM      $TBIT      ;;IS IT TIME FOR TRACE TRAP
3208 025446 100402              BMI      1$        ;;BR IF NO
3209 025450 052716 000020      BIS      #20,(SP)   ;;SET TRACE TRAP
3210 025454 012746 025462      1$:      MOV      #SLOOP,-(SP)  ;;JUMP TO START OF TEST
3211 025460 000002      $RTN:    RTI        ;;RETURN--THIS IS CHANGED TO
                          ;;AN 'RTT' IF 'RTT' IS A LEGAL
                          ;;INSTRUCTION
3212
3213
3214 025462              SLOOP:
3215 025462 000137 040272      JMP      @#LOOP     ;;RETURN
3216 025466 000000              $TBIT:  .WORD      0      ;;'T' BIT STATE INDICATOR
3217 025470 377 377 000      $ENULL: .BYTE      -1,-1,0  ;;NULL CHARACTER STRING
3218              .EVEN
3219      ;;*****
3220
3221      .SBTTL  SCOPE HANDLER ROUTINE
3222
3223      ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
3224      ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
3225      ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
3226      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
3227      ;*SW14=1      LOOP ON TEST
3228      ;*SW11=1      INHIBIT ITERATIONS
3229      ;*SW09=1      LOOP ON ERROR
3230      ;*SW08=1      LOOP ON TEST IN SWR<6:0>
3231      ;*CALL
3232      ;*      SCOPE      ;;SCOPE=IOT
3233
3234      $SCOPE:
3235 025474 005037 001314      CLR      RETRY      ;;CLEAR THE RETRY FLAG BEFORE THIS TEST
3236 025500 005037 001302      CLR      ERRCNT     ;;CLEAR THE MULTIPLE ERROR COUNTER
3237 025504 006137 177570      ROL      @#SWR      ;;LOOP ON PRESENT TEST?
3238 025510 100514      BMI      $OVER      ;;YES IF SW14=1
3239      ;*****START OF CODE FOR THE XOR TESTER*****
3240 025512 000416      $XTSTR: BR      6$   ;;IF RUNNING ON THE 'X' TESTER CHANGE
3241              ;;THIS INSTRUCTION TO A 'JOP' (NOP=240)
3242 025514 013746 000004      MOV      @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
3243 025520 012737 025540 000004      MOV      #5$,@#ERRVEC  ;;SET FOR TIMEOUT
3244 025526 005737 177060      TST      @#177060     ;;TIME OUT ON XOR?
3245 025532 012637 000004      MOV      (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
3246 025536 000466              BR      $$VLAD      ;;GO TO THE NEXT TEST
3247 025540 022626      5$:      CMP      (SP)+,(SP)+  ;;CLEAR THE STACK AFTER A TIME OUT
3248 025542 012637 000004      MOV      (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
3249 025546 000426              BR      7$        ;;LOOP ON THE PRESENT TEST
3250 025550              6$::*****END OF CODE FOR THE XOR TESTER*****
    
```

```

3251 025550 032737 000400 177570 BIT #BIT08,@#SWR ;;LOOP ON SPFC. TEST?
3252 025556 001407 BEQ 2$ ;;BR IF NO
3253 025560 013746 177570 MOV @#SWR,-(SP) ;;SET DESIRED TEST NUM. FROM SWR
3254 025564 042716 000200 BIC #SSWRM#, (SP) ;;STRIP AWAY UNDESIRED BITS
3255 025570 122637 001102 CMPB (SP)+,$STNM ;;ON THE RIGHT TEST?
3256 025574 001462 BEQ $OVER ;;BR IF YES
3257 025576 105737 001103 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
3258 025602 001421 BEQ 3$ ;;BR IF NO
3259 025604 123737 001117 001103 CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
3260 025612 101015 BHI 3$ ;;BR IF NO
3261 025614 032737 001000 177570 BIT #BIT09,@#SWR ;;LOOP ON ERROR?
3262 025622 001404 BEQ 4$ ;;BR IF NO
3263 025624 013737 001112 001110 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
3264 025632 000443 BR $OVER
3265 025634 105037 001103 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
3266 025640 005037 001206 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
3267 025644 000415 BR 1$ ;;ESCAPE TO THE NEXT TEST
3268 025646 032737 004000 177570 3$: BIT #BIT11,@#SWR ;;INHIBIT ITERATIONS?
3269 025654 001011 BNE 1$ ;;BR IF YES
3270 025656 005737 001100 TST $PASS ;;IF FIRST PASS OF PROGRAM
3271 025662 001406 BEQ 1$ ;; INHIBIT ITERATIONS
3272 025664 005237 001106 INC $ICNT ;;INCREMENT ITERATION COUNT
3273 025670 023737 001206 001106 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
3274 025676 002021 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
3275 025700 012737 000001 001106 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
3276 025706 013737 025756 001206 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
3277 025714 105237 001102 $SVLAD: INCB $STNM ;;COUNT TEST NUMBERS
3278 025720 011637 001110 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
3279 025724 011637 001112 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
3280 025730 005037 001210 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
3281 025734 112737 000001 001117 MOVB #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
3282 025742 013737 001102 177570 $OVER: MOV $STNM,@#DISPLAY ;;DISPLAY TEST NUMBER
3283 025750 013716 001110 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
3284 025754 000002 RTI ;;FIXES PS
3285 025756 000310 $MXCNT: 200. ;;MAX. NUMBER OF ITERATIONS
3286 ;;*****
3287
3288 .SBTTL ERROR HANDLER ROUTINE
3289
3290 ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
3291 ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
3292 ;*AND GO TO ERTYPE ON ERROR
3293 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
3294 ;*SW15=1 HALT ON ERROR
3295 ;* HALT CAN OCCUR BEFORE AND AFTER THE ERROR TYPEOUT
3296 ;*SW13=1 INHIBIT ERROR TYPEOUTS
3297 ;*SW10=1 BELL ON ERROR
3298 ;*SW09=1 LOOP ON ERROR
3299 ;*CALL
3300 ;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
3301
3302 025760 $ERROR: MOVB $STNM,TESTNO ;;SAVE TEST NUMBER FOR ERROR TYPE OUT
3303 025760 113737 001102 001264 INC ERRCNT ;;KEEP COUNT OF MULTIPLE ERRORS
3304 025766 005237 001302 MOV R0,$REG0 ;;SAVE R0
3305 025772 010037 001156 MOV R1,$REG1 ;;SAVE R1
3306 025776 010137 001160
    
```



```

3307 026002 010237 001162      MOV     R2,$REG2      ;SAVE R2
3308 026006 010337 001164      MOV     R3,$REG3      ;SAVE R3
3309 026012 010437 001166      MOV     R4,$REG4      ;SAVE R4
3310 026016 010537 001170      MOV     R5,$REG5      ;SAVE R5
3311 026022 105237 001103      7$:    INCB    $ERFLG    ;SET THE ERROR FLAG
3312 026026 001775                BEQ     7$            ;DON'T LET THE FLAG GO TO ZERO
3313 026030 013737 001102 177570    MOV     $STNM,@#DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
3314 026036 005737 177570      TST     @#SWR        ;HALT ON ERROR = 1?
3315 026042 100001                BPL     8$            ;BRANCH IF NO
3316 026044 000000                HALT                    ;YES--HALT
3317 026046 032737 002000 177570    8$:    BIT     #BIT10,@#SWR ;BELL ON ERROR?
3318 026054 001402                BEQ     1$            ;NO - SKIP
3319 026056 104400 001212      TYPE    ,SBELL        ;RING BELL
3320 026062 005237 001114      1$:    INC     $ERTTL      ;COUNT THE NUMBER OF ERRORS
3321 026066 011637 001120      MOV     (SP),$ERRPC   ;GET ADDRESS OF ERROR INSTRUCTION
3322 026072 162737 000002 001120    SUB     #2,$ERRPC
3323 026100 117737 153014 001116    MOVB   @$ERRPC,$ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE
3324 026106 032737 020000 177570    BIT     #BIT13,@#SWR ;SKIP TYPEOUT IF SET
3325 026114 001004                BNE     2$            ;SKIP TYPEOUTS
3326 026116 004737 026666      JSR     PC,ERTYPE     ;GO TO USER ERROR ROUTINE
3327 026122 104400 001217      TYPE    ,$CRLF
3328 026126 005737 177570      2$:    TST     @#SWR        ;HALT ON ERROR
3329 026132 100001                BPL     9$            ;SKIP IF CONTINUE
3330 026134 000000                HALT                    ;HALT ON ERROR!
3331 026136 022737 025412 000042    9$:    CMP     #SENDAD,42 ;ACT-11?
3332 026144 001001                BNE     3$            ;BRANCH IF NO
3333 026146 000000                HALT                    ;YES
3334 026150 032737 001000 177570    3$:    BIT     #BIT09,@#SWR ;LOOP ON ERROR SWITCH SET?
3335 026156 001402                BEQ     4$            ;BR IF NO
3336 026160 013716 001112      MOV     $LPERR,(SP)   ;FUDGE RETURN FOR LOOPING
3337 026164 005737 001210      4$:    TST     $ESCAPE    ;CHECK FOR AN ESCAPE ADDRESS
3338 026170 001402                BEQ     5$            ;BR IF NONE
3339 026172 013716 001210      MOV     $ESCAPE,(SP) ;FUDGE RETURN ADDRESS FOR ESCAPE
3340 026176                5$:
3341 026176 032737 001000 177570      BIT     #SW9,SWR      ;IS THE LOOP ON ERROR SWITCH UP?
3342 026204 001421                BEQ     EREXIT        ;BRANCH IF NOT LOOPING ON ERROR
3343 026206 005037 177766      CLR     CPUERR        ;CLEAR CPU ERROR REGISTER
3344 026212 012737 177777 177744    MOV     #-1,MEMERR    ;CLEAR MEMORY ERROR REGISTER
3345 026220 042737 176776 177572    BIC     #176776,#PRO  ;CLEAR MEMORY MANAGEMENT REG 0
3346                ;LEAVE BIT0 AND BIT9 UNCHANGED
3347 026226 012737 177777 031672    MOV     #-1,CPFLAG    ;RE-INITIALIZE CP TRAP FLAG
3348 026234 012737 177777 031776    MOV     #-1,PAFLAG    ;RE-INITIALIZE PARITY TRAP FLAG
3349 026242 012737 177777 032230    MOV     #-1,MMFLAG    ;RE-INITIALIZE MEMORY MANAGE. TRAP FLAG
3350 026250 000002      EREXIT: RTI          ;RETURN TO TEST AFTER ERROR
3351                ;SBTTL SPURIOUS ERROR HANDLER
3352                ;* THIS ROUTINE IS ENTERED BY AN UNEXPECTED TRAP TO 4 OR 114.
3353                ;* IF SWITCH 13 IS OFF, AN ERROR MESSAGE, THE ERROR REGISTER,
3354                ;* THE ERROR PC, AND THE TEST NUMBER ARE TYPED OUT.
3355                ;* IF SWITCH 13 IS ON, ONLY THE ERROR MESSAGE WILL BE TYPED.
3356                ;*****
3357 026252 011637 001120      CPUSPUR:MOV (SP),$ERRPC ;SAVE THE ERROR PC
3358 026256 012706 001100      MOV     #STACK,SP    ;RESTORE THE SP
3359 026262 104400 026270      TYPE    ,65$         ;TYPE ASCIZ STRING
3360 026266 000414                BR      64$          ;GET OVER THE ASCIZ
3361                ;:65$: .ASCIZ /UNEXPECTED TRAP TO 4/<15><12>
3362 026320      64$:
    
```

```

3363 026320 032737 020000 177570      BIT      #BIT13,@#SWR      ;IS INHIBIT ERROR TYPEOUT ON?
3364 026326 001045                      BNE      1$              ;BRANCH IF YES
3365 026330 104400 026336              TYPE     ,67$           ;:TYPE ASCIZ STRING
3366 026334 000417                      BR       66$            ;:GET OVER THE ASCIZ
3367                                     ;:67$: .ASCIZ /ERRORPC TSTNUM CPUERR REG/<15><12>
3368 026374                                     66$:
3369 026374 013737 177766 001172      MOV      @#CPUERR,$TMP0 ;GET CPU ERROR REG
3370 026402 113737 001102 001104      MOV      $TSTNM,$$TSTNM ;GET TEST NUMBER
3371 026410 013746 001120              MOV      $ERRPC,-(SP)   ;SAVE $ERRPC FOR TYPEOUT
3372                                     ;:TYPE ERROR PC
3373 026414 104402                      TYP0C   ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
3374 026416 104400 023526              TYPE     ,TWOSP
3375 026422 013746 001104              MOV      $$TSTNM,-(SP) ;SAVE $$TSTNM FOR TYPEOUT
3376                                     ;:TYPE TEST NUMBER
3377 026426 104402                      TYP0C   ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
3378 026430 104400 023526              TYPE     ,TWOSP
3379 026434 013746 001172              MOV      $TMP0,-(SP)   ;SAVE $TMP0 FOR TYPEOUT
3380                                     ;:TYPE ERROR REGISTER
3381 026440 104402                      TYP0C   ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
3382 026442 005037 177766 001210 1$:   CLR      @#CPUERR      ;CLEAR CPU ERROR REG
3383 026446 013737 001316 001210      MOV      N$TST,$ESCAPE ;SETUP ESCAPE ADDRESS
3384 026454 104171                      ERROR   171            ;MAKE THE ERROR CALL TO SYSMAC
3385 026456 011637 001120  CACHSPU:MOV (SP),$ERRPC ;SAVE ERROR PC
3386 026462 012706 001100              MOV      #STACK,SP    ;RESTORE STACK
3387 026466 104400 026474              TYPE     ,65$           ;:TYPE ASCIZ STRING
3388 026472 000415                      BR       64$            ;:GET OVER THE ASCIZ
3389                                     ;:65$: .ASCIZ /UNEXPECTED TRAP TO 114/<15><12>
3390 026526                                     64$:
3391 026526 032737 020000 177570      BIT      #BIT13,@#SWR      ;IS SWITCH 13 ON?
3392 026534 001045                      BNE      1$              ;BRANCH IF YES
3393 026536 104400 026544              TYPE     ,67$           ;:TYPE ASCIZ STRING
3394 026542 000417                      BR       66$            ;:GET OVER THE ASCIZ
3395                                     ;:67$: .ASCIZ /ERRORPC TSTNUM MEMERR REG/<15><12>
3396 026602                                     66$:
3397 026602 013737 177744 001172      MOV      @#MEMERR,$TMP0 ;SAVE MEMORY ERROR REG
3398 026610 113737 001102 001104      MOV      $TSTNM,$$TSTNM ;SAVE TEST NUMBER
3399 026616 013746 001120              MOV      $ERRPC,-(SP)   ;SAVE $ERRPC FOR TYPEOUT
3400                                     ;:TYPE ERROR PC
3401 026622 104402                      TYP0C   ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
3402 026624 104400 023526              TYPE     ,TWOSP
3403 026630 013746 001104              MOV      $$TSTNM,-(SP) ;SAVE $$TSTNM FOR TYPEOUT
3404                                     ;:TYPE TEST NUMBER
3405 026634 104402                      TYP0C   ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
3406 026636 104400 023526              TYPE     ,TWOSP
3407 026642 013746 001172              MOV      $TMP0,-(SP)   ;SAVE $TMP0 FOR TYPEOUT
3408                                     ;:TYPE MEM ERROR REG
3409 026646 104402                      TYP0C   ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
3410 026650 013737 177744 177744 1$:   MOV      @#MEMERR,@#MEMERR ;CLEAR MEMERR REG.
3411 026656 013737 001316 001210      MOV      N$TST,$ESCAPE ;SETUP ESCAPE ADDRESS
3412 026664 104171                      ERROR   171
3413
3414
3415                                     .SBTTL  ERROR MESSAGE TYPE OUT ROUTINE
3416
3417
3418                                     ;*      THIS SUBROUTINE IS CALLED BY THE ERROR HANDLER TO TYPE
    
```

```

3419      * THE ERROR MESSAGES. IT PICKS UP THE ITEM BYTE ($ITEMB) NUMBER
3420      * AND USES THAT TO INDEX THROUGH THE ERROR TABLE. THE ERROR
3421      * TABLE STARTS AT 'SERRTB' AND HAS FOUR (4) POINTERS FOR EACH
3422      * ENTRY, 'EM', 'DH', 'DT', 'DF'. THE 'EM' POINTS TO THE ERROR
3423      * MESSAGE WHICH IS AN ASCIZ STRING. THE 'DH' POINTS TO THE DATA
3424      * HEADER WHICH IS ANOTHER ASCIZ STRING. THE 'DT' POINTS TO THE
3425      * DATA TABLE WHICH IS A GROUP OF WORDS CONTAINING THE ADDRESSES
3426      * OF THE DATA TO BY TYPED. THE FORMAT OF THIS DATA IS
3427      * CONTROLLED BY THE 'DF' WHICH IS THE POINTER TO THE DATA FORMAT.
3428      * THE DATA FORMAT IS A GROUP OF BYTES WHICH CONTAIN NUMBERS
3429      * THAT CORRESPOND TO DIFFERENT TYPING FORMATS.
3430      * 0 -16 BIT OCTAL FORMAT
3431      * 1 -DECIMAL FORMAT
3432      * 2 -22 BIT OCTAL FORMAT. DATA IS LOWER 16 BITS OF THE
3433      * PHYSICAL ADDRESS, UPPER 6 BITS ARE ADJACENT TO LOWER 16
3434      * 3 -22 BIT OCTAL FORMAT. DATA IS THE 16 BIT VIRTUAL
3435      * ADDRESS IN KERNEL I-SPACE.
3436      * 4 -22 BIT OCTAL FORMAT. DATA IS A P.A.R. AND THE
3437      * OUTPUT IS THE BASE ADDRESS THAT THE P.A.R. POINTS TO.
3438 026666 010046      * ERTYPE: MOV R0,-(KSP) ;SAVE R0 ON STACK
3439 026670 005000      * CLR R0 ;CLEAR R0
3440 026672 113700 001116      * MOVB @#$ITEMB,R0 ;PUT ITEM NUMBER IN R0
3441 026676 001004      * BNE 1$ ;BRANCH IF IT IS NON-ZERO
3442 026700 013746 001120      * MOV SERRPC,-(KSP) ;PUT ERROR PC ON STACK FOR TYPING
3443 026704 104402      * TYPOC ;TYPE FAILING PC
3444 026706 000551      * BR 13$ ;GO TO RETURN
3445 026710 005300      * 1$: DEC R0 ;ADJUST ITEM NUMBER TO BE A FOINTER
3446 026712 072027 000003      * ASH #3,R0 ;LEFT SHIFT ITEM NO. 3 PLACES
3447 026716 100024      * BPL 22$ ;BRANCH IF ITEM #LESS THAN 200
3448 026720 032700 001000      * BIT #BIT9,R0 ;SEE IF ITEM # WAS 3XX
3449 026724 001415      * BEQ 21$ ;BRANCH IF ITEM # WAS 2XX
3450      * ;AND TYPE ERROR MESSAGE
3451 026726 032737 000200 177570      * BIT #SW7,@$SWR ;SEE IF SWITCH 7 IS UP
3452 026734 001404      * BEQ 20$ ;BRANCH IF SWITCH IS NOT UP
3453      * ;AND TYPE DATA, ON MULTIPLE ERRORS
3454 026736 062766 000004 000002      * ADD #4,2(KSP) ;SKIP 'TYPE $CRLF' IF SW 7 IS UP
3455      * ;INHIBIT MULTIPLE ERROR TYPEOUTS
3456 026744 000532      * BR 13$ ;BRANCH TO EXIT
3457 026746 042700 177000      * 20$: BIC #177000,R0 ;CLEAR UPPER BYTE OF R0
3458 026752 062700 003300      * ADD #ER200+4,R0 ;POINT TO DATA TABLE ENTRY
3459 026756 000424      * BR 5$ ;GO TYPE DATA TABLE
3460 026760 042700 177000      * 21$: BIC #177000,R0 ;CLEAR UPPER BYTE OF R0
3461 026764 062700 001710      * ADD #<ER200-SERRTB>,R0 ;ADD DIFFERENCE BETWEEN
3462      * ;ITEM 1 AND ITEM 201
3463      * GET POINTER TO ERROR MESSAGE AND TYPE IT
3464      * IF THE POINTER IS NOT ZERO
3465 026770 062700 001364      * 22$: ADD #SERRTB,R0 ;ADD BASE OF ERROR TABLE
3466 026774 012037 027004      * MOV (R0)+,2$ ;PUT MESSAGE POINTER IN TYPE STATEMENT
3467 027000 001404      * BEQ 3$ ;BRANCH IF NO ERROR MESSAGE
3468 027002 104400      * TYPE ;TYPE ERROR MESSAGE
3469 027004 000000      * 2$: .WORD 0 ;POINTER TO ERROR MESSAGE
3470 027006 104400 001217      * TYPE $CRLF ;TYPE CRLF
3471      * GET THE POINTER TO THE DATA HEADER AND
3472      * TYPE IT IF THE POINTER IS NOT ZERO
3473 027012 012037 027022      * 3$: MOV (R0)+,4$ ;PUT HEADER POINTER IN TYPE STATEMENT
3474 027016 001404      * BEQ 5$ ;BRANCH IF NO DATA HEADER
    
```

```

3475 027020 104400
3476 027022 000000
3477 027024 104400 001217
3478
3479
3480
3481
3482 027030 010146
3483 027032 012001
3484 027034 001475
3485 027036 012000
3486 027040 105710
3487 027042 001003
3488
3489 027044 013146
3490 027046 104402
3491 027050 000461
3492 027052 122710 000001
3493 027056 001003
3494
3495 027060 013146
3496 027062 104410
3497 027064 000453
3498 027066 122710 000002
3499 027072 001012
3500
3501 027074 012146
3502 027076 004737 031144
3503 027102 062716 000003
3504 027106 012637 027114
3505 027112 104400
3506 027114 000000
3507 027116 000436
3508 027120 122710 000003
3509 027124 001004
3510
3511
3512 027126 013146
3513 027130 004737 027242
3514 027134 000427
3515
3516
3517 027136 010246
3518 027140 010346
3519 027142 013103
3520 027144 005002
3521 027146 073227 000006
3522 027152 010237 001232
3523 027156 010337 001230
3524 027162 012746 001230
3525 027166 004737 031144
3526 027172 062716 000003
3527 027176 012637 027204
3528 027202 104400
3529 027204 000000
3530 027206 011603

```

```

4$: TYPE
      .WORD 0
      TYPE ,SCLF
      THIS IS THE START OF THE DATA OUTPUT IF THE
      DATA POINTER IS NOT ZERO. R0 POINTS TO THE
      DATA FORMAT, R1 POINTS TO THE ADDRESS OF
      THE DATA WORDS.
5$: MOV R1,-(KSP) ;SAVE R1 ON THE STACK
      MOV (R0)+,R1 ;PUT DATA TABLE POINTER IN R1
      BEQ 12$ ;BRANCH IF NO DATA TABLE
      MOV (R0)+,R0 ;PICK UP DATA FORMAT POINTER
6$: TSTB (R0) ;IS THIS WORD OCTAL
      BNE 7$ ;BRANCH IF NOT 16-BIT OCTAL
      THIS IS 16 BIT OCTAL FORMAT (DF = 0)
      MOV @ (R1)+,-(KSP) ;PUT WORD ON STACK FOR TYPING
      TYPOC ;TYPE THE WORD ON STACK AS 16 BIT OCTAL
      BR 11$ ;GET READY FOR NEXT WORD
7$: CMPB #1,(R0) ;IS THE WORD DECIMAL
      BNE 8$ ;BRANCH IF NOT DECIMAL
      THIS IS DECIMAL FORMAT (DF = 1)
      MOV @ (R1)+,-(KSP) ;PUT WORD ON STACK FOR TYPING
      TYPDS ;TYPE THE WORD ON STACK AS DECIMAL
      BR 11$ ;GET READY FOR NEXT WORD
8$: CMPB #2,(R0) ;IS WORD 22-BIT PHYSICAL ADDRESS
      BNE 9$ ;BRANCH IF NOT 22-BIT PHYSICAL ADDR
      THIS IS 22-BIT PHYSICAL FORMAT (DF = 2)
      MOV (R1)+,-(KSP) ;PUT ADDR OF LOW WORD ON STACK
      JSR PC,$DB20 ;CONVERT NUMBER TO OCTAL ASCIZ
      ADD #3,(KSP) ;ONLY WANT 8 DIGITS
      MOV (KSP)+,30$ ;PUT POINTER AFTER 'TYPE' CALL
      TYPE ;TYPE ASCIZ STRING
30$: .WORD 0 ;WORD HOLDS POINTER TO ASCIZ STRING
      BR 11$ ;GET READY FOR NEXT WORD
9$: CMPB #3,(R0) ;IS THIS A 16-BIT VIRTUAL ADDRESS
      BNE 10$ ;BRANCH IF NOT 16-BIT VIRT. ADDR.
      THIS IS 22-BIT VIRTUAL ADDRESS FORMAT
      KERNEL ;-SPACE ASSUMED. (DF = 3)
      MOV @ (R1)+,-(KSP) ;PUT 16-BIT VIRTUAL ADDR ON STACK
      JSR PC,TYPVAD ;GO TYPE 22-BIT ADDRESS FROM 16-BIT V.A.
      BR 11$ ;GET READY FOR NEXT WORD
      THIS IS FORMAT 4. DATA WORD IS A P.A.R.
      OUTPUT WILL BE 22-BIT. PAR LEFT SHIFTED 6.
10$: MOV R2,-(KSP) ;SAVE R2 ON THE STACK
      MOV R3,-(KSP) ;SAVE R3 ON THE STACK
      MOV @ (R1)+,R3 ;LOAD DATA WORD INTO R3
      CLR R2 ;R2 WILL HOLD UPPER SIX BITS OF NUMBER
      ASHC #6,R2 ;LEFT SHIFT <R2:R3> 6 PLACES
      MOV R2,PADRSR ;STORE UPPER BITS OF ADDRESS
      MOV R3,PADRSL ;STORE LOWER 16 BITS OF ADDRESS
      MOV #PADRSL,-(KSP) ;PUT ADDRESS OF LOWER 16 BITS ON STACK
      JSR PC,$DB20 ;CONVERT TWO WORDS TO ASCIZ
      ADD #3,(KSP) ;I ONLY WANT 8 DIGITS
      MOV (KSP)+,31$ ;PUT POINTER AFTER TYPE CALL
31$: TYPE ;POINTER TO ASCIZ STRING FOLLOWS
      .WORD 0 ;POINTER TO ASCIZ STRING
      MOV (KSP)+,R3 ;RESTORE R3 FROM STACK

```

```

3531 027210 012602          MOV      (SP)+,R2      ;RESTORE R2 FROM STACK
3532 027212 000400          BR       .1$          ;GET READY FOR NEXT WORD
3533 027214 005200          11$: INC      R0      ;POINT TO NEXT FORMAT BYTE
3534 027216 104400 027236  TYPE      .32$      ;TYPE TWO SPACES
3535 027222 005711          TST     (R1)         ;IS THERE ANOTHER WORD?
3536 027224 001401          BEQ     12$          ;BRANCH IF ALL DONE
3537 027226 000704          BR      6$          ;GO BACK FOR NEXT NUMBER
3538 027230 012601          12$: MOV     (KSP)+,R1 ;RESTORE R1
3539 027232 012600          13$: MOV     (KSP)+,R0 ;RESTORE R0
3540 027234 000207          RTS     PC          ;RETURN TO ERROR ROUTINE
3541 027236 020040 000     32$: .ASCIZ  ? ?      ;TWO SPACES
3542          027242
3543
3544
3545          .SBTTL CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS
3546          ;*
3547          ;*
3548          ;* THIS ROUTINE IS CALLED BY A 'JSR PC' AFTER THE VIRTUAL ADDRESS
3549          ;* IS PUSHED ON THE KERNEL STACK. THE V.A. IS THEN LOADED INTO
3550          ;* R1 AND THE UPPER 3 BITS ARE SHIFTED INTO R0 TO SELECT THE
3551          ;* CORRECT KERNEL I-SPACE PAR. THE LOWER 12 BITS OF THE VIRTUAL
3552          ;* ADDRESS ARE ADDED TO THE PAR AS THEY ARE BY MEMORY MANAGEMENT
3553          ;* AND THE PHYSICAL ADDRESS IS SAVED IN MEMORY TO BE CONVERTED
3554          ;* TO ASCIZ AND TYPED.
3555 027242 104412          TYPVAD: SAVREG      ;SAVE ALL REGISTERS
3556 027244 016601 000002  MOV     2(KSP),R1    ;PUT VIRTUAL ADDR IN R1
3557 027250 005000          CLR     R0          ;CLEAR R0 FOR CALCULATIONS
3558 027252 073027 000003  ASHC   #3,R0        ;LEFT SHIFT R0,R1 3 PLACES
3559 027256 006300          ASL    R0          ;LEFT SHIFT R0 ONE MORE PLACE
3560 027260 006001          ROR    R1          ;RIGHT SHIFT R1 SO OFFSET IS CORRECT
3561 027262 006001          ROR    R1          ;RIGHT SHIFT R1
3562 027264 006001          ROR    R1          ;RIGHT SHIFT R1
3563 027266 062700 172340  ADD     #KIPAR0,R0   ;FORM DESIRED PAR ADDR IN R0
3564 027272 011003          MOV     (R0),R3     ;PUT CONTENTS OF PAR IN R3
3565 027274 005002          CLR     R2          ;CLEAR R2 FOR PHYSICAL ADDR CALCULATIONS
3566 027276 073227 000006  ASHC   #6,R2        ;LEFT SHIFT <R2,R3> 6 PLACES
3567 027302 060103          ADD     R1,R3       ;ADD OFFSET IN R1 TO BASE IN R3
3568 027304 005502          ADC    R2          ;ADD ANY POSSIBLE CARRY TO UPPER 6 BITS
3569 027306 010237 001232  MOV     R2,PADRSR   ;PUT UPPER 6 BITS OF ADDR IN CORE
3570 027312 010337 001230  MOV     R3,PADRSL   ;PUT LOWER 16 BITS OF ADDR IN CORE
3571 027316 012746 001230  MOV     #PADRSL,-(KSP) ;PUT POINTER TO LOWER 16 BITS ON STACK
3572 027322 004737 031144  JSR     PC,$DB20    ;CONVERT NUMBER TO OCTAL ASCIZ
3573 027326 062716 000003  ADD     #3,(KSP)    ;ONLY TYPE 8 DIGITS
3574 027332 012637 027340  MOV     (KSP)+,3$   ;PUT POINTER AFTER TYPE INST
3575 027336 104400          TYPE          ;TYPE THE 22-BIT VIRTUAL ADDRESS
3576 027340 000000          3$: .WORD 0        ;THIS WORD HOLDS THE POINTER TO
3577          ;* THE ASCIZ STRING
3578 027342 104414          RESREG      ;RESTORE ALL THE REGISTERS
3579 027344 012616          MOV     (KSP)+,(KSP) ;LEAVE ONLY RETURN ADDR ON STACK
3580 027346 000207          RTS     PC          ;RETURN TO ERROR HANDLER
3581
3582
3583          ;:*****
3584
3585          .SBTTL SAVE AND RESTORE R0-R5 ROUTINES
3586

```

```

3587      ;*SAVE R0-R5
3588      ;*CALL:
3589      ;*   SAVREG
3590      ;*UPON RETURN FROM $$SAVREG THE STACK WILL LOOK LIKE:
3591      ;*
3592      ;*TOP----(+16)
3593      ;* +2----(+18)
3594      ;* +4----R5
3595      ;* +6----R4
3596      ;* +8----R3
3597      ;*+10----R2
3598      ;*+12----R1
3599      ;*+14----R0
3600
3601      027350      $$SAVREG:
3602      027350      010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
3603      027352      010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
3604      027354      010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
3605      027356      010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
3606      027360      010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
3607      027362      010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
3608      027364      016646      000022      MOV      22(SP),-(SP)  ;;SAVE PS OF MAIN FLOW
3609      027370      016646      000022      MOV      22(SP),-(SP)  ;;SAVE PC OF MAIN FLOW
3610      027374      016646      000022      MOV      22(SP),-(SP)  ;;SAVE PS OF CALL
3611      027400      016646      000022      MOV      22(SP),-(SP)  ;;SAVE PC OF CALL
3612      027404      000002      RTI
3613
3614      ;*RESTORE R0-R5
3615      ;*CALL:
3616      ;*   RESREG
3617      027406      $$RESREG:
3618      027406      012666      000022      MOV      (SP)+,22(SP)  ;;RESTORE PC OF CALL
3619      027412      012666      000022      MOV      (SP)+,22(SP)  ;;RESTORE PS OF CALL
3620      027416      012666      000022      MOV      (SP)+,22(SP)  ;;RESTORE PC OF MAIN FLOW
3621      027422      012666      000022      MOV      (SP)+,22(SP)  ;;RESTORE PS OF MAIN FLOW
3622      027426      012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
3623      027430      012604      MOV      (SP)+,R4      ;;POP STACK INTO R4
3624      027432      012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
3625      027434      012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
3626      027436      012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
3627      027440      012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
3628      027442      000002      RTI
3629
3630      ;:*****
3631      .SBTTL  TYPE ROUTINE
3632
3633      ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
3634      ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
3635      ;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
3636      ;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
3637      ;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
3638      ;*
3639      ;*CALL:
3640      ;*1) USING A TRAP INSTRUCTION
3641      ;*   TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
3642      ;*OR
    
```

```

3643      :*      TYPE
3644      :*      MESADR
3645      :*
3646      :*2) USING A JSR INSTRUCTION
3647      :*      MOV      PS,-(SP)      ;;PUSH PROCESSOR STATUS WORD ON THE STACK
3648      :*      JSR      PC,$TYPE      ;;CALL TYPE ROUTINE
3649      :*      MESADDR      ;;FIRST ADDRESS OF MESSAGE
3650
3651      027444 105737 001153      $TYPE:  TSTB      $TPFLG      ;;IS THERE A TERMINAL?
3652      027450 100002      BPL      1$      ;;BR IF YES
3653      027452 000000      HALT      ;;HALT HERE IF NO TERMINAL
3654      027454 000407      BR      ;;LEAVE
3655      027456 010046      1$:      MOV      RO,-(SP)      ;;SAVE RO
3656      027460 017600 000002      MOV      @2(SP),RO      ;;GET ADDRESS OF ASCIZ STRING
3657      027464 112046      2$:      MOVB      (RO)+,-(SP)      ;;PUSH CHARACTER TO BE TYPED ONTO STACK
3658      027466 001005      BNE      4$      ;;BR IF IT ISN'T THE TERMINATOR
3659      027470 005726      TST      (SP)+      ;;IF TERMINATOR POP IT OFF THE STACK
3660      027472 012600      MOV      (SP)+,RO      ;;RESTORE RO
3661      027474 062716 000002      3$:      ADD      #2,(SP)      ;;ADJUST RETURN PC
3662      027500 000002      RTI      ;;RETURN
3663      027502 122716 000011      4$:      CMPB      #HT,(SP)      ;;BRANCH IF <HT>
3664      027506 001424      BEQ      8$
3665      027510 122716 000200      CMPB      #CRLF,(SP)      ;;BRANCH IF NOT
3666      027514 001004      BNE      5$
3667      027516 005726      TST      (SP)+      ;;POP <CR><LF> EQUIV
3668      027520 104400 001217      TYPE      ,SCRLF
3669      027524 000757      BR      2$      ;;GET NEXT CHARACTER
3670      027526 004737 027604      5$:      JSR      PC,$TYPEC      ;;GO TYPE THIS CHARACTER
3671      027532 123726 001152      6$:      CMPB      $FILLC,(SP)+      ;;IS IT TIME FOR FILLER CHARS.?
3672      027536 001352      BNE      2$      ;;IF NO GO GET NEXT CHAR.
3673      027540 013746 001150      MOV      $NULL,-(SP)      ;;GET # OF FILLER CHAR'S. NEEDED
3674      ;;AND THE NULL CHAR.
3675      027544 105366 000001      7$:      DECB      1(SP)      ;;DOES A NULL NEED TO BE TYPED?
3676      027550 002770      BLT      6$      ;;BR IF NO--GO POP THE NULL OFF OF STACK
3677      027552 004737 027604      JSR      PC,$TYPEC      ;;GO TYPE A NULL
3678      027556 000772      BR      7$      ;;LOOP
3679
3680      ;;HORIZONTAL TAB PROCESSOR
3681
3682      027560 112716 000040      8$:      MOVB      #' ,(SP)      ;;REPLACE TAB WITH SPACE
3683      027564 004737 027604      9$:      JSR      PC,$TYPEC      ;;TYPE A SPACE
3684      027570 132737 000007 027650      BITB      #7,$CHARCNT      ;;BRANCH IF NOT AT
3685      027576 001372      BNE      9$      ;;TAB STOP
3686      027600 005726      TST      (SP)+      ;;POP SPACE OFF STACK
3687      027602 000730      BR      2$      ;;GET NEXT CHARACTER
3688      027604 105777 151334      $TYPEC:  TSTB      @2$TPS      ;;WAIT UNTIL PRINTER IS READY
3689      027610 100375      BPL      $TYPEC
3690      027612 116677 000002 151326      MOVB      2(SP),@2$TPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
3691      027620 122766 000015 000002      CMPB      #CR,2(SP)      ;;BRANCH IF
3692      027626 001003      BNE      1$      ;;NOT <CR>
3693      027630 105037 027650      CLRB      $CHARCNT
3694      027634 000406      BR      $TYPEX      ;;EXIT
3695      027636 122766 000012 000002      1$:      CMPB      #LF,2(SP)      ;;BRANCH IF
3696      027644 001402      BEQ      $TYPEX      ;;<LF>
3697      027646 105227      INCB      (PC)+      ;;INC SPACE
3698      027650 000000      $CHARCNT: .WORD 0      ;;COUNT
    
```

```

3699 027652 000207 $TYPEX: RTS PC
3700
3701 ::*****
3702
3703 .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
3704
3705 :*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
3706 :*OCTAL (ASCII) NUMBER AND TYPE IT.
3707 :*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
3708 :*CALL:
3709 :*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
3710 :*      TYPOS    ;;CALL FOR TYPEOUT
3711 :*      .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3712 :*      .BYTE   M              ;;M=1 OR 0
3713 :*                                  ;;1=TYPE LEADING ZEROS
3714 :*                                  ;;0=SUPPRESS LEADING ZEROS
3715
3716 :*$STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
3717 :*$TYPOS OR $TYPOC
3718 :*CALL:
3719 :*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
3720 :*      TYPON    ;;CALL FOR TYPEOUT
3721
3722 :*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
3723 :*CALL:
3724 :*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
3725 :*      TYPOC    ;;CALL FOR TYPEOUT
3726
3727 027654 017646 000000 030077 $TYPOS: MOV @ (SP),-(SP) ;;PICKUP THE MODE
3728 027660 116637 000001 030077 MOVB 1(SP),SOFILL ;;LOAD ZERO FILL SWITCH
3729 027666 112637 030101 MOVB (SP)+,SOMODE+1 ;;NUMBER OF DIGITS TO TYPE
3730 027672 062716 000002 ADD #2,(SP) ;;ADJUST RETURN ADDRESS
3731 027676 000406 BR STYPON
3732 027700 112737 000001 030077 $TYPOC: MOVB #1,SOFILL ;;SET THE ZERO FILL SWITCH
3733 027706 112737 000006 030101 MOVB #6,SOMODE+1 ;;SET FOR SIX(6) DIGITS
3734 027714 112737 000005 030076 $STYPON: MOVB #5,SOCNT ;;SET THE ITERATION COUNT
3735 027722 010346 MOV R3,-(SP) ;;SAVE R3
3736 027724 010446 MOV R4,-(SP) ;;SAVE R4
3737 027726 010546 MOV R5,-(SP) ;;SAVE R5
3738 027730 113704 030101 MOVB SOMODE+1,R4 ;;GET THE NUMBER OF DIGITS TO TYPE
3739 027734 005404 NEG R4
3740 027736 062704 000006 ADD #6,R4 ;;SUBTRACT IT FOR MAX. ALLOWED
3741 027742 110437 030100 MOVB R4,SOMODE ;;SAVE IT FOR USE
3742 027746 113704 0177 MOVB SOFILL,R4 ;;GET THE ZERO FILL SWITCH
3743 027752 016605 MOV 12(SP),R5 ;;PICKUP THE INPUT NUMBER
3744 027756 005003 CLR R3 ;;CLEAR THE OUTPUT WORD
3745 027760 006105 1$: ROL R5 ;;ROTATE MSB INTO 'C'
3746 027762 000404 BR 3$ ;;GO DO MSB
3747 027764 006105 2$: ROL R5 ;;FORM THIS DIGIT
3748 027766 006105 ROL R5
3749 027770 006105 ROL R5
3750 027772 010503 MOV R5,R3
3751 027774 006103 3$: ROL R3 ;;GET LSB OF THIS DIGIT
3752 027776 105337 030100 DECB SOMODE ;;TYPE THIS DIGIT?
3753 030002 100016 BPL 7$ ;;BR IF NO
3754 030004 042703 177770 BIC #177770,R3 ;;GET RID OF JUNK
    
```



```

3755 030010 001002      BNE      4$      ::TEST FOR 0
3756 030012 005704      TST      R4      ::SUPPRESS THIS 0?
3757 030014 001403      BEQ      5$      ::BR IF YES
3758 030016 005204      4$: INC      R4      ::DON'T SUPPRESS ANYMORE 0'S
3759 030020 052703 000060      BIS      #'0,R3  ::MAKE THIS DIGIT ASCII
3760 030024 052703 000040      5$: BIS      #' ,R3  ::MAKE ASCII IF NOT ALREADY
3761 030030 110337 030074      MOV      R3,8$   ::SAVE FOR TYPING
3762 030034 104400 030074      TYPE    ,8$     ::GO TYPE THIS DIGIT
3763 030040 105337 030076      7$: DECB   $OCNT  ::COUNT BY 1
3764 030044 003347      BGT      2$     ::BR IF MORE TO DO
3765 030046 002402      BLT      6$     ::BR IF DONE
3766 030050 005204      INC      R4     ::INSURE LAST DIGIT ISN'T A BLANK
3767 030052 000744      BR       2$     ::GO DO THE LAST DIGIT
3768 030054 012605      6$: MOV     (SP)+,R5  ::RESTORE R5
3769 030056 012604      MOV     (SP)+,R4  ::RESTORE R4
3770 030060 012603      MOV     (SP)+,R3  ::RESTORE R3
3771 030062 016666 000002 000004      MOV     2(SP),4(SP)  ::SET THE STACK FOR RETURNING
3772 030070 012616      MOV     (SP)+,(SP)
3773 030072 000002      RTI      ::RETURN
3774 030074      000      8$: .BYTE  0      ::STORAGE FOR ASCII DIGIT
3775 030075      000      .BYTE  0      ::TERMINATOR FOR TYPE ROUTINE
3776 030076      000      $OCNT: .BYTE 0      ::OCTAL DIGIT COUNTER
3777 030077      000      $OFILL: .BYTE 0     ::ZERO FILL SWITCH
3778 030100 000000      $OMODE: .WORD 0     ::NUMBER OF DIGITS TO TYPE
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3790
3791
3792 030102
3793 030102 010046
3794 030104 010146
3795 030106 010246
3796 030110 010346
3797 030112 010546
3798 030114 012746 020200
3799 030120 016605 000020
3800 030124 100004
3801 030126 005405
3802 030130 112766 000055 000001
3803 030136 005000
3804 030140 012703 030316
3805 030144 112723 000040
3806 030150 005002
3807 030152 016001 030306
3808 030156 160105
3809 030160 002402
3810 030162 005202
    
```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

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

::*CALL:
 ;* MOV NUM,-(SP) ::PUT THE BINARY NUMBER ON THE STACK
 ;* TYPDS ::GO TO THE ROUTINE

\$TYPDS:

```

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

```

3811 030164 000774
3812 030166 060105
3813 030170 005702
3814 030172 001002
3815 030174 105716
3816 030176 100407
3817 030200 106316
3818 030202 103003
3819 030204 116663 000001 177777
3820 030212 052702 000060
3821 030216 052702 000040
3822 030222 110223
3823 030224 005720
3824 030226 020027 000010
3825 030232 002746
3826 030234 003002
3827 030236 010502
3828 030240 000764
3829 030242 105726
3830 030244 100003
3831 030246 116663 177777 177776
3832 030254 105013
3833 030256 012605
3834 030260 012603
3835 030262 012602
3836 030264 012601
3837 030266 012600
3838 030270 104400 030316
3839 030274 016666 000002 000004
3840 030302 012616
3841 030304 000002
3842 030306 023420
3843 030310 001750
3844 030312 000144
3845 030314 000012
3846 030316 000004
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856 030326 010046
3857 030330 016600 000002
3858 030334 005740
3859 030336 111000
3860 030340 016000 030346
3861 030344 000200
3862
3863
3864
3865
3866

4$: BR 3$
ADD R1,R5
TST R2
BNE 5$
TSTB (SP)
BMI 7$
5$: ASLB (SP)
BCC 6$
MOVB 1(SP),-1(R3)
6$: BIS #'0,R2
7$: BIS #' ,R2
MOVB R2,(R3)+
TST (R0)+
CMP R0,#10
BLT 2$
BGT 8$
MOV R5,R2
BR 6$
8$: TSTB (SP)+
BPL 9$
MOVB -1(SP),-2(R3)
9$: CLRB (R3)
MOV (SP)+,R5
MOV (SP)+,R3
MOV (SP)+,R2
MOV (SP)+,R1
MOV (SP)+,R0
TYPE $DBLK
MOV 2(SP),4(SP)
MOV (SP)+,(SP)
RTI
SDTBL: 1000.
100.
100.
10.
$DBLK: .BLKW 4
*****
.SBTTL TRAP DECODER
:*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
:*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
:*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
:*GO TO THAT ROUTINE.
STRAP: MOV R0,-(SP)
MOV 2(SP),R0
TST -(R0)
MOVB (R0),R0
MOV $TRPAD(R0),R0
RTS R0
.SBTTL TRAP TABLE
:*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
    
```

C 7

```

3867 ;*BY THE 'TRAP' INSTRUCTION.
3868
3869 : ROUTINE
3870 :
3871 $TRPAD:
3872 $TYPE ::CALL=TYPE TRAP+0(104400) TTY TYPEOUT ROUTINE
3873 $TYPOC ::CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3874 $TYPOS ::CALL=TYPOS TRAP+4(104404) TYPE OCTAL NUMBER (NO LEADING ZEROS)
3875 $TYPON ::CALL=TYPON TRAP+6(104406) TYPE OCTAL NUMBER (AS PER LAST CALL)
3876 $TYPDS ::CALL=TYPDS TRAP+10(104410) TYPE DECIMAL NUMBER (WITH SIGN)
3877 $SAVREG ::CALL=SAVREG TRAP+12(104412) SAVE R0-R5 ROUTINE
3878 $RESREG ::CALL=RESREG TRAP+14(104414) RESTORE R0-R5 ROUTINE
3879 $TBITOFF ::CALL=TBITO TRAP+16(104416) THIS WILL TURN OFF T BIT TRAPPING
3880 $TBITRESTORE ::CALL=TBITR TRAP+20(104420) THIS WILL RETURN THE T BIT TO PR
3881
3882 ;:*****
3883
3884 .SBTTL POWER DOWN AND UP ROUTINES
3885
3886 :POWER DOWN ROUTINE
3887 030370 012737 030516 000024 $PWDRN: MOV #SILLUP,@#PWVVEC ::SET FOR FAST UP
3888 030376 012737 000340 000026 MOV #340,@#PWVVEC+2 ::PRIO:7
3889 030404 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
3890 030406 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
3891 030410 010246 MOV R2,-(SP) ::PUSH R2 ON STACK
3892 030412 010346 MOV R3,-(SP) ::PUSH R3 ON STACK
3893 030414 010446 MOV R4,-(SP) ::PUSH R4 ON STACK
3894 030416 010546 MOV R5,-(SP) ::PUSH R5 ON STACK
3895 030420 010637 030522 MOV SP,$SAVR6 ::SAVE SP
3896 030424 012737 030436 000024 MOV #SPWRUP,@#PWVVEC ::SET UP VECTOR
3897 030432 000000 HALT
3898 030434 000776 BR .-2 ::HANG UP
3899
3900 :POWER UP ROUTINE
3901 030436 013706 030522 $PWURUP: MOV $SAVR6,SP ::GET SP
3902 030442 005037 030522 CLR $SAVR6 ::WAIT LOOP FOR THE TTY
3903 030446 005237 030522 1$: INC $SAVR6 ::WAIT FOR THE INC
3904 030452 001375 BNE 1$ ::OF WORD
3905 030454 012605 MOV (SP)+,R5 ::POP STACK INTO R5
3906 030456 012604 MOV (SP)+,R4 ::POP STACK INTO R4
3907 030460 012603 MOV (SP)+,R3 ::POP STACK INTO R3
3908 030462 012602 MOV (SP)+,R2 ::POP STACK INTO R2
3909 030464 012601 MOV (SP)+,R1 ::POP STACK INTO R1
3910 030466 012600 MOV (SP)+,R0 ::POP STACK INTO R0
3911 030470 012737 030370 000024 MOV #SPWRDN,@#PWVVEC ::SET UP THE POWER DOWN VECTOR
3912 030476 012737 000340 000026 MOV #340,@#PWVVEC+2 ::PRIO:7
3913 030504 104400 TYPE ::REPORT THE POWER FAILURE
3914 030506 030524 $PWMSG: .WORD PWMSG ::POWER FAIL MESSAGE POINTER
3915 030510 012716 MOV (PC)+,(SP) ::RESTART AT START
3916 030512 040076 $PWRAD: .WORD START ::RESTART ADDRESS
3917 030514 000002 RTI
3918 030516 000000 $SILLUP: HALT ::THE POWER UP SEQUENCE WAS STARTED
3919 030520 000776 BR .-2 :: BEFORE THE POWER DOWN WAS COMPLETE
3920 030522 000000 $SAVR6: 0 ::PUT THE SP HERE
3921 030524 006412 047520 042527 PWMSG: .ASCIZ <12><15>?POWER FAILURE, RESTARTING PROGRAM?
3922 030532 020122 040506 046111
    
```

3923 030540 051125 026105 051040
 3924 030546 051505 040524 052122
 3925 030554 047111 020107 051120
 3926 030562 043517 040522 000'15
 3927
 3928
 3929
 3930
 3931
 3932
 3933
 3934
 3935
 3936
 3937
 3938
 3939
 3940
 3941
 3942
 3943
 3944
 3945
 3946
 3947
 3948
 3949
 3950 030570 010046
 3951 030572 010146
 3952 030574 010246
 3953 030576 010346
 3954 030600 013746 000004
 3955 030604 013746 000006
 3956 030610 013746 000114
 3957 030614 013746 000116
 3958 030620 010600
 3959 030622 013737 177776 000006
 3960 030630 012701 003776
 3961 030634 105727
 3962 030636 000200
 3963 030640 100060
 3964 030642 012737 030774 000004
 3965 030650 005737 177572
 3966 030654 052737 100000 030636
 3967 030662 005046
 3968 030664 012702 172340
 3969 030670 012703 000010
 3970 030674 012762 077406 177740 1\$:
 3971 030702 011622
 3972 030704 062716 000200
 3973 030710 077307
 3974 030712 012742 177600
 3975 030716 005042
 3976 030720 012737 000020 172516
 3977 030726 005237 177572
 3978 030732 012737 030764 000004

```

;:*****
.SBTTL ROUTINE TO SIZE MEMORY

;*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
;*$LSTBK WILL CONTAIN THE LAST BANK AS A SAF
;*$KT11 IS THE MEMORY MANAGEMENT KEY
;*BIT07 = 0 DON'T USE MEMORY MANAGEMENT
;* MUST BE SETUP BEFORE THE CALL
;*BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
;* DETERMINED BY ROUTINE
;* --NOTE--
;*THIS ROUTINE SUPPORTS PDP 11/74.
;*IF ACTUAL MEMORY IS LESS THAN THAT INDICATED BY SIZE REGISTER
;*AND A REFERENCE IS MADE TO A MEMORY ADDRESS THAT IS GREATER THAN
;*ACTUAL MEMORY BUT LESS THAN SIZE REGISTER (INDICATED), THEN A
;*MEMORY REFERENCE TIMEOUT TO VECTOR 114 WILL OCCUR.

$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 @114,-(SP) ;;SAVE PRESENT PARITY VECTOR PS & PC
        MOV @116,-(SP)
        MOV SP,R0 ;;SAVE THE STACK POINTER
        MOV @PS,@ERRVEC+2 ;;SET ERRVEC PS TO PRESENT PS
        MOV #3776,R1 ;;SETUP ADDRESS
        TSTB (PC)+ ;;USE MEMORY MANAGEMENT?
$KT11: .WORD 200 ;;SET TO USE MEMORY MANAGEMENT
        BPL SCORE ;;BR IF NO
        MOV #SKTNEX,@ERRVEC ;;SET FOR TIMEOUT
        TST @SR0 ;;KT11 ARE YOU THERE?
        BIS #100000,$KT11 ;;YES--SET KT11 KEY
        CLR -(SP) ;;INITIALIZE FOR 'PAR' LOADING
        MOV #KIPAR0,R2 ;;ADDRESS OF FIRST 'PAR'
        MOV #^D8,R3 ;;LOAD EIGHT 'PAR.'S' AND EIGHT 'PDR.'S'
        MOV #77406,-4(R2) ;;PDR = 4K, UP, READ/WRITE
        MOV (SP),(R2)+ ;;LOAD 'PAR'
        ADD #200,(SP) ;;UPDATE FOR NEXT 'PAR'
        SOB R3,1$ ;;LOOP UNTIL ALL EIGHT ARE LOADED
        MOV #177600,-(R2) ;;SETUP KIPAR7 FOR I/O
        CLR -(R2) ;;SETUP KIPAR6 FOR TESTING
        MOV #20,@SR3 ;;ENABLE 22-BIT ADDRESSING
        INC @SR0 ;;TURN ON MEMORY MANAGEMENT
        MOV #SKTOUT,@ERRVEC ;;SET FOR TIME OUT
    
```



```

4035 031152 012705 031263      MOV    #SOCTVL+13.,R5    ;; POINTER TO DATA TABLE
4036 031156 012704 000014      MOV    #12.,R4         ;; DO ELEVEN CHARACTERS
4037 031162 012703 177770      MOV    #^C7,R3        ;; MASK
4038 031166 012100              MOV    (R1)+,R0       ;; LOWER WORD
4039 031170 012101              MOV    (R1)+,R1       ;; HIGH WORD
4040 031172 005002              CLR    R2             ;; TERMINATOR
4041 031174 110245      1$:  MOV    R2,-(R5)       ;; PUT CHARACTER IN DATA TABLE
4042 031176 010002              MOV    R0,R2         ;; GET THIS DIGIT
4043 031200 005304              DEC    R4            ;; COUNT THIS CHARACTER
4044 031202 003007              BGT    3$           ;; BR IF NOT THE LAST DIGIT
4045 031204 001405              BEQ    2$           ;; BR IF IT IS THE LAST DIGIT
4046 031206 005205              INC    R5            ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
4047 031210 010566 000002      MOV    R5,2(SP)      ;; ASCII CHAR. & PUT IT ON THE STACK
4048 031214 104414              RESREG              ;; RESTORE ALL REGISTERS
4049 031216 000207              RTS    PC            ;; RETURN TO USER
4050 031220 006203      2$:  ASR    R3            ;; POSITION THE MASK FOR THE LAST DIGIT
4051 031222 006001      3$:  ROR    R1            ;; POSITION THE BINARY NUMBER FOR
4052 031224 006000              ROR    R0            ;;
4053 031226 006001              ROR    R1            ;; THE NEXT OCTAL DIGIT
4054 031230 006000              ROR    R0
4055 031232 006001              ROR    R1
4056 031234 006000              ROR    R0
4057 031236 040302              BIC    R3,R2         ;; MASK OUT ALL JUNK
4058 031240 062702 000060      ADD    #^0,R2        ;; MAKE THIS CHAR. ASCII
4059 031244 000753              BR     1$           ;; GO PUT IT IN THE DATA TABLE
4060 031246 000016      SOCTVL: .BLKB 14.    ;; RESERVE DATA TABLE

```

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

.SBTTL TURN OFF AND SAVE T-BIT

;*
;* THIS SUBROUTINE IS REACHED BY THE TRAP CALL 'TBITO', IT IS
;* USED TO TURN OFF THE T-BIT IF IT IS ON. THE PROCESSOR STATUS
;* IS SAVED IN 'OLDPSW' SO THAT THE T-BIT CAN BE RESTORED TO ITS
;* PREVIOUS STATUS WHEN CONDITIONS WARRANT.
;*

```

4074              .EQUIV BIT4,TBIT    ;T-BIT IS BIT04 IN PROC. STATUS
4075 031264      TBITOFF: BIT    #TBIT,2(KSP)    ;IS THE T-BIT ON?
4076 031264 032766 000020 000002  BEQ    1$           ;BRANCH TO EXIT IF IT IS NOT ON
4077 031272 001406              MOV    2(KSP),OLDPSW ;SAVE OLD PSW FOR RESTORING T BIT
4078 031274 016637 000002 001312  BIC    #TBIT,2(KSP)  ;CLEAR T BIT IF IT IS ON
4079 031302 042766 000020 000002  1$:  RTT           ;RETURN TO PROGRAM
4080 031310 000006

```

.SBTTL RESTORE T-BIT TO ITS PREVIOUS CONDITION

;*
;* THIS SUBROUTINE CAN BE REACHED BY THE TRAP CALL 'TBITR', IT IS
;* USED TO RESTORE THE T-BIT AFTER A PARTICULAR TEST THAT CANNOT
;* BE RUN WITH THE T-BIT ON. IT USES THE PROCESSOR STATUS STORED
;* IN 'OLDPSW' BY 'TBITO', REPLACES THE PS ON THE STACK WITH IT
;* AND DOES AN 'RTT'.
;*

4090

```
4091
4092 031312
4093 031312 013766 001312 000002
4094 031320 042737 000020 001312
4095
4096 031326 000006
4097
4098
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116 031330 010046
4117 031332 012700 000020
4118 031336 005025
4119 031340 077002
4120 031342 012600
4121 031344 000207
4122
4123
4124
4125
4126
4127
4128
4129
4130
4131
4132
4133 031346
4134 031346 005037 001266
4135 031352 005037 001276
4136 031356 005037 001272
4137 031362 012700 177777
4138 031366 010037 001270
4139 031372 010037 001300
4140 031376 010037 001274
4141 031402 000207
4142
4143
4144
4145
4146

;*
TBITRESTORE:
MOV OLDPSW,2(KSP) ;PUT OLD PSW ON STACK
BIC #TBIT,OLDPSW ;CLEAR T-BIT IN 'OLDPSW' SO THAT
;IT WON'T BE TURNED ON BY ACCIDENT
RTT ;RETURN TO PROGRAM AND INHIBIT
;T BIT TRAP AFTER THIS INSTRUCTION.

.SBTTL CLEAR 16 PARS OR PDRS STARTING FROM ADDRESS IN R5
;*****
SUBROUTINE CLRREG
;
; THIS SUBROUTINE CLEARS 16 CONSECUTIVE MEMORY MANAGEMENT
; REGISTERS STARTING WITH THE REGISTER POINTED TO BY R5.
; IT SAVES R0 ON THE STACK AND LOADS A COUNT IN R0,
; CLEARS THE PAR'S OR PDR'S, AND THEN RESTORES R0 BEFORE
; RETURNING.
; THE CALLING SEQUENCE IS:
; MOV #KIPAR0,R5 ;PUT ADDRESS OF FIRST KERNEL PAR IN R5
; JSR PC,CLRREG ;GO CLEAR ALL KERNEL PAR'S
;*****
CLRREG: MOV R0,-(KSP) ;SAVE R0 ON STACK
MOV #20,R0 ;PUT COUNT IN R0
1$: CLR (R5)+ ;CLEAR PAR ORPDR POINTED TO BY R5
SOB R0,1$ ;BRANCH BACK 15 DECIMAL TIMES.
MOV (KSP)+,R0 ;RESTORE R0 FROM STACK
RTS PC ;RETURN TO TEST

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

.SBTTL P.A.R. OR P.D.R. ADDRESS TIMED OUT WHEN REFERENCED
;*****
;
; THIS SUBROUTINE IS USED TO LOG AND REPORT THE FACT THAT A
```

4147
4148
4149
4150
4151 031404
4152 031404 005227
4153 031406 177777
4154 031410 001401
4155 031412 000000
4156
4157
4158
4159
4160 031414 012637 001306
4161 031420 012637 001310
4162 031424 013737 177766 001260
4163 031432 013737 001260 177766
4164 031440 023737 001260 001224
4165 031446 001402
4166 031450 104001
4167 031452 000414
4168 031454 050037 001276
4169 031460 005100
4170 031462 040037 001300
4171 031466 005100
4172 031470 105737 001103
4173 031474 001002
4174 031476 104201
4175 031500 000401
4176 031502 104301
4177 031504 012737 177777 031406
4178 031512 013746 001310
4179 031516 013746 001306
4180 031522 000006
4181
4182
4183
4184
4185
4186
4187
4188
4189
4190
4191
4192 031524
4193 031524 012637 001306
4194 031530 050037 001276
4195 031534 005100
4196 031536 040037 001300
4197 031542 005100
4198 031544 050137 001266
4199 031550 005101
4200 031552 040137 001270
4201 031556 005101
4202 031560 105737 001103

```

: * REFERENCE TO A P.A.R. OR A P.D.R. TIMED OUT ON THE UNIBUS. IT
: * KEEPS A LOGICAL AND AND A LOGICAL OR OF EACH ADDRESS THAT TIMES
: * OUT.
: *****
TIMEOUT: ; STARTING ADDRESS OF SUBROUTINE
          INC (PC)+ ; INCREMENT ONE TIME GATE
TOFLAG: .WORD -1 ; ONE TIME ENTRANCE FLAG
         BEQ 10$ ; BRANCH IF FLAG IS NOW ZERO
         HALT ; I HAVE ENTERED THIS ROUTINE BEFORE
          ; I FINISHED REPORTING THE FIRST ERROR
          ; THE SECOND ENTRY ADDRESS IS ON THE
          ; STACK AND THE FIRST ERROR CONDITION
          ; IS PROBABLY STILL LOCKED UP .
10$: MOV (KSP)+,OLDPC ; SAVE RETURN ADDRESS
     MOV (KSP)+,OLDPS ; SAVE OLD PSW
     MOV @#CPUERR,PCPUER ; SAVE CPU ERROR REGISTER
     MOV PCPUER,@#CPUERR ; CLEAR CPU ERROR REGISTER
     CMP PCPUER,CPUEXP ; SEE IF EXPECTED CONDITION CAME UP.
     BEQ 1$ ; BRANCH IF IT WAS A TIMEOUT
     ERROR 1 ; NOT RIGHT CONDITION
     BR 3$ ; BRANCH TO EXIT
1$: BIS RO,ADDROR ; PERFORM LOGICAL OR OF FAILING ADDRESS
     COM RO ; GET RO READY FOR AND
     BIC RO,ADRAND ; PERFORM LOGICAL AND
     COM RO ; PUT RO BACK AS IT WAS
     TSTB $ERFLG ; IS HIS THE FIRST ERROR
     BNE 2$ ; BRANCH IF NOT FIRST ERROR
     ERROR 201 ; NO REGISTER RESPONSE.
     BR 3$ ; BRANCH TO EXIT
2$: ERROR 301 ; CONTINUE NO RESPONSE TABLE
3$: MOV #-1,TOFLAG ; RESET ONE TIME GATE
     MOV OLDPS,-(KSP) ; RESTORE OLD PSW
     MOV OLDPC,-(KSP) ; PUSH RETURN ADDRESS BACK ON THE STACK
     RTT ; RETURN TO THE TEST

```

```

.SBTTL DUAL ADDRESSING WHEN LOADING A P.A.R. OR P.D.R.
: *****
: *
: * THIS SUBROUTINE WILL LOG AND REPORT ALL DUAL ADDRESSING ERRORS
: * FOUND IN BOTH P.A.R.'S AND P.D.R.'S. A 'LOGICAL OR' AND A
: * 'LOGICAL AND' OF THE WRITTEN ADDRESSES WILL BE MAINTAINED IN
: * 'ADDROR' AND 'ADRAND'. THE LOG ON THE ADDITIONAL OR FAILING,
: * ADDRESSES WILL BE MAINTAINED IN 'DATAOR' AND 'DATAND'.
: *****

```

```

DUALADR: MOV (KSP)+,OLDPC ; STARTING LOCATION OF SUBROUTINE
          ; SAVE RETURN ADDRESS IN CASE OF LOOP
        BIS RO,ADDROR ; LOGICAL OR OF WRITTEN ADDRESS
        COM RO ; GET RO READY FOR AND
        BIC RO,ADRAND ; PERFORM LOGICAL AND
        COM RO ; PUT RO BACK AS IT WAS
        BIS R1,DATAOR ; LOGICAL OR OF DUALED ADDRESS
        COM R1 ; GET R1 READY FOR AND
        BIC R1,DATAND ; PERFORM LOGICAL AND
        COM R1 ; PUT R1 BACK AS IT WAS
        TSTB $ERFLG ; SEE IF THIS IS FIRST ERROR

```



```

4203 031564 001002          BNE      1$          ;BRANCH IF NOT FIRST ERROR
4204 031566 104202          ERROR   202
4205 031570 000401          BR      2$          ;BRANCH TO EXIT
4206 031572 104302          1$:     ERROR   302
4207 031574 000177 147506 2$:     JMP      @OLDPC    ;RETURN TO TEST
4208
4209

```

.SBTTL COUNT PATTERN ERRORS IN P.A.R.'S OR P.D.R.'S
 :*****

```

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

```

```

4221
4222 031600          PARCOUNT:
4223 031600 012637 001306      MOV      (KSP)+,OLDPC ;STARTING ADDRESS OF THIS SUBROUTINE
4224 031604 050037 001276      BIS      R0,ADDROR   ;SAVE RETURN ADDRESS IN CASE OF LOOP
4225 031610 005100          COM      R0          ;LOGICAL OR OF FAILING ADDRESS
4226 031612 040037 001300      BIC      R0,ADRAND   ;GET R0 READY FOR AND
4227 031616 005100          COM      R0          ;PERFORM LOGICAL AND
4228 031620 050137 001272      BIS      R1,PATTOR   ;PUT R0 BACK AS IT WAS
4229 031624 005101          COM      R1          ;LOGICAL OR OF PATTERN LOADED
4230 031626 040137 001274      BIC      R1,PATAND   ;GET R1 READY FOR AND
4231 031632 005101          COM      R1          ;PERFORM LOGICAL AND
4232 031634 050237 001266      BIS      R2,DATAOR   ;PUT R1 BACK AS IT WAS
4233 031640 005102          COM      R2          ;LOGICAL OR OF DATA FETCHED
4234 031642 040237 001270      BIC      R2,DATAND   ;GET R2 READY FOR AND
4235 031646 005102          COM      R2          ;PERFORM LOGICAL AND
                        ;PUT R2 BACK AS IT WAS

```

```

4236 031650 105737 001103          TSTB  $ERFLG      ;SEE IF THIS IS THE FIRST ERROR
4237 031654 001002          BNE   1$         ;BRANCH IF NOT FIRST ERROR
4238 031656 104203          ERROR 203
4239 031660 000401          BR    2$         ;BRANCH TO EXIT
4240 031662 104303          1$:  ERROR 303
4241 031664 000177 147416      2$:  JMP    @OLDPC ;RETURN TO TEST
4242
4243
4244          .SBTTL ***** TRAP HANDLING ROUTINES *****
4245
4246          .SBTTL CPU TRAP HANDLER ROUTINE
4247          :*****
4248          :*
4249          :* THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS, THRU
4250          :* 'ERRVEC' (000004). IF THIS SUBROUTINE IS ENTERED BY A SECOND
4251          :* TRAP BEFORE THE FIRST HAS BEEN PROCESSED A HALT IS EXECUTED.
4252          :* IF THE WORD 'CPUEXP' IS ZERO, NO TRAP WAS EXPECTED AND AN
4253          :* UNEXPECTED ERROR MESSAGE IS GIVEN. IF THE WORD 'CPUEXP' IS
4254          :* NOT ZERO THEN THE CPU ERROR REGISTER 'CPUERR' IS COMPARED WITH
4255          :* 'CPUEXP' TO SEE IF THE PROPER CONDITION OCCURRED. 'PCPUER' CAN
4256          :* BE USED AS A FLAG TO INDICATE THAT A TRAP HAS OCCURRED SINCE IT
4257          :* IS LOADED WITH THE ERROR REGISTER IF A TRAP VECTORS HERE
4258          :*
4259          :*****
4260 031670 005227          CPUER: INC      (PC)+ ;MAKE FLAG ZERO IF FIRST TIME
4261 031672 177777          CPFLAG: .WORD -1    ;NEGATIVE ONE FOR A FLAG
4262 031674 001401          BEQ   10$        ;BRANCH IF FIRST TIME IN
4263 031676 000000          HALT           ;I HAVE ENTERED THIS ROUTINE BEFORE
4264          :* ;I FINISHED REPORTING THE FIRST ERROR
4265          :* ;THE SECOND ENTRY ADDRESS IS ON THE
4266          :* ;STACK AND THE FIRST ERROR CONDITION
4267          :* ;IS PROBABLY STILL LOCKED UP
4268 031700 012637 001306      10$:  MOV    (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
4269 031704 012637 001310      MOV    (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP
4270 031710 013737 177766 001260  MOV    CPUERR,PCPUER ;SAVE CPU ERROR REGISTER
4271 031716 013737 001306 001262  MOV    OLDPC,BADPC  ;SAVE PC+2 AT TIME OF ABORT
4272 031724 005737 001224      TST   CPUEXP      ;SEE IF ANY CONDITION WAS EXPECTED
4273 031730 001406          BEQ   2$         ;BRANCH IF NO TRAP WAS EXPECTED
4274 031732 023737 001260 001224  CMP   PCPUER,CPUEXP ;SEE IF EXPECTED ERROR OCCURED
4275 031740 001403          BEQ   1$         ;BRANCH IF ERROR CODES MATCH
4276 031742 104001          ERROR 1         ;ERROR TYPE OUT ITEM 1
4277 031744 000401          BR    1$         ;SKIP NEXT INSTRUCTION
4278 031746 104002          2$:  ERROR 2      ;ERROR ITEM 2 NO CPU TRAP EXPECTED
4279 031750 005037 177766      1$:  CLR   CPUERR     ;CLEAR CPU ERROR REGISTER
4280 031754 012737 177777 031672  MOV   #-1,CPFLAG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
4281 031762 013746 001310      MOV   OLDPS,-(KSP) ;PUSH OLD PSW BACK ON STACK
4282 031766 013746 001306      MOV   OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON STACK
4283 031772 000006          RTT           ;RETURN FROM INTERRUPT OR ABORT
4284
4285
4286
4287          .SBTTL CACHE TRAPS AND ABORTS HANDLER ROUTINE
4288          :*****
4289          :*
4290          :* THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED PARITY ERRORS. IF
4291          :* THE PARITY ERROR IS AN ABORT THE TEST THAT WAS RUNNING WILL
    
```

4292 :* BE RESTARTED ONCE AFTER THE ABORT CONDITION IS REPORTED. ON THE
4293 :* SECOND ABORT IN A SINGLE TEST THE NEXT TEST IS ATTEMPTED
4294 :* AFTER THE ABORT IS REPORTED.
4295 :* IF THE PARITY ERROR IS A TRAP THE CACHE WILL BE CLEANED UP BY
4296 :* REMOVING THE BAD WORD THAT MAY BE IN THE CACHE, AND THE TEST
4297 :* WILL BE CONTINUED AFTER THE PARITY ERROR IS REPORTED.
4298 :*
4299 :*

```
*****  
4300 031774 005227 MEMER: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME  
4301 031776 177777 PAFLAG: .WORD -1 ;NEGATIVE ONE FOR A FLAG  
4302 032000 001401 BEQ 10$ ;BRANCH IF FIRST TIME IN  
4303 032002 000000 HALT ;I HAVE ENTERED THIS ROUTINE BEFORE  
4304 ;I FINISHED REPORTING THE FIRST ERROR  
4305 ;THE SECOND ENTRY ADDRESS IS ON THE  
4306 ;STACK AND THE FIRST ERROR CONDITION  
4307 ;IS PROBABLY STILL LOCKED UP  
4308 032004 012737 032224 000114 10$: MOV #5$,CACHVEC ;SET CACHE VECTOR TO RTI UNTIL THE  
4309 ;THE CACHE HAS BEEN FIXED.  
4310 032012 012637 001306 MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP  
4311 032016 012637 001310 MOV (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP  
4312 032022 013737 177740 001234 MOV @#LOADRS,PLOADR ;SAVE LOWER CACHE ADDR REG  
4313 032030 013737 177742 001236 MOV @#HIADRS,PHIADR ;SAVE HIGH BITS OF FAILING ADDR  
4314 032036 013737 177744 001240 MOV @#MEMERR,PPARER ;SAVE MEMORY ERROR REGISTER  
4315 032044 013737 177746 001242 MOV @#CONTRL,PCONTR ;SAVE CONTROL REGISTER FOR TYPE OUT  
4316 032052 013737 177750 001244 MOV @#MAINT,PMaint ;SAVE MAINTENANCE REGISTER  
4317 032060 013737 177752 001246 MOV @#HITMIS,PHITMJ ;SAVE HIT/MISS REGISTER  
4318 032066 013737 001306 001262 MOV OLDPC,BADPC ;SAVE PC+2 AT TIME OF ABORT  
4319 032074 032737 000014 001240 BIT #14,PPARER ;WAS THIS A MAIN MEMORY REFERENCE  
4320 032102 001005 BNE 1$ ;BRANCH IF IT WAS A MAIN MEMORY ERROR  
4321 032104 012737 031774 000114 MOV #MEMER,CACHVEC ;LOAD ADDRESS OF THIS ROUTINE IN VECTOR  
4322 032112 104003 ERROR 3 ;UNEXPECTED CACHE PARITY ERROR  
4323 032114 000415 BR 2$ ;BRANCH TO EXIT POINT  
4324 032116 010046 1$: MOV RO,-(KSP) ;SAVE RO ON STACK  
4325 032120 013700 001234 MOV PLOADR,RO ;PUT LOW 16 BITS OF BAD ADDR IN RO  
4326 032124 042700 176000 BIC #176000,RO ;CLEAR UPPER 6 BITS OF ADDRESS  
4327 032130 074027 000002 XOR RO,#BIT1 ;REFERENCE OTHER WORD OF PAIR  
4328 032134 005710 TST (RO) ;READ AT SAME INDEX AS BAD WORD  
4329 032136 012600 MOV (KSP)+,RO ;RESTORE RO FROM STACK  
4330 032140 012737 031774 000114 MOV #MEMER,CACHVEC ;LOAD ADDRESS OF THIS ROUTINE IN VECTOR  
4331 032146 104004 ERROR 4 ;UNEXPECTED MAIN MEMORY PARITY ERROR  
4332 032150 005737 001314 2$: TST RETRY ;ARE YOU RETRYING THIS TEST?  
4333 032154 001006 BNE 3$ ;BRANCH IF THIS IS SECOND TRY  
4334 032156 005237 001314 INC RETRY ;SET RETRY FLAG  
4335 032162 013737 001110 001306 MOV $LPADR,OLDPC ;RETURN TO START OF THIS TEST  
4336 032170 000403 BR 4$ ;BRANCH TO EXIT  
4337 032172 013737 001316 001306 3$: MOV NXTTST,OLDPC ;RETURN TO START OF NEXT TEST  
4338 ;SINCE THIS IS THE SECOND ABORT  
4339 032200 013737 001240 177744 4$: MOV PPARER,MEMERR ;CLEAR MEMORY ERROR REGISTER  
4340 032206 012737 177777 031776 MOV #-1,PAFLAG ;RESTORE A NEGATIVE ONE FOR NEXT TIME  
4341 032214 013746 001310 MOV OLDPS,-(KSP) ;PUSH OLD PSW BACK ON STACK  
4342 032220 013746 001306 MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON STACK  
4343 032224 000006 5$: RTT ;RETURN FROM INTERRUPT.  
4344  
4345  
4346  
4347
```

.SBTTL MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE

4348
4349
4350
4351
4352
4353
4354
4355
4356
4357
4358
4359
4360 032226 005227
4361 032230 177777
4362 032232 001401
4363 032234 000000
4364
4365
4366
4367
4368 032236 011637 001262
4369 032242 012637 001306
4370 032246 012637 001310
4371 032252 013737 177572 001250
4372 032260 013737 177574 001252
4373 032266 013737 177576 001254
4374 032274 005737 001226
4375 032300 001002
4376 032302 104005
4377 032304 000405
4378 032306 023737 001226 001250
4379 032314 001401
4380 032316 104006
4381 032320 042737 177376 177572
4382 032326 012737 177777 032230
4383 032334 013746 001310
4384 032340 013746 001306
4385 032344 000006
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4400
4401
4402
4403 032346

```
*****  
* THIS SUBROUTINE WILL HANDLE MOST OF THE MEMORY MANAGEMENT TRAPS  
* AND ABORTS THAT ARE GENERATED DURING THIS PROGRAM. ANY M.M.  
* ABORTS OR TRAPS THAT OCCUR WHILE 'MMEXP' IS ZERO ARE UNEXPECTED  
* AND WILL BE REPORTED AS SUCH. THIS MAY OCCUR BECAUSE SOME LOGIC  
* THAT IS TO INHIBIT A TRAP HAS FAILED.  
* THERE ARE ALSO TIMES WHEN I AM EXPECTING A CERTAIN CONDITION TO  
* OCCUR, AND WHEN THIS CONDITION IN 'MMEXP' DOES NOT MATCH THE  
* CONTENTS OF 'PMMRO' (SAME AS 'MMRO') AN ERROR IS REPORTED.  
*****  
MMTRAP: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME  
MMFLAG: .WORD -1 ;FLAG SHOULD BE NEG ONE  
BEQ 10$ ;BRANCH IF FIRST TIME INTO ROUTINE  
HALT ;I HAVE ENTERED THIS ROUTINE BEFORE  
 ;I FINISHED REPORTING THE FIRST ERROR  
 ;THE SECOND ENTRY ADDRESS IS ON THE  
 ;STACK AND THE FIRST ERROR CONDITION  
 ;IS PROBABLY STILL LOCKED UP  
10$: MOV (KSP),BADPC ;SAVE PC AT TIME OF ABORT OR TRAP  
MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP  
MOV (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP  
MOV PMRO,PMPRO ;SAVE STATUS REGISTER  
MOV PMR1,PMPR1 ;SAVE AUTO INC/DEC REGISTER  
MOV PMR2,PMPR2 ;SAVE VIRTUAL ADDRESS REGISTER  
TST MMEXP ;SEE IF ANY M.M. TRAPS WERE EXPECTED  
BNE 5$ ;BRANCH IF ONE WAS EXPECTED  
ERROR 5 ;UNEXPECTED M.M. ABORT OR TRAP  
BR 1$ ;BRANCH TO EXIT  
5$: CMP MMEXP,PMMRO ;SEE IF ABORT OR TRAP WAS EXPECTED  
BEQ 1$ ;BRANCH IF CONDITION CORRECT  
ERROR 6 ;ERROR TYPE OUT ITEM 6  
1$: BIC #177376,&PMMRO ;CLEAR ALL BITS EXCEPT 0 AND 8  
MOV #-1,MMFLAG ;RESTORE A NEGATIVE ONE TO FLAG  
MOV OLDPS,-(KSP) ;PUSH OLD PSW ONTO STACK  
MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS ON STACK  
RTT ;RETURN TO MAIN PROGRAM
```

```
.SBTTL D-SPACE TESTS MEMORY MANAGEMENT ABORT SERVICE ROUTINE  
*****  
* THIS ROUTINE WILL BE ENTERED IF A MEMORY MANAGEMENT ABORT  
* OCCURS DURING THE D-SPACE ENABLE TESTS. IF THE ABORT IS A  
* NON-RESIDENT ABORT THE PROBLEM IS PROBABLY IN THE D-SPACE  
* ENABLE LOGIC ON PAGES 'SAPK' AND 'SSRB' IN THE PRINTS.  
* IN ALL OF THE D-SPACE ENABLE TESTS, D-SPACE PAGES 2 & 3  
* ARE MAPPED NON-RESIDENT AND I-SPACE PAGE 4 IS MAPPED NON-  
* RESIDENT. ALL OTHER PAGES ARE MAPPED RESIDENT, 4K, READ/WRITE.  
* THEREFORE IF THE N.R. PAGE IS 2 OR 3 YOU ARE PROBABLY NOT  
* FORCING I-SPACE WHEN YOU SHOULD. BUT IF THE N.R. PAGE IS 4  
* YOU ARE PROBABLY FORCING I-SPACE WHEN YOU SHOULD ALLOW D-SPACE  
*****  
NODSPAC: ;STARTING ADDRESS FOR ABORT SERVICE
```

```

4404 032346 005227
4405 032350 177777
4406 032352 001401
4407 032354 000000
4408
4409
4410
4411
4412 032356 011637 001262
4413 032362 012637 001306
4414 032366 012637 001310
4415 032372 013737 177572 001250
4416 032400 013737 177574 001252
4417 032406 013737 177576 001254
4418 032414 005737 001250
4419 032420 100002
4420 032422 104132
4421 032424 000401
4422 032426 104005
4423 032430 042737 177376 177572
4424 032436 012737 177777 032350
4425 032444 013746 001310
4426 032450 013746 001306
4427 032454 000006
4428
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4440
4441
4442
4443
4444
4445 032456 013700 177776
4446 032462 012637 001306
4447 032466 012637 001310
4448 032472 122700 000340
4449 032476 001401
4450 032500 104122
4451 032502 000137 032546
4452
4453 032506 012637 001306
4454 032512 012637 001310
4455 032516 013700 177776
4456 032522 104123
4457 032524 000137 032546
4458
4459 032530 012637 001306
    
```

```

NDFLAG: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME
          .WORD -1 ;FLAG SHOULD BE NEG ONE
          BEQ 10$ ;BRANCH IF FIRST TIME INTO ROUTINE
          HALT ;I HAVE ENTERED THIS ROUTINE BEFORE
           ;I FINISHED REPORTING THE FIRST ERROR
           ;THE SECOND ENTRY ADDRESS IS ON THE
           ;STACK AND THE FIRST ERROR CONDITION
           ;IS PROBABLY STILL LOCKED UP
10$: MOV (KSP),BADPC ;SAVE PC AT TIME OF ABORT OR TRAP
     MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF LOOP
     MOV (KSP)+,OLDPS ;SAVE OLD PSW IN CASE OF LOOP
     MOV MMRO,MMRO ;SAVE STATUS REGISTER
     MOV MMR1,MMR1 ;SAVE AUTO INC/DEC REGISTER
     MOV MMR2,MMR2 ;SAVE VIRTUAL ADDRESS REGISTER
     TST MMRO ;WAS ABORT NON-RESIDENT?
     BPL 1$ ;BRANCH IF NOT, IT IS UNEXPECTED.
     ERROR 132 ;D-SPACE ENABLE FAULTY
     BR 2$ ;BRANCH TO EXIT
1$: ERROR 5 ;UNEXPECTED M.M. ABORT
2$: BIC #177376,MMRO ;CLEAR ALL BITS EXCEPT 0 AND 8
     MOV #-1,NDFLAG ;RESTORE A NEGATIVE ONE TO FLAG
     MOV OLDPS,-(KSP) ;PUSH OLD PSW ONTO STACK
     MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS ON STACK
     RTT ;RETURN TO MAIN PROGRAM
    
```

```

.SBTTL TRAP ROUTINES FOR ABORT IN SUPERVISOR OR USER MODE
*****
*
* THESE NEXT THREE SUBROUTINES ARE USED FOR THE TESTS THAT VERIFY
* THE VECTOR IS PICKED UP FROM KERNEL SPACE DURING AN ABORT.
* 'KERVEC' IS WHERE I SHOULD GO SINCE IT IS AT 250 WHICH IS
*   KERNEL SPACE VIRTUAL 250.
* 'SUPVEC' IS AT 350 WHICH IS SUPERVISOR VIRTUAL 250
* 'USEVEC' IS AT 450 WHICH IS USER SPACE VIRTUAL 250
*
* THEY ALL READ THE PROCESSOR STATUS WHICH IS DIFFERENT FOR EACH
* VECTOR AND THEN JUMP TO 'RDMMRO'. 'RDMMRO' READS MMRO, MMR1,
* AND MMR2 AND RETURN TO THE TEST WHERE THEY ARE TESTED.
*****
    
```

```

KERVEC: MOV PSW,R0 ;PUT PROCESSOR STATUS INTO R0
        MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS
        MOV (KSP)+,OLDPS ;SAVE OLD PROCESSOR STATUS
        CMPB #340,R0 ;SEE IF CORRECT PSW WAS PICKED UP
        BEQ 1$ ;BRANCH IF PSW IS CORRECT
        ERROR 122 ;WRONG PSW PICKED UP
1$: JMP RDMMRO ;JUMP TO READ MMRO

SUPVEC: MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS
        MOV (KSP)+,OLDPS ;SAVE OLD PROCESSOR STATUS
        MOV PSW,R0 ;READ PSW FOR ERROR PRINT OUT
        ERROR 123 ;WRONG VECTOR, POSSIBLE WRONG PSW
        JMP RDMMRO ;GO READ MMRO

USEVEC: MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS
    
```

```

4460 032534 012637 001310      MOV      (KSP)+,OLDPS      ;SAVE OLD PROCESSOR STATUS
4461 032540 013700 177776      MOV      PSW,R0           ;READ PSW FOR ERROR PRINT OUT
4462 032544 104124      ERROR   124              ;WRONG VECTOR, POSSIBLE WRONG PSW
4463
4464
4465 032546 013737 177572 001250  RDMMRO: MOV      MMR0,PMR0      ;READ MMR0 TO BE CHECKED LATER
4466 032554 013737 177574 001252      MOV      MMR1,PMR1      ;READ MMR1 FOR POSSIBLE ERROR REPORT
4467 032562 013737 177576 001254      MOV      MMR2,PMR2      ;READ MMR2 FOR POSSIBLE ERROR REPORT
4468 032570 013746 001310      MOV      OLDPS,-(KSP)     ;PUSH OLD PROCESSOR STATUS ON STACK
4469 032574 013746 001306      MOV      OLDPC,-(KSP)     ;PUSH RETURN ADDRESS ON STACK
4470 032600 000006      RTT                      ;RETURN TO TEST TO CHECK PMMR0
4471
4472
4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486 032602 105037 001362      KBTST: CLR      @MKB11CM   ;RESET THE MP FLAG
4487 032606 005037 001360      CLR      @MKB11E         ;CLEAR KB11E AND KB11EM FLAGS
4488 032612 012737 033050 000010      MOV      @MFPTTR,@RESVEC ;SET UP TRAP ADDRESS FOR MFPT AT RESERV VECTOR
4489 032620 000007      MFPT                    ;EXECUTE MFPT. WILL TRAP ON 1170 (KB11B/C) OR
4490                                     ;KB11CM (11/74 )
4491 032622 012737 000001 001360      T1:   MOV      #1,@MKB11E ;HERE IF KB11E OR KB11EM. SET FLAG
4492 032630 005037 177750      CLR      @MAINT          ;CLEAR THE MAINTENANCE REGISTER
4493 032634 005005      CLR      R5              ;RESET THE TEST COUNTER
4494 032636 012700 177746      MOV      @CONTRL,R0      ;GET THE ADDRESS OF...
4495 032642 012701 177750      MOV      @MAINT,R1       ;CCR,MAINT,AND MAPH00...
4496 032646 012702 170202      MOV      @MAPH00,R2      ;AND PLACE IN R0-R2
4497 032652 052710 040000      BIS      @BIT14,(R0)     ;TRY TO SET IVSS BIT
4498 032656 032710 040000      BIT      @BIT14,(R0)     ;DID IT SET?
4499 032662 001403      BEQ      T2              ;NO, GO TO NEXT TEST
4500 032664 042710 040000      BIC      @BIT14,(R0)     ;CLEAR IT.
4501 032670 005205      INC      R5              ;TEST IS POSITIVE
4502 032672 052711 000001 000001      T2:   BIS      @BIT0,(R1) ;SET EDMA IN MAINT REGISTER
4503 032676 032711 000001      BIT      @BIT0,(R1)
4504 032702 001410      BEQ      T3
4505 032704 052710 004000      BIS      @BIT11,(R0)    ;TRY TO SET DMA IN CCR
4506 032710 032710 004000      BIT      @BIT11,(R0)
4507 032714 001403      BEQ      T3
4508 032716 042710 004000      BIC      @BIT11,(R0)
4509 032722 005205      INC      R5
4510 032724 042711 000001 000001      T3:   BIC      @BIT0,(R1)  ;MAKE SURE EDMA IS CLEAR
4511 032730 052737 100000 172300      BIS      @BIT15,KIPDR0 ;TRY TO SET BYP ON A PDR
4512 032736 032737 100000 172300      BIT      @BIT15,KIPDR0
4513 032744 001404      BEQ      T4
4514 032746 042737 100000 172300      BIC      @BIT15,KIPDR0
4515 032754 005205      INC      R5
    
```

```

*** TEST FOR VARIOUS KB11 PROCESSORS ***
*THIS ROUTINE POLES THE RESULTS OF ATTEMPTS TO SET TO ONE
*CERTAIN CRITICAL BITS THAT ARE KNOWN TO BE OPERATIVE ON A KB11CM,
*OR KB11EM PROCESSOR. IF TWO OUT OF FOUR OF THE TESTS ARE
*POSITIVE THEN THE KB11CM OR KB11EM FLAG IS SET, IF LESS THAN TWO OF THE
*TESTS ARE POSITIVE THEN THE KB11E FLAG OR NO FLAG IS SET. THE DETERMINATION
*OF WHICH PAIR IS VALID IS BASED ON THE RESULTS OF EXECUTING AN MFPT OPCODE
*(OPCODE 7). IF THIS INSTRUCTION TRAPS THIS IS AN KB11CM OR
*A PLAIN 1170 (KB11-B OR KB11-C). IF THE INSTRUCTION DOES NOT TRAP THEN
*THIS IS A KB11-E OR KB11-EM.
    
```

```

4516 032756 052712 100000 T4: BIS #BIT15,(R2) ;TRY TO SET BYP ON UNIBUS MAP
4517 032762 032712 100000 BIT #BIT15,(R2)
4518 032766 001403 BEQ T.END
4519 032770 042712 100000 BIC #BIT15,(R2)
4520 032774 005205 INC R5
4521 032776 022705 000002 T.END: CMP #2,R5 ;IS THE RESULT OF THE TEST >=2
4522 033002 101021 BHI 2$ ;NO,THIS IS A KB11E OR KB11-B/C (11/70)
4523 033004 005000 CLR R0
4524 033006 005037 177746 CLR @#CONTRL
4525 033012 013701 177746 3$: MOV @#CONTRL,R1
4526 033016 001402 BEQ 4$
4527 033020 005200 INC R0
4528 033022 001373 JNE 3$
4529 033024
4530 033024 005737 001360 4$: TST @#KB11E ;IS IT A KB11-E OR KB11-EM?
4531 033030 001404 BEQ 1$ ;BR IF NEITHER. MUST BE KB11CM
4532 033032 012737 000400 001360 MOV #BIT8,@#KB11E ;SET UPPER BYTE (KB11-EM)
4533 033040 000402 BR 2$ ;DONE
4534 033042 105237 001362 1$: INCB @#KB11CM ;YES, FLAG THIS AS A MODIFIED PROCESSOR
4535 033046 000207 2$: RTS PC ;DONE
4536
4537 033050 MFPTTR: ;HERE IF MFPT TRAPPED. SEE IF 1170 OR KB11CM
4538 033050 012716 032630 MOV #T1,(SP) ;SET UP RETURN ADDRESS FOR RTI
4539 033054 000002 RTI ;RETURN
4540 033056
4541 ENDKB:
4542 ;*****
4543 ;*
4544 ;SIZE MEMORY AND COMPARE IT WITH THE SYSTEM SIZE REGISTER
4545 ;PRINT A WARNING IF THEY DISAGREE.
4546 ;*
4547 ;*****
4548 033056 052737 000200 030636 SIZMEM: BIS #BIT07,#KT11
4549 033064 004737 030570 JSR PC,#SIZE
4550 033070 062737 000037 031142 ADD #37,#LSTBK
4551 033076 023737 177760 031142 CMP @#SIZELO,#LSTBK ;EQUAL?
4552 033104 001550 BEQ OKSIZ
4553 033106 104400 033114 TYPE ,65$ ;:TYPE ASCIZ STRING
4554 033112 000433 BR 64$ ;:GET OVER THE ASCIZ
4555 ;:65$: .ASCIZ <15><12>/WARNING- THE SIZE OF MEMORY IS DIFFERENT FROM THAT/
4556 033202 64$: TYPE ,67$ ;:TYPE ASCIZ STRING
4557 033202 104400 033210 BR 66$ ;:GET OVER THE ASCIZ
4558 033206 000425 ;:67$: .ASCIZ <15><12>/INDICATED BY THE SYSTEM SIZE REGISTER./
4559 66$:
4560 033262 68$: TYPE ,69$ ;:TYPE ASCIZ STRING
4561 033262 104400 033270 BR 68$ ;:GET OVER THE ASCIZ
4562 033266 000421 ;:69$: .ASCIZ <15><12>/ SIZEHI SIZELO ACTUAL/
4563 68$:
4564 033332
4565 033332 104400 001217 TYPE ,SCLF
4566 033336 013746 177762 MOV @#SIZEHI,-(SP) ;:SAVE @#SIZEHI FOR TYPEOUT
4567 033342 104404 TYPOS ;:GO TYPE--OCTAL ASCII
4568 033344 006 .BYTE 6 ;:TYPE 6 DIGIT(S)
4569 033345 000 .BYTE 0 ;:SUPPRESS LEADING ZEROS
4570 033346 104400 033354 TYPE ,71$ ;:TYPE ASCIZ STRING
4571 033352 000404 BR 70$ ;:GET OVER THE ASCIZ

```

```

4572
4573 033364
4574 033364 013746 177760
4575 033370 104404
4576 033372 006
4577 033373 000
4578 033374 104400 033402
4579 033400 000404
4580
4581 033412
4582 033412 013746 031142
4583 033416 104404
4584 033420 006
4585 033421 000
4586 033422 104400 001217
4587 033426 000207
4588
4589
4590
4591

::71$: .ASCIZ / /
70$:
MOV @#SIZELO,-(SP) ::SAVE @#SIZELO FOR TYPEOUT
TYPOS ::GO TYPE--OCTAL ASCII
.BYTE 6 ::TYPE 6 DIGIT(S)
.BYTE 0 ::SUPPRESS LEADING ZEROS
TYPE ,73$ ::TYPE ASCIZ STRING
BR ,72$ ::GET OVER THE ASCIZ

::73$: .ASCIZ / /
72$:
MOV $LSTBK,-(SP) ::SAVE $LSTBK FOR TYPEOUT
TYPOS ::GO TYPE--OCTAL ASCII
.BYTE 6 ::TYPE 6 DIGIT(S)
.BYTE 0 ::SUPPRESS LEADING ZEROS
TYPE ,$CRLF ::DO CARRIAGE RETURN,LINE F ED
OKSIZ: RTS PC ::RETURN
    
```



```

4592 .SBTTL
4593 .SBTTL ***** ENTRY POINT 1 --- STARTING ADDRESS 200 *****
4594 .SBTTL ***** TEST CODE STARTS AT ADDRESS 40000 *****
4595 040000 .S=40000
4596
4597 040000 012737 000000 001222 STRT1: MOV #0,FSTTST ;LOAD INDEX TO START TABLE
4598 040006 000433 BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
4599 040010 012737 000002 001222 STRT2: MOV #2,FSTTST ;LOAD INDEX TO START TABLE
4600 040016 000427 BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
4601 040020 012737 000004 001222 STRT3: MOV #4,FSTTST ;LOAD INDEX TO START TABLE
4602 040026 000423 BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
4603 040030 012737 000006 001222 STRT4: MOV #6,FSTTST ;LOAD INDEX TO START TABLE
4604 040036 000417 BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
4605 040040 012737 000010 001222 STRT5: MOV #10,FSTTST ;LOAD INDEX TO START TABLE
4606 040046 000413 BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
4607 040050 012737 000012 001222 STRT6: MOV #12,FSTTST ;LOAD INDEX TO START TABLE
4608 040056 000407 BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
4609 040060 012737 000014 001222 STRT7: MOV #14,FSTTST ;LOAD INDEX TO START TABLE
4610 040066 000403 BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
4611 040070 012737 000016 001222 STRT8: MOV #16,FSTTST ;LOAD INDEX TO START TABLE
4612
4613
4614
4615
4616 040076 012706 001100 START: MOV #KERSTK,KSP ;SET UP KERNAL STACK
4617 040102 012737 001014 177746 MOV #1014,CONTRL ;TURN OFF CACHE
4618 040110 012706 001100 MOV #SCMTAG,R6 ;FIRST LOCATION TO BE CLEARED
4619 040114 005026 2$: CLR (R6)+ ;CLEAR MEMORY LOCATION
4620 040116 022706 001140 CMP #STKS,R6 ;DONE?
4621 040122 001374 BNE 2$ ;LOOP BACK IF NO
4622 040124 012737 025474 000020 MOV #SSCOPE,IOTVEC ;IOT VECTOR FOR SCOPE ROUTINE
4623 040132 012737 000340 000022 MOV #340,IOTVEC+2 ;PRIORITY LEVEL 7
4624 040140 012737 025760 000030 MOV #SEERRR,EMTVEC ;EMT VECTOR FOR ERROR ROUTINE
4625 040146 012737 000340 000032 MOV #340,EMTVEC+2 ;PRIORITY LEVEL 7
4626 040154 012737 030326 000034 MOV #STRAP,TRAPVEC ;TRAP VECTOR FOR TRAP CALLS
4627 040162 012737 000340 000036 MOV #340,TRAPVEC+2 ;PRIORITY LEVEL 7
4628 040170 012737 030370 000024 MOV #SPWRDN,PWRVEC ;POWER FAILURE VECTOR
4629 040176 012737 000340 000026 MOV #340,PWRVEC+2 ;PRIORITY LEVEL 7
4630 040204 013737 025240 025232 MOV SENDCT,SEOPCT ;SET UP END-OF-PROGRAM COUNTER
4631 040212 005037 001206 CLR $TIMES ;INITIALIZE NUMBER OF ITERATIONS
4632 040216 005037 001210 CLR $ESCAPE ;CLEAR THE ESCAPE ON ERROR ADDRESS
4633 040222 012737 000001 001117 MOV# #1,SEMAX ;ALLOW ONE ERROR PER TEST
4634 040230 012737 025460 000014 MOV #SRTRN,TBITVEC ;SET 'T' BIT VECTOR TO SRTRN
4635 040236 012737 000340 000016 MOV #340,TBITVEC+2 ;PRIORITY LEVEL 7
4636 040244 012737 000006 025460 MOV #RTT,SRTRN ;SET SRTRN TO A RTT
4637 040252 005037 025466 CLR $TBIT ;CLEAR 'T' BIT SWITCH
4638 040256 012737 040256 001110 MOV #.,SLPADR ;INITIALIZE LOOP ADDRESS
4639 040264 012737 040264 001112 MOV #.,SLPERR ;INITIALIZE LOOP ON ERROR
4640
4641 040272 005037 177572 LOOP: CLR #R0 ;GET INTO 16 BIT MODE ON SECOND PASS
4642
4643 040276 004737 032602 JSR PC,KBTST ;SEE IF KB11-E PROCESSOR
4644 040302 005037 172516 CLR #R3 ;TURN OFF EVERY THING POSSIBLE
4645 040306 012700 177777 MOV #-1,R0 ;PUT NEG ONE IN R0 TO INITIALIZE FLAGS
4646 040312 010037 031672 MOV R0,CPFLAG ;INITIALIZE CPU ERROR FLAG
4647 040316 010037 031776 MOV R0,PAFLAG ;INITIALIZE PARITY ERROR FLAG
    
```

```

4648 040322 010037 032230      MOV      R0,MMFLAG      ;INITIALIZE MEMORY MANAGEMENT TRAP FLAG
4649 040326 010037 032350      MOV      R0,NDFLAG      ;INITIALIZE NO D-SPACE TRAP FLAG
4650 040332 010037 177744      MOV      R0,MEMERR      ;CLEAR PARITY ERROR REGISTER (177744)
4651 040336 010037 177766      MOV      R0,CPUERR      ;CLEAR CPU ERROR REGISTER (177766)
4652 040342 005037 001224      CLR      CPUEXP         ;NOT EXPECTING ANY CPU TRAPS
4653 040346 005037 001226      CLR      MMEXP         ;NOT EXPECTING ANY MEMORY MANAGEMENT TRAPS
4654 040352 012737 031670 0C0004  MOV      #CPUER,ERRVEC  ;LOAD ADDRESS OF CPU TRAP ROUTINE
4655 040360 012737 000340 000006  MOV      #340,ERRVEC+2  ;SET PRIORITY LEVEL 7
4656 040366 012737 031774 000114  MOV      #MEMER,CACHVEC ;LOAD ADDRESS OF PARITY TRAP ROUTINE
4657 040374 012737 000340 000116  MOV      #340,CACHVEC+2 ;SET PRIORITY LEVEL 7
4658 040402 012737 032226 000250  MOV      #MMTRAP,MMVEC  ;LOAD ADDRESS OF MEMORY MANAGEMENT TRAP
4659 040410 012737 000340 000252  MOV      #340,MMVEC+2   ;SET PRIORITY LEVEL 7
4660 040416 004737 031346      JSR      PC,CLEANUP     ;INITIALIZE ALL ERROR LOCATIONS
4661 040422 012706 001100      MOV      #KERSTK,KSP   ;SET UP KERNEL STACK POINTER
4662 040426 005227 177777      INC      #-1           ;:FIRST TIME?
4663 040432 001024      BNE      64$          ;:BRANCH IF NO
4664 040434 022737 025412 000042  CMP      #SENDAD,@#42  ;:ACT-11?
4665 040442 001420      BEQ      64$          ;:BRANCH IF YES
4666 040444 104400 040452      TYPE    ,65$         ;:TYPE ASCIZ STRING
4667 040450 000415      BR       64$          ;:GET OVER THE ASCIZ
4668      ;:65$: .ASCIZ <CRLF>?CEKBEE0 11/70 MEM MGMT?<CRLF>
4669 040504      64$:
4670
4671 040504 005227 177777      INC      #-1           ;:FIRST TIME?
4672 040510 001026      BNE      100$        ;:BR IF NO
4673 040512 104400 003324      TYPE    ,MSG1         ;:<15><12>CPU UNDER TEST FOUND TO BE A
4674 040516 005737 001360      TST     @#KB11E       ;:IS THIS A KB11-E OR KB11-EM?
4675 040522 001011      BNE      101$        ;:BR IF EITHER ONE
4676 040524 105737 001362      TSTB   @#KB11CM      ;:IS IT A 11/74 (KB11CM)
4677 040530 001003      BNE      1$          ;:BR IF IT IS
4678 040532 104400 003374      TYPE    ,MSG3         ;:KB11-B/C<15><12>
4679 040536 000413      BR      100$        ;:SKIP OTHER MESSAGE
4680 040540 104400 003406      1$: TYPE    ,MSG4         ;:11/74 (KB11CM)<15><12>
4681 040544 000410      BR      100$        ;:SKIP CISP MESSAGE
4682 040546 105737 001360      101$: TSTB   @#KB11E       ;:IS IT A KB11-E?
4683 040552 001403      BEQ     102$        ;:BR IF NOT. MUST BE KB11-EM
4684 040554 104400 003437      TYPE    ,MSG5         ;:KB11-E<15><12>
4685 040560 000402      BR      100$        ;:SKIP KB11-EM MESSAGE
4686 040562 104400 003363      102$: TYPE    ,MSG2         ;:KB11-EM<15><12>
4687 040566      100$:
4688 040566 005227 177777      INC      #-1           ;:FIRST TIME?
4689 040572 001002      BNE      110$        ;:BR IF NOT
4690 040574 004737 033056      JSR     PC,SIZMEM     ;:SIZE MEM AND COMPARE WITH SIZE REG
4691 040600 013700 001222      110$: MOV     FSTTST,R0     ;:LOAD START TABLE INDEX
4692 040604 001417      BEQ     TST1         ;:START WITH TEST ONE IF ZERO
4693 040606 012737 040000 177776  MOV     #40000,PSW    ;:GO TO SUPERVISOR MODE
4694 040614 012706 000700      MOV     #SUPSTK,SSP   ;:SET UP SUPERVISOR STACK POINTER
4695 040620 012737 140000 177776  MOV     #140000,PSW   ;:GO TO USER MODE
4696 040626 012706 000600      MOV     #USESTK,USP   ;:SET UP USER STACK POINTER
4697 040632 012737 000340 177776  MOV     #340,PSW      ;:GO TO KERNEL MODE PRIORITY LEVEL 7
4698 040640 000170 001336      JMP     @STRTAB(R0)   ;:JUMP TO CORRECT ENTRY POINT
4699
4700
4701      ;* THIS FIRST GROUP OF TESTS IS FOR TESTING THE ADDRESS DECODE
4702      ;* LOGIC FOR THE INTERNAL REGISTERS ON PAGE 'SCCE' AND TO MAKE
4703      ;* SOME DETERMINATION ABOUT THE RESPONDING ADDRESSES. IF AN
    
```

4704 : * ADDRESS DOES NOT FUNCTION AS THE CORRESPONDING REGISTER SHOULD
 4705 : * AN ERROR WILL BE FLAGGED. THE MULTIPLEXERS AND INTERNAL BUS
 4706 : * DRIVERS ON PAGES 'SCCM' AND 'SCCN' ARE ALSO UTILIZED IN
 4707 : * THESE TESTS AND COULD BE THE SOURCE OF ANY PROBLEMS ENCOUNTERED.
 4708
 4709

4710 : *****
 4711 : *TEST 1 TRY TO READ ALL CPU REGISTERS
 4712 : *

4713 : * THIS TEST ENSURES THAT SOMETHING RESPONDS TO ADDRESSES 177760
 4714 : * THRU 177776. IF A TRAP TO 'ERRVEC' (004) OCCURS IT IS ASSUMED
 4715 : * THAT A REGISTER HAS TIMED OUT, AND AN ERROR IS REPORTED.
 4716 : *

4717 : * ERRORS THAT OCCUR IN THIS TEST ARE REPORTED FROM AN AREA
 4718 : * AT THE END OF THIS TEST, STARTING AT '50\$'.
 4719 : *

4720 : *****

4721	040644				TST1:			
4722	040644	012737	040702	001110		MOV	#20\$,SLPADR	:SET LOOP ADDRESS POINTER TO 20\$
4723	040652	012737	040702	001112		MOV	#20\$,SLPERR	:SET LOOP ON ERROR POINTER TO 20\$
4724	040660	012737	000001	001102		MOV	#1,\$TSTNM	:LOAD TEST NUMBER INTO MEMORY
4725	040666	013737	001102	177570		MOV	\$TSTNM,DISPLAY	:DISPLAY TEST NUMBER FOR THIS TEST
4726	040674	012737	040766	001316		MOV	#TST2,NXTTST	:SAVE STARTING ADDRESS OF NEXT TEST
4727								:FOR ESCAPE ON PARITY ERRORS
4728	040702	012737	040726	001112	20\$:	MOV	#1\$,SLPERR	:SET LOOP ON ERROR POINTER TO 1\$
4729	040710	012737	040756	000004		MOV	#50\$,ERRVEC	:SET TIME OUT VECTOR TO SPECIAL ROUTINE
4730	040716	012706	001100			MOV	#1100,KSP	:SET KERNEL STACK POINTER TO 1100
4731	040722	012700	177760			MOV	#177760,R0	:PUT FIRST CPU REGISTER ADDRESS IN R0
4732	040726	000240			1\$:	NOP		:THIS IS A SYNC POINT FOR SCOPING
4733	040730	011001				MOV	(R0),R1	:TRY TO READ THE CPU REGISTERS
4734	040732	062700	000002		100\$:	ADD	#2,R0	:POINT TO NEXT CPU REGISTER
4735	040736	001373				BNE	1\$:BRANCH IF NOT ALL READ YET
4736	040740	012737	040702	001112		MOV	#20\$,SLPERR	:SET LOOP POINTER TO START OF TEST
4737	040746	012737	031670	000004		MOV	#CPUER,ERRVEC	:SET UP C.P.U. ERROR VECTOR
4738	040754	000404				BR	TST2	:TEST OVER BRANCH TO NEXT TEST
4739								

4740								
4741	040756	062706	000004		50\$:	ADD	#4,KSP	:RE ADJUST KERNEL STACK POINTER
4742	040762	104024				ERROR	24	:CPU REGISTER TIMED OUT
4743	040764	000762				BR	100\$:GO BACK AND FINISH TEST
4744								

4745 : *****

4746 : *TEST 2 SYSTEM SIZE REGISTERS
 4747 : *

4748 : * BOTH THE LO SIZE AND THE HI SIZE REGISTERS ARE READ TO SEE THAT
 4749 : * THEY HOLD SOMETHING THAT LOOKS CORRECT. (IE. LO SIZE SHOULD
 4750 : * HAVE 377 IN ITS LOW BYTE, AND HI SIZE SHOULD BE ZERO) THEY
 4751 : * ARE ALSO VERIFIED TO BE READ ONLY REGISTERS.
 4752 : *

4753 : *****

4754					TST2:			
4755	040766					SCOPE		
4756	040766	000004				MOV	#TST3,NXTTST	:SAVE STARTING ADDRESS OF NEXT
4757	040770	012737	041122	001316				:TEST FOR ESCAPE ON PARITY ERRORS
4758								:SET LOOP ON ERROR POINTER TO 21\$
4759	040776	012737	041010	001112	20\$:	MOV	#21\$,SLPERR	

```

4760 041004 012700 177760          MOV      #177760,R0          ;PUT ADDRESS OF SIZE LO REGISTER INTO R0
4761 041010 000240          21$:    NOP                      ;THIS IS A SYNC POINT FOR SCOPING
4762 041012 011001          MOV      (R0),R1          ;READ SYSTEM SIZE LO REGISTER
4763 041014 122701 000377          CMPB    #377,R1          ;SEE IF LOW BYTE IS ALL ONES
4764 041020 001401          BEQ     1$                ;BRANCH IF LOW BYTE IS ALL ONES
4765 041022 104025          ERROR   25                ;LOW BYTE OF LO SIZE IS NOT -1
4766 041024 012737 041032 001112 1$:    MOV      #22$,SLPERR      ;SET LOOP ON ERROR POINTER TO 22$
4767 041032 000240          22$:    NOP                      ;THIS IS A SYNC POINT FOR SCOPING
4768 041034 005010          CLR     (R0)              ;TRY TO CLEAR SIZE LO REGISTER
4769 041036 011002          MOV      (R0),R2          ;READ SIZE LO REGISTER INTO R2
4770 041040 001002          BNE     2$                ;BRANCH IF IT IS NOT ZERO
4771 041042 104026          ERROR   26                ;COULD WRITE SIZE LO REG
4772 041044 010110          MOV      R1,(R0)          ;RESTORE DATA TO ADDRESS
4773 041046 012737 041060 001112 2$:    MOV      #23$,SLPERR      ;SET LOOP ON ERROR POINTER TO 23$
4774 041054 012700 177762          MOV      #177762,R0      ;PUT ADDRESS OF SIZE HIGH REG INTO R0
4775 041060 000240          23$:    NOP                      ;THIS IS A SYNC POINT FOR SCOPING
4776 041062 011001          MOV      (R0),R1          ;READ SYSTEM SIZE HIGH REGISTER
4777 041064 001401          BEQ     3$                ;BRANCH IF IT IS ZERO
4778 041066 104027          ERROR   27                ;(177762) IS NOT ZERO
4779 041070 012737 041076 001112 3$:    MOV      #24$,SLPERR      ;SET LOOP ON ERROR POINTER TO 24$
4780 041076 000240          24$:    NOP                      ;THIS IS A SYNC POINT FOR SCOPING
4781 041100 012710 177777          MOV      #-1,(R0)         ;TRY TO LOAD SIZE HIGH REGISTER
4782 041104 011002          MOV      (R0),R2          ;READ SIZE HIGH REGISTER
4783 041106 001402          BEQ     4$                ;BRANCH IF REGISTER IS STILL ZERO
4784 041110 104026          ERROR   26                ;COULD WRITE SIZE REGISTER
4785 041112 010110          MOV      R1,(R0)         ;RESTORE DATA TO ADDRESS
4786 041114 012737 040776 001112 4$:    MOV      #20$,SLPERR      ;SET LOOP POINTER TO START OF TEST
    
```

4787
4788
4789

```

:*****
:*TEST 3          CPU ERROR REGISTER
    
```

```

:*
:*      THE CPU ERROR REGISTER IS TESTED TO SEE THAT IT IS CLEAR
:*      AFTER A NEGATIVE ONE HAS BEEN LOADED INTO IT.  THERE IS NO
:*      WAY TO SET ANY BITS UNDER PROGRAM CONTROL.
    
```

4790
4791
4792
4793
4794
4795
4796
4797 041122
4798 041122 000004
4799 041124 012737 041166 001316

```

:*****
:TST3:
    
```

```

4800          SCOPE
4801          MOV      #TST4,NXTTST          ;SAVE STARTING ADDRESS OF NEXT
4802          20$:    MOV      #1$,SLPERR          ;TEST FOR ESCAPE ON PARITY ERRORS
4803          1$:    MOV      #177766,R0          ;SET LOOP ON ERROR POINTER TO 1$
4804          MOV      #-1,(R0)              ;LOAD ADDRESS OF C.P.U. ERROR REG
4805          MOV      (R0),R1              ;THIS IS A SYNC POINT FOR SCOPING
4806          BEQ     2$                ;WRITE A NEGATIVE ONE INTO CPU ERROR REG
4807          ERROR   30                ;READ C.P.U. ERROR REGISTER
4808          2$:    MOV      #20$,SLPERR      ;CPU ERROR REGISTER SHOULD BE ZERO
4809          ;C.P.U. ERROR REGISTER NOT ZERO
4810          ;SET LOOP POINTER TO START OF TEST
    
```

4811
4812

```

:*****
:*TEST 4          MICRO PROGRAM BREAK REGISTER
    
```

```

:*
:*      IN A STANDARD KB11-C PROCESSOR
    
```

4813
4814
4815

```

4816      : * THE MICRO BREAK REGISTER IS A LOW BYTE ONLY REGISTER, WITH
4817      : * THE UPPER BYTE ALWAYS ZERO. THIS TEST LOADS A FULL WORD INTO
4818      : * ADDRESS 177770 AND CHECKS TO SEE THAT ONLY THE LOW BYTE IS
4819      : * STORED IN THE REGISTER. IF THIS IS A (KB11-E)
4820      : * THEN ALL 16 BITS ARE WRITABLE AND THE REG IS TESTED AS SUCH.
4821      : * A ONE IS SHIFTED ACROSS THE REG IN BOTH CASES IN THIS TEST.
4822      : *
4823      : *
4824      : * *****
4825      : * TST4:
4826      : * SCOPE
4827      : * MOV #TST5,NXTTST ;SAVE STARTING ADDRESS OF NEXT
4828      : * ;TEST FOR ESCAPE ON PARITY ERRORS
4829      : * ;SET LOOP ON ERROR POINTER TO 1$
4830      : * ;LOAD ADDRESS OF MICRO BREAK REGISTER
4831      : * ;THIS IS A SYNC POINT FOR SCOPING
4832      : * ;START WITH ALL ZERO DATA
4833      : * ;READ MICRO BREAK REGISTER
4834      : * ;CHECK FOR ZERO
4835      : * ;DID YOU REFERENCE THE MICRO BREAK REG
4836      : * ;BRANCH IF DATA WRONG
4837      : * ;START SHIFTING A ONE
4838      : * ;SET UP EXPECTED DATA IN R2
4839      : * ;WRITE MICRO BREAK REG
4840      : * ;READ IT BACK
4841      : * ;IS THIS A KB11E OR KB11EM PROCESSOR?
4842      : * ;BR IF KB11-E
4843      : * ;ONLY LOWER BYTE SHOULD BE WRITTEN
4844      : * ;MATCH?
4845      : * ;BR IF NOT
4846      : * ;CLEAR CARRY FOR ROTATE
4847      : * ;SHIFT A 1 IN DATA TO WRITE
4848      : * ;DONE IF CARRY SETS
4849      : * ;REPEAT IF NOT DONE
4850      : * ;ERROR COMES HERE. IS THIS A KB11-E OR KB11-EM?
4851      : * ;BR TO ERROR EM32 IF NOT KB11-E OR KB11-EM
4852      : * ;ERROR FOR KB11-E PROCESSOR
4853      : * ;DONE
4854      : * ;DIDN'T LOAD MICRO BREAK REGISTER
4855      : * ;LEAVE MICRO BREAK REGISTER ZERO
4856      : * ;SET LOOP POINTER TO START OF TEST
4857
4858
4859      : * *****
4860      : * TEST 5 PROGRAM INTERRUPT REQUEST REGISTER
4861      : *
4862      : * THE PROGRAM INTERRUPT REQUEST REGISTER'S UPPER BYTE IS LOADED
4863      : * DIRECTLY AND THE LOWER BYTE IS AN ENCODE OF THE UPPER BYTE.
4864      : * THIS TEST LOADS A FULL WORD INTO ADDRESS 177772 AND CHECKS
4865      : * THAT THE UPPER BYTE IS LOADED DIRECTLY AND THE LOWER BYTE
4866      : * IS ENCODED CORRECTLY.
4867      : *
4868      : * *****
4869      : * TST5:
4870      : * SCOPE
4871      : * MOV #TST6,NXTTST ;SAVE STARTING ADDRESS OF NEXT
    
```

```

4872                                     ;TEST FOR ESCAPE ON PARITY ERRORS
4873 041320 012737 041334 001112 20$: MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
4874 041326 000237                                     ;SET PRIORITY LEVEL AT SEVEN
4875 041330 012700 177772                                     ;LOAD ADDRESS OF PROGRAM INTERRUPT REG
4876 041334 000240                                     ;THIS IS A SYNC POINT FOR SCOPING
4877 041336 012710 004777 1$: NOP ;ONLY UPPER BYTE (LESS BIT 8) IS LOADED
4878                                     ;DIRECTLY, LOWER BYTE IS ENCODED
4879 041342 011001                                     ;READ PROGRAM INTERRUPT REGISTER
4880 041344 012702 004146 MOV #004146, R2 ;LOAD EXPECTED DATA
4881 041350 020102 CMP R1, R2 ;DID YOU REFERENCE PROGRAM INTERRUPT REG
4882 041352 001401 BEQ 2$ ;BRANCH IF IT WAS THE PROG. INTERRUPT
4883 041354 104033 ERROR 33 ;NOT THE PROG. INT. REQUEST REGISTER
4884 041356 005010 2$: CLR (R0) ;LEAVE THE P.I.R. REGISTER ZERO
4885 041360 012737 041320 001112 MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4899
4900
4901
4902
4903
4904
4905
4906
4907
4908
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
    
```

 :TEST 6 STACK LIMIT REGISTER

THE STACK LIMIT REGISTER IS A HIGH BYTE ONLY REGISTER, SO THIS TEST TRIES TO LOAD BOTH BYTES OF ADDRESS 177774 AND CHECKS THAT ONLY THE HIGH BYTE IS LOADED. THE LOW BYTE WILL ALWAYS BE READ AS ZERO.

 TST6:

```

4897 041366 SCOPE
4898 041366 000004 MOV #TST7, NXTTST ;SAVE STARTING ADDRESS OF NEXT
4899 041370 012737 041442 001316 ;TEST FOR ESCAPE ON PARITY ERRORS
4900                                     ;SET LOOP ON ERROR POINTER TO 1$
4901 041376 012737 041410 001112 20$: MOV #1$, $LPERR ;LOAD ADDRESS OF STACK LIMIT REGISTER
4902 041404 012700 177774 1$: MOV #177774, R0 ;THIS IS A SYNC POINT FOR SCOPING
4903 041410 000240                                     ;ONLY HIGH BYTE SHOULD BE LOADED
4904 041412 012710 000777 MOV #777, (R0) ;READ STACK LIMIT REGISTER
4905 041416 011001 MOV (R0), R1 ;LEAVE STACK LIMIT REGISTER CLEAR
4906 041420 005010 CLR (R0) ;LOAD EXPECTED DATA INTO R2
4907 041422 012702 000400 MOV #400, R2 ;DID YOU REFERENCE THE STACK LIMIT REG
4908 041426 020102 CMP R1, R2 ;BRANCH IF IT WAS STACK LIMIT
4909 041430 001401 BEQ 2$ ;NOT STACK LIMIT REGISTER
4910 041432 104034 ERROR 34 ;SET LOOP POINTER TO START OF TEST
4911 041434 012737 041376 001112 2$: MOV #20$, $LPERR
4912
4913
4914
4915
4916
4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
    
```

 :TEST 7 PROCESSOR STATUS WORD

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

 TST7:

```

4923 041442 SCOPE
4924 041442 000004 MOV #TST10, NXTTST ;SAVE STARTING ADDRESS OF NEXT
4925 041444 012737 041520 001316 ;TEST FOR ESCAPE ON PARITY ERRORS
4926                                     ;SET LOOP ON ERROR POINTER TO 1$
4927 041452 012737 041470 001112 20$: MOV #1$, $LPERR
    
```

```

4928 041460 012700 177776      MOV      #177776,R0      ;LOAD ADDRESS OF P.S.W. INTO R0
4929 041464 012702 177740      MOV      #177740,R2      ;LOAD EXPECTED DATA INTO R2
4930 041470 000237      1S:    SPL      7          ;SET PRIORITY TO LEVEL SEVEN
4931 041472 000257      CCC      ;CLEAR ALL CONDITION CODES
4932 041474 000240      NOP      ;THIS IS A SYNC POINT FOR SCOPING
4933 041476 111001      MOVB     (R0),R1         ;LOAD LOWER BYTE OF P.S.W. INTO R1
4934 041500 042701 000020      BIC     #TBIT,R1        ;CLEAR T-BIT IF IT IS ON
4935 041504 020201      CMP     R2,R1           ;SIGN EXTEND OF LOWER BYTE OF PSW
4936                                     ;LOWER BYTE SHOULD BE 340
4937 041506 001401      BEQ     2$             ;BRANCH IF PSW IS CORRECT
4938 041510 104031      ERROR   31            ;MUST HAVE READ SOME OTHER REGISTER
4939 041512 012737 041452 001112 2S:    MOV     #20$,$LPERR    ;SET LOOP POINTER TO START OF TEST
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4950

```

```

:*****
:TEST 10      SET UP THE STACK POINTERS FOR REST OF TESTS
:
:      THIS TEST IS USED TO SET THE KERNEL STACK POINTER AT
:      1100, THE SUPERVISOR STACK POINTER AT 700, AND THE USER
:      STACK POINTER AT 600. IT THEN RETURNS TO KERNEL MODE AND
:      VERIFIES THAT THE KERNEL STACK POINTER IS STILL AT 1100.
:*****

```

```

4951 041520      TST10:
4952 041520 000004      SCOPE
4953 041522 012737 041602 001316      MOV     #TST11,NXTTST ;SAVE STARTING ADDRESS OF NEXT
4954                                     ;TEST FOR ESCAPE ON PARITY ERRORS
4955 041530 005037 177776      CLR     PSW           ;GET TO KERNEL MODE, PRIORITY 0
4956 041534 012706 001100      MOV     #KERSTK,KSP   ;SETUP KERNEL STACK POINTER
4957 041540 012737 040000 177776      MOV     #40000,PSW    ;GET INTO SUPERVISOR MODE
4958 041546 012706 000700      MOV     #SUPSTK,SSP   ;SET UP SUPERVISOR STACK POINTER
4959 041552 012737 140000 177776      MOV     #140000,PSW   ;GET INTO USER MODE
4960 041560 012706 000600      MOV     #USESTK,USP   ;SET UP USER STACK POINTER
4961 041564 005037 177776      CLR     PSW           ;GET BACK INTO KERNEL MODE
4962 041570 010600      MOV     KSP,R0        ;READ KSRNEL STACK POINTER
4963 041572 022700 001100      CMP     #KERSTK,R0    ;SEE IF KERNEL STACK IS STILL 1100
4964 041576 001401      BEQ     TST11         ;:BRANCH IF KERNEL STACK IS OKAY
4965 041600 104035      ERROR   35            ;:KERNEL STACK POINTER IS NOT RIGHT
4966
4967
4968
4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4980
4981
4982
4983

```

```

.SBTTL ***** ENTRY POINT 2 --- STARTING ADDRESS 204 *****
.SBTTL ** TEST READ/WRITE BITS IN MEMORY MANAGEMENT STATUS REGISTERS **
:
:      THIS GROUP OF TESTS EXERCISES ALL OF THE READ/WRITE BITS
:      IN MMR0, TESTS THE PROPER FUNCTIONING OF MMR1, AND TRIES A
:      FEW VIRTUAL ADDRESSES IN MMR2. IT ALSO EXERCISES THE BITS
:      IN MMR3. THESE TESTS SHOW THAT ONCE THE 'SSRC NO ERROR' FLIP-
:      FLOP IS SET MMR1 AND MMR2 STOP TRACKING THE C.P.U. OPERATIONS.
:

```

```

:*****
:TEST 11      BIT TEST OF MEMORY MANAGEMENT REGISTER 0
:
:      THIS TEST TRIES TO SET AND CLEAR BITS <15:12> AND <09> OF
:      MMR0. 'INIT' IS ISSUED TO SEE THAT IT WILL CLEAR THESE BITS.
:*****

```

```

4984      : * THE OTHER BITS OF MPRO ARE CLOKED ONLY ON MEMORY
4985      : * MANAGEMENT ERROR CONDITIONS IF RELOCATION IS ENABLED.
4986      : * BIT <00> ENABLES FULL RELOCATION AND BIT <08> ENABLES
4987      : * RELOCATION ON THE DESTINATION CYCLE ONLY. THESE BITS WILL
4988      : * BE TESTED IN A LATER TEST.
4989      : *
4990      : *****
4991      TST11:
4992      041602 000004          SCOPE
4993      041604 012737 042006 001316  MOV #TST12,NXTTST ;SAVE STARTING ADDRESS OF NEXT
4994      : ;TEST FOR ESCAPE ON PARITY ERRORS
4995      041612 ENTPT2:
4996      041612 012737 041642 001110  MOV #20$,SLPADR ;SET LOOP ADDRESS POINTER TO 20$
4997      041620 012737 041642 001112  MOV #20$,SLPERR ;SET LOOP ON ERROR POINTER TO 20$
4998      041626 012737 000011 001102  MOV #11,$TSTNM ;LOAD TEST NUMBER INTO MEMORY
4999      041634 013737 001102 177570  MOV $TSTNM,DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
5000      041642 012700 177572 20$:  MOV MPRO,R0 ;PUT ADDRESS OF MPRO IN R0
5001      041646 012710 171000  MOV #171000,(R0) ;LOAD WRITABLE BITS IN MPRO
5002      041652 000240  NOP ;THIS IS A SYNC POINT FOR SCOPING
5003      041654 000005  RESET ;'INIT', SHOULD CLEAR ALL BITS OF MPRO
5004      041656 011001  MOV (R0),R1 ;READ MPRO
5005      041660 001401  BEQ 1$ ;BRANCH IF MPRO IS CLEAR
5006      041662 104007  ERROR 7 ;MPRO WON'T CLEAR
5007      041664 012737 041672 001112 1$:  MOV #11$,SLPERR ;SET LOOP ON ERROR POINTER TO 11$
5008      041672 000240 11$:  NOP ;THIS IS SYNC POINT FOR SCOPING
5009      041674 012710 171000  MOV #171000,(R0) ;SET BITS <15:12><9> OF MPRO
5010      041700 011001  MOV (R0),R1 ;READ MPRO
5011      041702 022701 171000  CMP #171000,R1 ;SEE IF BITS <15:12><9> ARE SET
5012      041706 001401  BEQ 2$ ;BRANCH IF CORRECT BITS ARE SET
5013      041710 104010  ERROR 10 ;CAN'T SET 171000 IN MPRO
5014      041712 012737 041720 001112 2$:  MOV #12$,SLPERR ;SET LOOP ON ERROR POINTER TO 12$
5015      041720 000240 12$:  NOP ;THIS IS SYNC POINT FOR SCOPING
5016      041722 005010  CLR (R0) ;CLEAR UPPER 7 BITS OF MPRO
5017      041724 011001  MOV (R0),R1 ;READ MPRO
5018      041726 001401  BEQ 3$ ;BRANCH IF MPRO IS CLEAR
5019      041730 104007  ERROR 7 ;MPRO WON'T CLEAR
5020      041732 012737 041750 001112 3$:  MOV #5$,SLPERR ;SET LOOP ON ERROR POINTER TO 5$
5021      041740 012702 100000  MOV #BIT15,R2 ;SET BIT IN R2 TO FLOAT THRU MPRO
5022      041744 012703 000007  MOV #7,R3 ;PUT LOOP COUNT IN R3
5023      041750 000240 5$:  NOP ;THIS A IS SYNC POINT FOR SCOPING
5024      041752 010210  MOV R2,(R0) ;LOAD R2 INTO MPRO
5025      041754 011001  MOV (R0),R1 ;READ MPRO IN R1
5026      041756 005010  CLR (R0) ;CLEAR MPRO
5027      041760 010204  MOV R2,R4 ;PUT PATTERN IN R4
5028      041762 042704 006777  BIC #006777,R4 ;MASK OFF BITS <11:10><8:0>
5029      041766 020401  CMP R4,R1 ;COMPARE PATTERN WITH MPRO
5030      041770 001401  BEQ 4$ ;BRANCH IF DATA MATCHES
5031      041772 104011  ERROR 11 ;GOT WRONG DATA FROM MPRO
5032      041774 006002 4$:  ROR R2 ;SHIFT BIT TO RIGHT
5033      041776 077314  SOB R3,5$ ;LOOP BACK 6 TIMES
5034      042000 012737 041642 001112  MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
5035
5036
5037      : *****
5038      : *TEST 12 BIT TEST OF MEMORY MANAGEMENT REGISTER 1
5039      : *
    
```



```

5040 :* THIS TEST WILL CAUSE BITS IN MMR1 TO BE LOCKED BY SETTING
5041 :* BITS <15:13> IN MMR0. THE REGISTER ONLY GETS CLOKED ON
5042 :* AN INSTRUCTION THAT AUTO INCREMENTS OR AUTO DECREMENTS A
5043 :* REGISTER. THE LOWER BYTE IS ALWAYS CLOKED FIRST AND THE
5044 :* UPPER BYTE IS CLOKED ONLY IF BOTH SOURCE AND DESTINATION
5045 :* AUTO INCREMENT OR DECREMENT A REGISTER.
5046 :* ALL COUNTS (+1,-1,+2,-2) THAT CAN BE GENERATED WITH JUST THE
5047 :* C.P. ARE CLOKED INTO BOTH HIGH AND LOW BYTES. ALL REGISTER
5048 :* NUMBERS ARE GENERATED, INCLUDING ALL 3 R6'S, AND ALL THE
5049 :* BITS THAT HOLD THE REGISTER NUMBER IN BOTH BYTES ARE TESTED.
5050 :* HOWEVER NOT ALL REGISTER NUMBERS ARE CLOKED INTO BOTH HIGH
5051 :* AND LOW BYTES, BUT ENOUGH ARE USED TO TEST THE LOGIC.
5052 :*
5053 :*
5054 :*****
    
```

```

5054 042006 000004 TST12: SCOPE
5055 042006 012737 042440 001316 MOV #TST13,NXTTST ;SAVE STARTING ADDRESS OF NEXT
5056 042010 012701 177574 20$: MOV #MMR1,R1 ;TEST FOR ESCAPE ON PARITY ERRORS
5057 042016 012704 020000 MOV #BIT13,R4 ;PUT ADDRESS OF MMR1 IN R1
5058 042022 012737 042034 001112 MOV #11$,SLPERR ;SET READ ONLY BIT IN R4
5059 042026 012737 100000 177572 11$: NOP ;SET LOOP ON ERROR POINTER TO 11$
5060 042034 012737 013427 MOV #BIT15,@MMR0 ;THIS IS SYNC POINT FOR SCOPING
5061 042036 012703 011102 MOV #013427,R3 ;LOCK UP MMR1 LOWER BYTE 027
5062 042044 011102 (R1),R2 ;UPPER BYTE 027-WORD 013427
5063 042050 020203 CMP R2,R3 ;PUT EXPECTED DATA IN R3
5064 042052 001401 BEQ 10$ ;READ MMR1 INTO R2
5065 042054 104013 ERROR 13 ;SEE IF DATA MATCHES
5066 042056 012737 042066 001112 10$: MOV #12$,SLPERR ;BRANCH IF DATA MATCHES
5067 042060 000240 12$: NOP ;MMR1 DID NOT TRACK PROPERLY
5068 042066 000005 RESET ;SET LOOP ON ERROR POINTER TO 12$
5069 042070 011102 (R1),R2 ;THIS IS SYNC POINT FOR SCOPING
5070 042072 001401 BEQ 1$ ;ISSUE INIT
5071 042074 104012 ERROR 12 ;SEE IF MMR1 IS CLEARED
5072 042100 012737 042106 001112 1$: MOV #13$,SLPERR ;BRANCH IF MMR1 IS ZERO
5073 042106 012700 177572 13$: MOV #MMR0,R0 ;CAN'T CLEAR MMR1
5074 042112 000240 MOV #14$,SLPERR ;SET LOOP ON ERROR POINTER TO 13$
5075 042114 012720 040000 MOV #MMR0+2,R0 ;PUT ADDRESS OF MMR0 INTO R0
5076 042120 011102 (R1),R2 ;THIS IS SYNC POINT FOR SCOPING
5077 042122 012703 010027 MOV #010027,R3 ;LOCK UP MMR1-LOWER BYTE 027
5078 042126 020203 CMP R2,R3 ;UPPER BYTE 020-WORD 010027
5079 042130 001401 BEQ 2$ ;READ MMR1
5080 042132 104013 ERROR 13 ;PUT EXPECTED DATA IN R3
5081 042134 005037 177572 2$: CLR @MMR0 ;SEE IF DATA MATCHES
5082 042140 012737 042146 001112 14$: MOV #14$,SLPERR ;BRANCH IF DATA MATCHES
5083 042146 012700 177574 MOV #MMR0+2,R0 ;MMR1 DID NOT TRACK PROPERLY
5084 042152 000240 MOV R4,-(R0) ;CLEAR MMR0
5085 042154 010440 MOV (R1),R2 ;SET LOOP ON ERROR POINTER TO 14$
5086 042156 011102 MOV #000360,R3 ;PUT ADDRESS OF MMR0 PLUS 2 IN R0
5087 042160 012703 000360 CMP R2,R3 ;THIS IS SYNC POINT FOR SCOPING
5088 042164 001401 BEQ 3$ ;LOCK UP MMR1-LOWER BYTE 360
5089 042170 104013 ERROR 13 ;UPPER BYTE 000-WORD 000360
5090 042156 011102 (R1),R2 ;READ MMR1
5091 042160 012703 000360 MOV #000360,R3 ;PUT EXPECTED DATA IN R3
5092 042164 020203 CMP R2,R3 ;SEE IF DATA MATCHES
5093 042166 001401 BEQ 3$ ;BRANCH IF DATA MATCHES
5094 042170 104013 ERROR 13 ;MMR1 DID NOT TRACK PROPERLY
    
```

5096	042172	005010			3\$:	CLR	(R0)	:CLEAR MMR0
5097	042174	012737	042202	001112		MOV	#15\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 15\$
5098	042202	012705	177573		15\$:	MOV	MMR0+1, R5	:PUT ADDRESS OF MMR0'S UPPER BYTE IN R5
5099	042206	012702	001321			MOV	#READON+1, R2	:PUT ADDRESS OF READ ONLY BIT <13> IN R2
5100	042212	000240				NOP		:THIS IS SYNC POINT FOR SCOPING
5101	042214	112225				MOVB	(R2)+, (R5)+	:LOCK UP MMR1-LOWER BYTE 012
5102								:UPPER BYTE 015-WORD 006412
5103	042216	011102				MOV	(R1), R2	:READ MMR1
5104	042220	012703	006412			MOV	#006412, R3	:PUT EXPECTED DATA IN R3
5105	042224	020203				CMP	R2, R3	:SEE IF DATA MATCHES
5106	042226	001401				BEQ	4\$:BRANCH IF DATA MATCHES
5107	042230	104013				ERROR	13	:MMR1 DID NOT TRACK PROPERLY
5108	042232	005010			4\$:	CLR	(R0)	:CLEAR MMR0
5109	042234	012737	042242	001112		MOV	#16\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 16\$
5110	042242	012702	177574		16\$:	MOV	MMR0+2, R2	:PUT ADDRESS OF MMR0'S UPPER BYTE
5111								:PLUS 1 INTO R2
5112	042246	012705	001322			MOV	#READON+2, R5	:PUT ADDRESS OF READ ONLY BIT <13>
5113								:PLUS 2 IN R5
5114	042252	000240				NOP		:THIS IS SYNC POINT FOR SCOPING
5115	042254	114542				MOVB	-(R5), -(R2)	:LOCK UP MMR1-LOWER BYTE 375
5116								:UPPER BYTE 372-WORD 175375
5117	042256	011102				MOV	(R1), R2	:READ MMR1
5118	042260	012703	175375			MOV	#175375, R3	:PUT EXPECTED DATA IN R3
5119	042264	020203				CMP	R2, R3	:SEE IF DATA MATCHES
5120	042266	001401				BEQ	5\$:BRANCH IF DATA MATCHES
5121	042270	104013				ERROR	13	:MMR1 DID NOT TRACK PROPERLY
5122	042272	005010			5\$:	CLR	(R0)	:CLEAR MMR0
5123	042274	012737	042314	001112		MOV	#17\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 17\$
5124	042302	012737	040000	177776		MOV	#40000, @PSW	:SET SUPERVISOR MODE
5125	042310	010637	001172			MOV	SSP, @STMP0	:SAVE SUPERVISOR STACK POINTER
5126	042314	012706	177574		17\$:	MOV	MMR0+2, SSP	:PUT ADDRESS OF MMR0+2 IN SSP
5127	042320	000240				NOP		:THIS IS SYNC POINT FOR SCOPING
5128	042322	012746	040000			MOV	#40000, -(SSP)	:LOCK UP MMR1-LOWER BYTE 027
5129								:UPPER BYTE 366-WORD 173027
5130	042326	011102				MOV	(R1), R2	:READ MMR1
5131	042330	012703	173027			MOV	#173027, R3	:PUT EXPECTED DATA IN R3
5132	042334	020203				CMP	R2, R3	:SEE IF DATA MATCHES
5133	042336	001401				BEQ	6\$:BRANCH IF DATA MATCHES
5134	042340	104013				ERROR	13	:MMR1 DID NOT TRACK PROPERLY
5135	042342	013706	001172		6\$:	MOV	STMP0, SSP	:RESTORE THE SUPERVISOR STACK POINTER
5136	042346	005010				CLR	(R0)	:CLEAR MMR0
5137	042350	012737	042370	001112		MOV	#18\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 18\$
5138	042356	012737	140000	177776		MOV	#140000, @PSW	:SET USER MODE
5139	042364	010637	001172			MOV	USP, @STMP0	:SAVE USER STACK POINTER
5140	042370	012706	177572		18\$:	MOV	MMR0, USP	:PUT ADDRESS OF MMR0 IN USP
5141	042374	000240				NOP		:THIS IS SYNC POINT FOR SCOPING
5142	042376	012726	100000			MOV	#100000, (USP)+	:LOCK UP MMR1-LOWER BYTE 027
5143								:UPPER BYTE 026-WORD 013027
5144	042402	011102				MOV	(R1), R2	:READ MMR1
5145	042404	012703	013027			MOV	#013027, R3	:PUT EXPECTED DATA IN R3
5146	042410	020203				CMP	R2, R3	:SEE IF DATA MATCHES
5147	042412	001401				BEQ	7\$:BRANCH IF DATA IS GOOD
5148	042414	104013				ERROR	13	:MMR1 DID NOT TRACK PROPERLY
5149	042416	013706	001172		7\$:	MOV	@STMP0, USP	:RESTORE USER STACK POINTER
5150	042422	005037	177776			CLR	@PSW	:GO BACK TO KERNEL MODE
5151	042426	005037	177572			CLR	@MMR0	:LET ALL M.M.R'S TRACK

5152 042432 012737 042016 001112 MOV #20\$,SLPERR ;SET LOOP POINTER TO START OF TEST

5153
5154
5155
5156
5157
5158
5159
5160
5161
5162
5163
5164
5165
5166
5167
5168
5169
5170
5171
5172
5173
5174
5175
5176
5177
5178
5179
5180
5181
5182
5183
5184
5185
5186
5187
5188
5189
5190
5191
5192
5193
5194
5195
5196
5197
5198
5199
5200
5201
5202
5203
5204
5205
5206
5207

```

*****
*TEST 13      BIT TEST OF MEMORY MANAGEMENT REGISTER 2
*
*   HERE MMR2 WILL BE READ SEVERAL DIFFERENT TIMES TO
*   SEE THAT IT IS TRACKING PROPERLY.  THEN IT WILL BE
*   LOCKED UP AND READ IT TO SEE THAT IT DOES NOT CHANGE AFTER
*   IT IS ONCE LOCKED UP.  NOT ALL BITS WILL BE TESTED IN THIS
*   TEST BUT A LATER TEST RUNS A COUNT PATTERN THROUGH MMR2.  THIS
*   CANNOT BE DONE HERE SINCE IT REQUIRES THAT THE ABORT LOGIC IS
*   FUNCTIONING PROPERLY.
*****
    
```

```

*****
TST13:
SCOPE
MOV #TST14,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
20$: MOV MMR0,R0 ;PUT ADDRESS OF MMR0 IN R0
MOV MMR2,R1 ;PUT ADDRESS OF MMR2 IN R1
MOV #11$,SLPERR ;SET LOOP ON ERROR POINTER TO 11$
CLR @MMR0 ;MAKE SURE THAT MMR0 IS CLEAR
11$: NOP ;THIS A IS SYNC POINT FOR SCOPING
21$: MOV (R1),R2 ;READ MMR2 TO R2
MOV #21$,R3 ;PUT ADDRESS OF LAST INST IN R3
CMP R2,R3 ;SEE IF MMR2 HELD CORRECT ADDRESS
BEQ 1$ ;BRANCH IF DATA MATCHES
ERROR 14 ;MMR2 DID NOT TRACK PROPERLY
1$: MOV #12$,SLPERR ;SET LOOP ON ERROR POINTER TO 12$
12$: NOP ;THIS A IS SYNC POINT FOR SCOPING
22$: MOV @MMR2,R2 ;READ MMR2 TO R2
MOV #22$,R3 ;PUT ADDRESS OF LAST INST IN R3
CMP R2,R3 ;SEE IF MMR2 HELD CORRECT ADDRESS
BEQ 2$ ;BRANCH IF DATA MATCHES
ERROR 14 ;MMR2 DID NOT TRACK PROPERLY
2$: MOV #13$,SLPERR ;SET LOOP ON ERROR POINTER TO 13$
13$: NOP ;THIS A IS SYNC POINT FOR SCOPING
23$: MOV #40000,(R0) ;LOCK UP MMR1 AND MMR2
MOV (R1),R2 ;READ MMR2 INTO R2
MOV #23$,R3 ;PUT LOCKING INST'S ADDRESS IN R3
CMP R2,R3 ;SEE IF MMR2 HOLDS RIGHT ADDRESS
BEQ 3$ ;BRANCH IF DATA MATCHES
ERROR 14 ;MMR2 DID NOT TRACK PROPERLY
3$: MOV #14$,SLPERR ;SET LOOP ON ERROR POINTER TO 14$
14$: RESET ;ISSUE INIT TO CLEAR MMR1 & 2
15$: NOP ;THIS A IS SYNC POINT FOR SCOPING
25$: MOV (R1),R2 ;READ MMR2 INTO R2
MOV #25$,R3 ;PUT ADDRESS OF LAST INST IN R3
CMP R2,R3 ;SEE IF MMR2 IS STILL TRACKING
BEQ TST14 ;GO TO NEXT TEST IF IT IS
ERROR 14 ;MMR2 DID NOT TRACK PROPERLY
    
```

```

*****
*TEST 14      BIT TEST OF MEMORY MANAGEMENT REGISTER 3
*
    
```

```

5208      : *      THIS TEST SETS AND CLEARS BITS <05:04> AND <02:00> OF MMR3
5209      : *      IT DOES NOT TEST THAT THE BITS FUNCTION PROPERLY SINCE THAT
5210      : *      REQUIRES MORE LOGIC. BUT IT DOES TEST THE REGISTER AND THE
5211      : *      DATA PATH TO AND FROM THE REGISTER. THE PROPER FUNCTIONING
5212      : *      OF THESE BITS IS TESTED LATER IN SEVERAL TESTS.
5213      : *
5214      : *****
5215      TST14:
5216      042614 000004      SCOPE
5217      042616 012737 042760 001316      MOV      #TST15,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
5218      : *
5219      042624 012700 172516 20$:      MOV      #MMR3,R0      ;TEST FOR ESCAPE ON PARITY ERRORS
5220      042630 012701 000001      MOV      #1,R1      ;PUT ADDRESS OF MMR3 IN R0
5221      042634 005010      CLR      (R0)      ;SET BIT TO FLOAT THRU MMR3
5222      042636 011002      MOV      (R0),R2      ;CLEAR MMR3
5223      042640 001401      BEQ      5$      ;READ MMR3 INTO R2
5224      042642 104015      ERROR   15      ;BRANCH IF ZERO
5225      042644 012703 000006 5$:      MOV      #6,R3      ;CAN'T CLEAR MMR3
5226      042650 012737 042656 001112 1$:      MOV      #1$,SLPERR      ;SET UP LOOP COUNT
5227      042656 000240      NOP      ;SET LOOP ON ERROR POINTER TO 1$
5228      042660 010110      MOV      R1,(R0)      ;THIS A IS SYNC POINT FOR SCOPING
5229      042662 011002      MOV      (R0),R2      ;LOAD MMR3
5230      042664 020102      CMP      R1,R2      ;READ MMR3 INTO R2
5231      042666 001404      BEQ      2$      ;DOES DATA MATCH PATTERN
5232      042670 032701 000010      BIT      #BIT3,R1      ;BRANCH IF IT MATCHES
5233      042674 001001      BNE      2$      ;WAS THIS BIT 3
5234      042676 104016      ERROR   16      ;BRANCH IF IT WAS
5235      042700 006301 2$:      ASL      R1      ;MMR3 HELD WRONG DATA
5236      042702 077313      SOB      R3,1$      ;LEFT SHIFT DATA PATTERN
5237      042704 012701 000067      MOV      #6,R1      ;BRANCH BACK IF R3 NOT 0
5238      042710 012737 042716 001112 12$:      MOV      #12$,SLPERR      ;PUT DATA PATTERN IN R1
5239      042716 000240      NOP      ;SET LOOP ON ERROR POINTER TO 12$
5240      042720 010110      MOV      R1,(R0)      ;THIS A IS SYNC POINT FOR SCOPING
5241      042722 011002      MOV      (R0),R2      ;TRY TO SET ALL BITS IN MMR3
5242      042724 020102      CMP      R1,R2      ;READ MMR3 INTO R2
5243      042726 001401      BEQ      3$      ;SEE IF ALL BITS GOT SET
5244      042730 104016      ERROR   16      ;BRANCH IF ALL BITS WERE SET
5245      : *
5246      042732 012737 042740 001112 3$:      MOV      #13$,SLPERR      ;MMR3 HELD WRONG DATA
5247      042740 000240 13$:      NOP      ;SET LOOP ON ERROR POINTER TO 13$
5248      042742 000005      RESET   ;THIS A IS SYNC POINT FOR SCOPING
5249      042744 011001      MOV      (R0),R1      ;ISSUE 'INIT'
5250      042746 001401      BEQ      4$      ;READ MMR3 INTO R2
5251      042750 104015      ERROR   15      ;BRANCH IF MMR3 INTO R2
5252      042752 012737 042624 001112 4$:      MOV      #20$,SLPERR      ;CAN'T CLEAR MMR3
5253      : *
5254      .SBTTL ***** ENTRY POINT 3 --- STARTING ADDRESS 210 *****
5255      .SBTTL ***** PAGE ADDRESS AND DESCRIPTOR REGISTER TESTS *****
5256      : *
5257      : *      THIS GROUP OF TESTS IS USED TO CHECK ALL THE PAR'S AND
5258      : *      PDR'S; THEIR ADDRESS DECODING ON DIRECT REFERENCES, THEIR
5259      : *      READ/WRITE BITS, AND DUAL ADDRESSING WITHIN A GROUP OR BETWEEN
5260      : *      TWO GROUPS.
5261      : *
5262      : *
5263      : *
    
```

5264
5265
5266
5267
5268
5269
5270
5271
5272
5273
5274
5275
5276
5277
5278
5279
5280
5281
5282 042760
5283 042760 000004
5284 042762 012737 043114 001316
5285
5286 042770
5287 042770 012737 043020 001110
5288 042776 012737 043020 001112
5289 043004 012737 000015 001102
5290 043012 013737 001102 177570
5291 043020 004737 031346
5292 043024 012737 031404 000004
5293 043032 012737 043044 001112
5294 043040 012700 172340
5295 043044 000240
5296 043046 011001
5297 043050 062700 000002
5298 043054 022700 172376
5299 043060 103371
5300 043062 012737 043020 001112
5301 043070 012737 031670 000004
5302 043076 005737 001302
5303 043102 001404
5304 043104 013737 001302 001204
5305 043112 104017
5306
5307
5308
5309
5310
5311
5312
5313
5314
5315
5316
5317
5318
5319 043114

```
.SBTTL TEST THAT ALL P.A.R.'S & P.D.R.'S RESPOND
* THESE NEXT SIX (6) TESTS VERIFY THAT THERE ARE 6 GROUPS OF
* 16 CONTIGUOUS ADDRESSES IN THE I/O PAGE THAT WILL RESPOND
* WITHOUT TIMING OUT. AT THIS POINT THE TEST IS NOT CONCERNED
* WITH EXACTLY WHAT IS RESPONDING, JUST THAT SOMETHING RESPONDS.
*****
*TEST 15 READ ALL KERNEL PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT
*
* THIS TEST DOES A READ FROM ALL THE KERNEL PAGE ADDRESS REGISTERS
* 'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE TEST AND
* IF ANY OF THE 16 KERNEL ADDRESS REGISTERS TIME OUT THEIR I/O
* PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END OF THE TEST A
* SUMMARY OF THE ERRORS IS GIVEN AND 'ERRVEC' IS RE-SET TO 'CPUER'.
*
* ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT'
*****
TST15:
SCOPE
MOV #TST16,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
ENTPT3:
MOV #20$,SLPADR ;SET LOOP ADDRESS POINTER TO 20$
MOV #20$,SLPERR ;SET LOOP ON ERROR POINTER TO 20$
MOV #15,$TSTNM ;LOAD TEST NUMBER INTO MEMORY
MOV STSTNM,DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
20$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
MOV #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
MOV #KIPAR0,R0 ;PUT ADDRESS OF FIRST PAR IN R0
1$: NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV (R0),R1 ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
ADD #2,R0 ;POINT TO NEXT REGISTER
CMP #KDPAR7,R0 ;SEE IF KDPAR7 HAS BEEN TRIED
BHS 1$ ;BRANCH IF NOT DONE
MOV #20$,SLPERR ;PUT LOOP ON ERROR POINTER TO START OF TEST
MOV #CPUER,ERRVEC ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
TST ERRCNT ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
BEQ TST16 ;BRANCH TO NEXT TEST IF NO ERRORS
MOV ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
ERROR 17 ;SUMMARY OF PAR ERRORS
*****
*TEST 16 READ ALL SUPERVISOR PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT
*
* THIS TEST DOES A READ FROM ALL THE SUPERVISOR PAGE ADDRESS
* REGISTERS. 'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
* TEST, AND IF ANY OF THE 16 SUPERVISOR ADDRESS REGISTERS TIME OUT
* THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END
* OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
* 'CPUER'.
* ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT'
*****
TST16:
```

```

5320 043114 000004          SCOPE
5321 043116 012737 043220 001316 MOV      #TST17,NXTTST ;SAVE STARTING ADDRESS OF NEXT
5322                                     ;TEST FOR ESCAPE ON PARITY ERRORS
5323 043124 004737 031346          20$: JSR      PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
5324 043130 012737 031404 000004 MOV      #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
5325 043136 012737 043150 001112 MOV      #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
5326 043144 012700 172240          MOV      #SIPARO,R0 ;PUT ADDRESS OF FIRST PAR IN R0
5327 043150 000240          1$:  NOP      ;THIS IS A SYNC POINT FOR SCOPING
5328 043152 011001          MOV      (R0),R1 ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
5329 043154 062700 000002 ADD      #2,R0 ;POINT TO NEXT REGISTER
5330 043160 022700 172276          CMP      #SDPAR7,R0 ;SEE IF SDPAR7 HAS BEEN TRIED
5331 043164 103371          BHS     1$ ;BRANCH IF NOT DONE
5332 043166 012737 043124 001112 MOV      #20$,SLPERR ;PUT LOOP ON ERROR POINTER TO START OF TEST
5333 043174 012737 031670 000004 MOV      #CPUER,ERRVEC ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
5334 043202 005737 001302          TST     ERRCNT ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
5335 043206 001404          BEQ     TST17 ;BRANCH TO NEXT TEST IF NO ERRORS
5336 043210 013737 001302 001204 MOV      ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
5337 043216 104017          ERROR   17 ;SUMMARY OF PAR ERRORS
5338
5339

```

 *TEST 17 READ ALL USER PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT

THIS TEST DOES A READ FROM ALL THE USER PAGE ADDRESS
 REGISTERS. 'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
 TEST, AND IF ANY OF THE 16 USER ADDRESS REGISTERS TIME OUT
 THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END
 OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
 'CPUER'.
 ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT'

```

5350
5351 043220          TST17:
5352 043220 000004          SCOPE
5353 043222 012737 043324 001316 MOV      #TST20,NXTTST ;SAVE STARTING ADDRESS OF NEXT
5354                                     ;TEST FOR ESCAPE ON PARITY ERRORS
5355 043230 004737 031346          20$: JSR      PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
5356 043234 012737 031404 000004 MOV      #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
5357 043242 012737 043254 001112 MOV      #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
5358 043250 012700 177640          MOV      #UIPARO,R0 ;PUT ADDRESS OF FIRST PAR IN R0
5359 043254 000240          1$:  NOP      ;THIS IS A SYNC POINT FOR SCOPING
5360 043256 011001          MOV      (R0),R1 ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
5361 043260 062700 000002 ADD      #2,R0 ;POINT TO NEXT REGISTER
5362 043264 022700 177676          CMP      #UDPAR7,R0 ;SEE IF UDPAR7 HAS BEEN TRIED
5363 043270 103371          BHS     1$ ;BRANCH IF NOT DONE
5364 043272 012737 043230 001112 MOV      #20$,SLPERR ;PUT LOOP ON ERROR POINTER TO START OF TEST
5365 043300 012737 031670 000004 MOV      #CPUER,ERRVEC ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
5366 043306 005737 001302          TST     ERRCNT ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
5367 043312 001404          BEQ     TST20 ;BRANCH TO NEXT TEST IF NO ERRORS
5368 043314 013737 001302 001204 MOV      ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
5369 043322 104017          ERROR   17 ;SUMMARY OF PAR ERRORS
5370
5371

```

 *TEST 20 READ ALL KERNEL PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT

THIS TEST DOES A READ FROM ALL THE KERNEL PAGE DESCRIPTOR
 REGISTERS. 'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE

5372
 5373
 5374
 5375

5376
5377
5378
5379
5380
5381
5382
5383
5384
5385
5386
5387
5388
5389
5390
5391
5392
5393
5394
5395
5396
5397
5398
5399
5400
5401
5402
5403
5404
5405
5406
5407
5408
5409
5410
5411
5412
5413
5414
5415
5416
5417
5418
5419
5420
5421
5422
5423
5424
5425
5426
5427
5428
5429
5430
5431

```

:: TEST, AND IF ANY OF THE 16 KERNEL DESCRIPTOR REGISTERS TIME OUT
:: THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END
:: OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
:: 'CPUER'.
:: ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT'
::*****
TST20:

```

```

SCOPE
MOV #TST21,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
20$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
MOV #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
MOV #KDPDR0,R0 ;PUT ADDRESS OF FIRST PDR IN R0
1$: NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV (R0),R1 ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
ADD #2,R0 ;POINT TO NEXT REGISTER
CMP #KDPDR7,R0 ;SEE IF KDPDR7 HAS BEEN TRIED
BHS 1$ ;BRANCH IF NOT DONE
MOV #20$,SLPERR ;PUT LOOP ON ERROR POINTER TO START OF TEST
MOV #CPUER,ERRVEC ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
TST ERRCNT ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
BEQ TST21 ;BRANCH TO NEXT TEST IF NO ERRORS
MOV ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
ERROR 17 ;SUMMARY OF PDR ERRORS

```

```

::*****
::TEST 21 READ ALL SUPERVISOR PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT
::
:: THIS TEST DOES A READ FROM ALL THE SUPERVISOR PAGE DESCRIPTOR
:: REGISTERS. 'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
:: TEST, AND IF ANY OF THE 16 SUPERVISOR DESCRIPTOR REGISTERS TIME OUT
:: THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END
:: OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
:: 'CPUER'.
:: ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT'
::*****
TST21:

```

```

SCOPE
MOV #TST22,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
20$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
MOV #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
MOV #SDPDR0,R0 ;PUT ADDRESS OF FIRST PDR IN R0
1$: NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV (R0),R1 ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
ADD #2,R0 ;POINT TO NEXT REGISTER
CMP #SDPDR7,R0 ;SEE IF SDPDR7 HAS BEEN TRIED
BHS 1$ ;BRANCH IF NOT DONE
MOV #20$,SLPERR ;PUT LOOP ON ERROR POINTER TO START OF TEST
MOV #CPUER,ERRVEC ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
TST ERRCNT ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
BEQ TST22 ;BRANCH TO NEXT TEST IF NO ERRORS

```

5432 043524 013737 001302 001204
 5433 043532 104017
 5434
 5435
 5436
 5437
 5438
 5439
 5440
 5441
 5442
 5443
 5444
 5445
 5446
 5447 043534
 5448 043534 000004
 5449 043536 012737 043640 001316
 5450
 5451 043544 004737 031346 20\$:
 5452 043550 012737 031404 000004
 5453 043556 012737 043570 001112
 5454 043564 012700 177600
 5455 043570 000240 1\$:
 5456 043572 011001
 5457 043574 062700 000002
 5458 043600 022700 177636
 5459 043604 103371
 5460 043606 012737 043544 001112
 5461 043614 012737 031670 000004
 5462 043622 005737 001302
 5463 043626 001404
 5464 043630 013737 001302 001204
 5465 043636 104017
 5466
 5467
 5468
 5469
 5470
 5471
 5472
 5473
 5474
 5475
 5476
 5477
 5478
 5479
 5480
 5481
 5482
 5483
 5484
 5485
 5486
 5487

```

MOV      ERRCNT,$TMP5      ;SAVE NUMBER OF ERRORS FOR TIMEOUT
ERROR    17                ;SUMMARY OF PDR ERRORS

*****
*TEST 22      READ ALL USER PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT
:
:      THIS TEST DOES A READ FROM ALL THE USER PAGE DESCRIPTOR
:      REGISTERS. 'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
:      TEST, AND IF ANY OF THE 16 USER DESCRIPTOR REGISTERS TIME OUT
:      THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END
:      OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
:      'CPUER'.
:      ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT'
*****
TST22:
SCOPE
MOV      #TST23,NXTTST    ;SAVE STARTING ADDRESS OF NEXT
:      ;TEST FOR ESCAPE ON PARITY ERRORS
20$:    JSR      PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
MOV      #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
MOV      #1$,SLPERR      ;SET LOOP ON ERROR POINTER TO 1$
MOV      #UIPDRO,R0      ;PUT ADDRESS OF FIRST PDR IN R0
1$:     NOP
MOV      (R0),R1         ;THIS IS A SYNC POINT FOR SCOPING
ADD      #2,R0           ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
CMP      #UDPDR7,R0     ;POINT TO NEXT REGISTER
BHS      1$             ;SEE IF UDPDR7 HAS BEEN TRIED
MOV      #20$,SLPERR    ;BRANCH IF NOT DONE
MOV      #CPUER,ERRVEC  ;PUT LOOP ON ERROR POINTER TO START OF TEST
TST      ERRCNT         ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
BEQ      TST23         ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
MOV      ERRCNT,$TMP5   ;BRANCH TO NEXT TEST IF NO ERRORS
ERROR    17            ;SAVE NUMBER OF ERRORS FOR TIMEOUT
:      ;SUMMARY OF PDR ERRORS

.SBTTL      TEST FOR DUAL ADDRESSING ON LOAD WITHIN GROUPS
:      *      THESE NEXT SIX (6) TESTS WILL CHECK FOR DUAL ADDRESSING WITHIN A
:      *      GROUP OF PAR'S OR PDR'S. FIRST ALL OF THE REGISTERS IN A GROUP
:      *      ARE CLEARED AND READ TO SEE THAT THEY CAN EACH HOLD ZEROES, AND
:      *      THAT THE DATA PATH DOES NOT HAVE A BIT STUCK AT ONE.
:      *      THEN ONE REGISTER AT A TIME, WITHIN THAT GROUP, IS LOADED
:      *      WITH A NEGATIVE ONE WHILE ALL REGISTERS ARE READ TO SEE
:      *      THAT ONLY THE REGISTER UNDER TEST WAS NOT ZERO.
*****
*TEST 23      DUAL ADDRESS KERNEL PAGE ADDRESS REGISTERS, ON LOADING
:
:      THIS TEST FIRST CLEARS ALL THE KERNEL PAGE ADDRESS REGISTERS,
:      AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING
:      WITH I-SPACE ADDRESS REGISTER ZERO, ONE REGISTER AT A TIME
:      IS LOADED WITH A NEGATIVE ONE. ALL KERNEL ADDRESS REGISTERS
:      ARE NOW READ TO SEE THAT ONLY THE REGISTER UNDER TEST IS NON-
:      ZERO. INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM
:      IS GIVEN AT THE END OF THIS TEST.
:
:      ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
    
```



```

5488          : * 'DUALADR'.
5489          : *****
5490 043640    TST23:
5491 043640    000004    SCOPE
5492 043642    012737    044050    001316    MOV    #TST24,NXTTST    ;SAVE STARTING ADDRESS OF NEXT
5493          :                               ;TEST FOR ESCAPE ON PARITY ERRORS
5494 043650    004737    031346    20$:    JSR    PC,CLEANUP    ;INITIALIZE ERROR LOCATIONS
5495 043654    012737    043702    001112    MOV    #1$,SLPERR    ;SET LOOP ON ERROR POINTER TO 1$
5496 043662    012705    172340    MOV    #KIPAR0,R5    ;PUT ADDRESS OF FIRST PAR IN R5
5497 043666    004737    031330    JSR    PC,CLRREG    ;CLEAR 16 REGISTERS POINTED TO BY R5
5498 043672    012700    172340    MOV    #KIPAR0,R0    ;PUT ADDRESS OF FIRST PAR IN R0
5499 043676    012701    000020    MOV    #20,R1    ;BRANCH COUNT IS 16 DECIMAL
5500 043702    000240    1$:    NOP    ;THIS IS A SYNC POINT FOR SCOPING
5501 043704    011002    MOV    (R0),R2    ;READ PAR TO R2
5502 043706    001401    BEQ    2$    ;BRANCH IF PAR IS 0
5503 043710    104020    ERROR    20    ;PAR NOT ZERO
5504 043712    062700    000002    2$:    ADD    #2,R0    ;POINT TO NEXT REGISTER
5505 043716    077107    SOB    R1,1$    ;BRANCH BACK TO 1$ 15 TIMES
5506          :
5507          :
5508          : NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
5509          : REGISTER AND READING THE NEXT TO SEE ONLY THAT REGISTER
5510          : IS NON-ZERO. (DROPPED BITS WILL BE FOUND IN NEXT TEST.)
5511          :
5511 043720    012737    043732    001112    MOV    #3$,SLPERR    ;SET LOOP ON ERROR POINTER TO 3$
5512 043726    012700    172340    MOV    #KIPAR0,R0    ;PUT ADDRESS OF FIRST PAR IN R0
5513 043732    012705    172340    3$:    MOV    #KIPAR0,R5    ;LOAD STARTING ADDRESS INTO R5
5514 043736    004737    031330    JSR    PC,CLRREG    ;CLEAR 16 REGISTERS POINTED TO BY R5
5515 043742    012701    172376    MOV    #KDPAR7,R1    ;PUT KDPAR7 ADDRESS IN R1
5516 043746    000240    NOP    ;THIS IS A SYNC POINT FOR SCOPING
5517 043750    012710    177777    MOV    #-1,(R0)    ;LOAD REGISTER UNDER TEST
5518 043754    005037    001172    4$:    CLR    $TMP0    ;FLAG TO INDICATE THERE WAS A MATCH
5519 043760    011102    MOV    (R1),R2    ;READ ALL REGISTERS
5520 043762    001406    BEQ    6$    ;BRANCH IF REGISTER IS 0
5521 043764    020001    CMP    R0,R1    ;IS ADDRESS OF NON-ZERO REGISTER SAME
5522          :                               ;AS THAT OF REGISTER UNDER TEST
5523 043766    001402    BEQ    5$    ;BRANCH IF ADDRESSES MATCH
5524 043770    004737    031524    JSR    PC,DUALADR    ;LOG AND REPORT ERRORS
5525 043774    005237    001172    5$:    INC    $TMP0    ;SET FLAG WHEN ADDRESSES MATCH
5526 044000    162701    000002    6$:    SUB    #2,R1    ;POINT TO NEXT REGISTER
5527 044004    022701    172340    CMP    #KIPAR0,R1    ;SEE IF ALL REGISTERS HAVE BEEN READ
5528 044010    101761    BLOS    4$    ;BRANCH IF MORE TO READ
5529 044012    062700    000002    ADD    #2,R0    ;NOW LOAD NEXT REGISTER
5530 044016    022700    172376    CMP    #KDPAR7,R0    ;SEE IF THERE ARE MORE REGISTERS TO TEST
5531 044022    103343    BHIS    3$    ;BRANCH IF MORE REGISTERS TO TEST
5532 044024    012737    043650    001112    MOV    #20$,SLPERR    ;SET LOOP ON ERROR POINTER TO START OF TEST
5533 044032    005737    001302    TST    ERRCNT    ;SEE IF THERE WERE ANY ERRORS
5534 044036    001404    BEQ    TST24    ;BRANCH TO NEXT TEST IF NO ERRORS
5535 044040    013737    001302    001204    MOV    ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR PAR OUT
5536 044046    104021    ERROR    21    ;SUMMARY OF DUAL ADDRESSING ERRORS
5537
    
```

```
5538 .....
5539 :*****
5540 :*TEST 24      DUAL ADDRESS SUPERVISOR PAGE ADDRESS REGISTERS, ON LOADING
5541 :
5542 :           THIS TEST FIRST CLEARS ALL THE SUPERVISOR PAGE ADDRESS
5543 :           AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING WITH
5544 :           I-SPACE ADDRESS REGISTER ZERO, ONE REGISTER AT A TIME IS
5545 :           LOADED WITH A NEGATIVE ONE. ALL SUPERVISOR ADDRESS REGISTERS
5546 :           ARE NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
5547 :           INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN AT
5548 :           THE END OF THIS TEST.
5549 :
5550 :           ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
5551 :           'DUALADR'.
5552 :*****
5553 TST24:
5554 044050 000004
5555 044052 012737 044260 001316      SCOPE
5556                                     MOV      #TST25,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
5557 044060 004737 031346      20$:   JSR      PC,CLEANUP      ;TEST FOR ESCAPE ON PARITY ERRORS
5558 044064 012737 044112 001112      MOV      #1$,SLPERR      ;INITIALIZE ERROR LOCATIONS
5559 044072 012705 172240      MOV      #SIPARO,R5      ;SET LOOP ON ERROR POINTER TO 1$
5560 044076 004737 031330      JSR      PC,CLRREG      ;PUT ADDRESS OF FIRST PAR IN R5
5561 044102 012700 172240      MOV      #SIPARO,R0      ;CLEAR 16 REGISTERS POINTED TO BY R5
5562 044106 012701 000020      MOV      #20,R1          ;PUT ADDRESS OF FIRST PAR IN R0
5563 044112 000240      1$:   NOP                      ;BRANCH COUNT IS 16 DECIMAL
5564 044114 011002      MOV      (R0),R2        ;THIS IS A SYNC POINT FOR SCOPING
5565 044116 001401      BEQ      2$            ;READ PAR TO R2
5566 044120 104020      ERROR   20            ;BRANCH IF PAR IS 0
5567 044122 062700 000002      2$:   ADD      #2,R0        ;PAR NOT ZERO
5568 044126 077107      SOB     R1,1$         ;POINT TO NEXT REGISTER
5569                                     ;BRANCH BACK TO 1$ 15 TIMES
5570 :
5571 :           NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
5572 :           REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
5573 :           IS NON-ZERO. (DROPPED BITS WILL BE FOUND IN NEXT TEST.)
5574 044130 012737 044142 001112      MOV      #3$,SLPERR      ;SET LOOP ON ERROR POINTER TO 3$
5575 044136 012700 172240      MOV      #SIPARO,R0      ;PUT ADDRESS OF FIRST PAR IS R0
5576 044142 012705 172240      3$:   MOV      #SIPARO,R5      ;LOAD STARTING ADDRESS INTO R5
5577 044146 004737 031330      JSR      PC,CLRREG      ;CLEAR 16 REGISTERS POINTED TO BY R5
5578 044152 012701 172276      MOV      #SDPAR7,R1      ;PUT SDPAR7 ADDRESS IN R1
5579 044156 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
5580 044160 012710 177777      MOV      #-1,(R0)        ;LOAD REGISTER UNDER TEST
5581 044164 005037 001172      4$:   CLR      $TMP0        ;FLAG TO INDICATE THERE WAS A MATCH
5582 044170 011102      MOV      (R1),R2        ;READ ALL REGISTERS
5583 044172 001406      BEQ      6$            ;BRANCH IF REGISTER IS 0
5584 044174 020001      CMP     R0,R1          ;IS ADDRESS OF NON-ZERO REGISTER SAME
5585                                     ;AS THAT OF REGISTER UNDER TEST
5586 044176 001402      BEQ     5$            ;BRANCH IF ADDRESSES MATCH
5587 044200 004737 031524      JSR     PC,DUALADR      ;LOG AND REPORT ERRORS
5588 044204 005237 001172      5$:   INC     $TMP0        ;SET FLAG WHEN ADDRESSES MATCH
5589 044210 162701 000002      6$:   SUB     #2,R1          ;POINT TO NEXT REGISTER
5590 044214 022701 172240      CMP     #SIPARO,R1      ;SEE IF ALL REGISTERS HAVE BEEN READ
5591 044220 101761      BLOS   4$            ;BRANCH IF MORE TO READ
5592 044222 062700 000002      ADD     #2,R0          ;NOW LOAD NEXT REGISTER
5593 044226 022700 172276      CMP     #SDPAR7,R0      ;SEE IF THERE ARE MORE REGISTERS TO TEST
```

```

5594 044232 103343          BHS 3$          :BRANCH IF MORE REGISTERS TO TEST
5595 044234 012737 044060 001112  MOV #20$, $LPERR :SET LOOP ON ERROR POINTER TO START OF TEST
5596 044242 005737 001302          TST ERRCNT      :SEE IF THERE WERE ANY ERRORS
5597 044246 001404          BEQ TST25       :BRANCH TO NEXT TEST IF NO ERRORS
5598 044250 013737 001302 001204  MOV ERRCNT, $TMP5 :SAVE NUMBER OF ERRORS FOR PAR OUT
5599 044256 104021          ERROR 21        :SUMMARY OF DUAL ADDRESSING ERRORS
    
```

 :TEST 25 DUAL ADDRESS USER PAGE ADDRESS REGISTERS, ON LOADING

THIS TEST FIRST CLEARS ALL THE USER PAGE ADDRESS 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.

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

 TST25:

```

5615 044260          SCOPE
5616 044260 000004          MOV #TST26, NXTTST :SAVE STARTING ADDRESS OF NEXT
5617 044262 C12737 044470 001316  MOV #20$, $LPERR :TEST FOR ESCAPE ON PARITY ERRORS
5618          JSR PC, CLEANUP :INITIALIZE ERROR LOCATIONS
5619 044270 004737 031346 20$:  MOV #1$, $LPERR :SET LOOP ON ERROR POINTER TO 1$
5620 044274 012737 044322 001112  MOV #UIPAR0, R5 :PUT ADDRESS OF FIRST PAR IN R5
5621 044302 012705 177640          JSR PC, CLRREG :CLEAR 16 REGISTERS POINTED TO BY R5
5622 044306 004737 031330          MOV #UIPAR0, R0 :PUT ADDRESS OF FIRST PAR IN R0
5623 044312 012700 177640          MOV #20, R1 :BRANCH COUNT IS 16 DECIMAL
5624 044316 012701 000020          NOP :THIS IS A SYNC POINT FOR SCOPING
5625 044322 000240 1$:  MOV (R0), R2 :READ PAR TO R2
5626 044324 011002          BEQ 2$ :BRANCH IF PAR IS 0
5627 044326 001401          ERROR 20 :PAR NOT ZERO
5628 044330 104020          ADD #2, R0 :POINT TO NEXT REGISTER
5629 044332 062700 000002 2$:  SOB R1, 1$ :BRANCH BACK TO 1$ 15 TIMES
5630 044336 077107
    
```

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

```

5631
5632
5633
5634
5635
5636 044340 012737 044352 001112  MOV #3$, $LPERR :SET LOOP ON ERROR POINTER TO 3$
5637 044346 012700 177640          MOV #UIPAR0, R0 :PUT ADDRESS OF FIRST PAR IN R0
5638 044352 012705 177640 3$:  MOV #UIPAR0, R5 :LOAD STARTING ADDRESS INTO R5
5639 044356 004737 031330          JSR PC, CLRREG :CLEAR 16 REGISTERS POINTED TO BY R5
5640 044362 012701 177676          MOV #UDPAR7, R1 :PUT UDPAR7 ADDRESS IN R1
5641 044366 000240          NOP :THIS IS A SYNC POINT FOR SCOPING
5642 044370 012710 177777          MOV #-1, (R0) :LOAD REGISTER UNDER TEST
5643 044374 005037 001172 4$:  CLR $TMP0 :FLAG TO INDICATE THERE WAS A MATCH
5644 044400 011102          MOV (R1), R2 :READ ALL REGISTERS
5645 044402 001406          BEQ 6$ :BRANCH IF REGISTER IS 0
5646 044404 020001          CMP R0, R1 :IS ADDRESS OF NON-ZERO REGISTER SAME
5647          :AS THAT OF REGISTER UNDER TEST
5648 044406 001402          BEQ 5$ :BRANCH IF ADDRESSES MATCH
5649 044410 004737 031524          JSR PC, DUALADR :LOG AND REPORT ERRORS
    
```

5650	044414	005237	001172	5\$:	INC	\$TMP0	:SET FLAG WHEN ADDRESSES MATCH
5651	044420	162701	000002	6\$:	SUB	#2,R1	:POINT TO NEXT REGISTER
5652	044424	022701	177640		CMP	#UIPAR0,R1	:SEE IF ALL REGISTERS HAVE BEEN READ
5653	044430	101761			BLOS	4\$:BRANCH IF MORE TO READ
5654	044432	062700	000002		ADD	#2,R0	:NOW LOAD NEXT REGISTER
5655	044436	022700	177676		CMP	#UDP4P7,R0	:SEE IF THERE ARE MORE REGISTERS TO TEST
5656	044442	103343			BHIS	3\$:BRANCH IF MORE REGISTERS TO TEST
5657	044444	012737	044270	001112	MOV	#20\$,\$LPERR	:SET LOOP ON ERROR POINTER TO START OF TEST
5658	044452	005737	001302		TST	ERRCNT	:SEE IF THERE WERE ANY ERRORS
5659	044456	001404			BEQ	TST26	:BRANCH TO NEXT TEST IF NO ERRORS
5660	044460	013737	001302	001204	MOV	ERRCNT,\$TMP5	:SAVE NUMBER OF ERRORS FOR PAR OUT
5661	044466	104021			ERROR	21	:SUMMARY OF DUAL ADDRESSING ERRORS

5662
 5663
 5664
 5665
 5666
 5667
 5668
 5669
 5670
 5671
 5672
 5673
 5674
 5675
 5676

```

*****
*TEST 26          DUAL ADDRESS KERNEL PAGE DESCRIPTOR REGISTERS, ON LOADING
:
:   THIS TEST FIRST CLEARS ALL THE KERNEL PAGE DESCRIPTOR
:   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.
:
:   ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
:   'DUALADR'.
:
*****
    
```

5677	044470				TST26:		
5678	044470	000004				SCOPE	
5679	044472	012737	044700	001316		MOV	#TST27,NXTTST
5680							:SAVE STARTING ADDRESS OF NEXT
5681	044500	004737	031346		20\$:	JSR	PC,CLEANUP
5682	044504	012737	044532	001112		MOV	#1\$,\$LPERR
5683	044512	012705	172300			MOV	#KIPDR0,R5
5684	044516	004737	031330			JSR	PC,CLRREG
5685	044522	012700	172300			MOV	#KIPDR0,R0
5686	044526	012701	000020			MOV	#20,R1
5687	044532	000240			1\$:	NOP	:BRANCH COUNT IS 16 DECIMAL
5688	044534	011002				MOV	(R0),R2
5689	044536	001401				BEQ	2\$
5690	044540	104020				ERROR	20
5691	044542	062700	000002		2\$:	ADD	#2,R0
5692	044546	077107				SOB	R1,1\$

```

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

5693							
5694							
5695							
5696							
5697							
5698	044550	012737	044562	001112		MOV	#3\$,\$LPERR
5699	044556	012700	172300			MOV	#KIPDR0,R0
5700	044562	012705	172300		3\$:	MOV	#KIPDR0,R5
5701	044566	004737	031330			JSR	PC,CLRREG
5702	044572	012701	172336			MOV	#KDPDR7,R1
5703	044576	000240				NOP	:THIS IS A SYNC POINT FOR SCOPING
5704	044600	012710	177777			MOV	#-1,(R0)
5705	044604	005037	001172		4\$:	CLR	\$TMP0

```

:   :LOAD REGISTER UNDER TEST
:   :FLAG TO INDICATE THERE WAS A MATCH
    
```

```

5706 044610 011102      MOV      (R1),R2      ;READ ALL REGISTERS
5707 044612 001406      BEQ      6$          ;BRANCH IF REGISTER IS 0
5708 044614 020001      CMP      R0,R1      ;IS ADDRESS OF NON-ZERO REGISTER SAME
5709                                     ;AS THAT OF REGISTER UNDER TEST
5710 044616 001402      BEQ      5$          ;BRANCH IF ADDRESSES MATCH
5711 044620 004737 031524    JSR      PC,DUALADR  ;LOG AND REPORT ERRORS
5712 044624 005237 001172    5$: INC      $TMP0     ;SET FLAG WHEN ADDRESSES MATCH
5713 044630 162701 000002    6$: SUB      #2,R1     ;POINT TO NEXT REGISTER
5714 044634 022701 172300    CMP      #KIPDR0,R1 ;SEE IF ALL REGISTERS HAVE BEEN READ
5715 044640 101761      BLOS     4$          ;BRANCH IF MORE TO READ
5716 044642 062700 000002    ADD      #2,R0      ;NOW LOAD NEXT REGISTER
5717 044646 022700 172336    CMP      #KDPDR7,R0 ;SEE IF THERE ARE MORE REGISTERS TO TEST
5718 044652 103343      BHIS     3$          ;BRANCH IF MORE REGISTERS TO TEST
5719 044654 012737 044500 001112    MOV      #20$,$LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
5720 044662 005737 001302      TST      ERRCNT     ;SEE IF THERE WERE ANY ERRORS
5721 044666 001404      BEQ      TST27      ;BRANCH TO NEXT TEST IF NO ERRORS
5722 044670 013737 001302 001204    MOV      ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR PDR OUT
5723 044676 104021      ERROR    21         ;SUMMARY OF DUAL ADDRESSING ERRORS
5724
5725
5726
5727
5728
5729
5730
5731
5732
5733
5734
5735
5736
5737
5738

```

 *TEST 27 DUAL ADDRESS SUPERVISOR PAGE DESCRIPTOR REGISTERS, ON LOADING

THIS TEST FIRST CLEARS ALL THE SUPERVISOR PAGE DESCRIPTOR 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.

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

TST27:

```

SCOPE
5741 044702 012737 045110 001316    MOV      #TST30,NXTTST ;SAVE STARTING ADDRESS OF NEXT
5742                                     ;TEST FOR ESCAPE ON PARITY ERRORS
5743 044710 004737 031346    20$: JSR      PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
5744 044714 012737 044742 001112    MOV      #1$,$LPERR   ;SET LOOP ON ERROR POINTER TO 1$
5745 044722 012705 172200      MOV      #SIPDR0,R5   ;PUT ADDRESS OF FIRST PDR IN R5
5746 044726 004737 031330      JSR      PC,CLRREG    ;CLEAR 16 REGISTERS POINTED TO BY R5
5747 044732 012700 172200      MOV      #SIPDR0,R0   ;PUT ADDRESS OF FIRST PDR IN R0
5748 044736 012701 000020      MOV      #20,R1       ;BRANCH COUNT IS 16 DECIMAL
5749 044742 000240    1$: NOP                ;THIS IS A SYNC POINT FOR SCOPING
5750 044744 011002      MOV      (R0),R2     ;READ PDR TO R2
5751 044746 001401      BEQ      2$          ;BRANCH IF PDR IS 0
5752 044750 104020      ERROR    20         ;PDR NOT ZERO
5753 044752 062700 000002    2$: ADD      #2,R0     ;POINT TO NEXT REGISTER
5754 044756 077107      SOB     R1,1$       ;BRANCH BACK TO 1$ 15 TIMES
5755
5756
5757
5758
5759

```

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

```

5760 044760 012737 044772 001112    MOV      #3$,$LPERR   ;SET LOOP ON ERROR POINTER TO 3$
5761 044766 012700 172200      MOV      #SIPDR0,R0   ;PUT ADDRESS OF FIRST PDR IS R0

```

```

5762 044772 012705 172200 3$: MOV #SIPDR0,R5 ;LOAD STARTING ADDRESS INTO R5
5763 044776 004737 031330 JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
5764 045002 012701 172236 MOV #SDPDR7,R1 ;PUT SDPDR7 ADDRESS IN R1
5765 045006 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
5766 045010 012710 177777 MOV #-1,(R0) ;LOAD REGISTER UNDER TEST
5767 045014 005037 001172 4$: CLR $TMP0 ;FLAG TO INDICATE THERE WAS A MATCH
5768 045020 011102 MOV (R1),R2 ;READ ALL REGISTERS
5769 045022 001406 BEQ 6$ ;BRANCH IF REGISTER IS 0
5770 045024 020001 CMP R0,R1 ;IS ADDRESS OF NON-ZERO REGISTER SAME
5771 ;AS THAT OF REGISTER UNDER TEST
5772 045026 001402 BEQ 5$ ;BRANCH IF ADDRESSES MATCH
5773 045030 004737 031524 JSR PC,DUALADR ;LOG AND REPORT ERRORS
5774 045034 005237 001172 5$: INC $TMP0 ;SET FLAG WHEN ADDRESSES MATCH
5775 045040 162701 000002 6$: SUB #2,R1 ;POINT TO NEXT REGISTER
5776 045044 022701 172200 CMP #SIPDR0,R1 ;SEE IF ALL REGISTERS HAVE BEEN READ
5777 045050 101761 BLOS 4$ ;BRANCH IF MORE TO READ
5778 045052 062700 000002 ADD #2,R0 ;NOW LOAD NEXT REGISTER
5779 045056 022700 172236 CMP #SDPDR7,R0 ;SEE IF THERE ARE MORE REGISTERS TO TEST
5780 045062 103343 BHIS 3$ ;BRANCH IF MORE REGISTERS TO TEST
5781 045064 012737 044710 001112 MOV #20$,$LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
5782 045072 005737 001302 TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
5783 045076 001404 BEQ TST30 ;BRANCH TO NEXT TEST IF NO ERRORS
5784 045100 013737 001302 001204 MOV ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR PDR OUT
5785 045106 104021 ERROR 21 ;SUMMARY OF DUAL ADDRESSING ERRORS

```

 :*TEST 30 DUAL ADDRESS USER PAGE DESCRIPTOR REGISTERS, ON LOADING

THIS TEST FIRST CLEARS ALL THE USER PAGE DESCRIPTOR 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.

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

```

5800 TST30:
5801 045110
5802 045110 000004
5803 045112 012737 045320 001316 MOV #TST31,NXTTST ;SAVE STARTING ADDRESS OF NEXT
5804 ;TEST FOR ESCAPE ON PARITY ERRORS
5805 045120 004737 031346 20$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
5806 045124 012737 045152 001112 MOV #1$,$LPERR ;SET LOOP ON ERROR POINTER TO 1$
5807 045132 012705 177600 MOV #UIPDR0,R5 ;PUT ADDRESS OF FIRST PDR IN R5
5808 045136 004737 031330 JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
5809 045142 012700 177600 MOV #UIPDR0,R0 ;PUT ADDRESS OF FIRST PDR IN R0
5810 045146 012701 000020 MOV #20,R1 ;BRANCH COUNT IS 16 DECIMAL
5811 045152 000240 1$: NOP ;THIS IS A SYNC POINT FOR SCOPING
5812 045154 011002 MOV (R0),R2 ;READ PDR TO R2
5813 045156 001401 BEQ 2$ ;BRANCH IF PDR IS 0
5814 045160 104020 ERROR 20 ;PDR NOT ZERO
5815 045162 062700 000002 2$: ADD #2,R0 ;POINT TO NEXT REGISTER
5816 045166 077107 SOB R1,1$ ;BRANCH BACK TO 1$ 15 TIMES
5817

```

```

5818      :: NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
5819      :: REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
5820      :: IS NON-ZERO. (DROPPED BITS WILL BE FOUND IN NEXT TEST.)
5821      ::
5822 045170 012737 045202 001112      MOV #3$,SLPERR      ;SET LOOP ON ERROR POINTER TO 3$
5823 045176 012700 177600              MOV #UIPDRO,R0     ;PUT ADDRESS OF FIRST PDR IS R0
5824 045202 012705 177600      3$: MOV #UIPDRO,R5     ;LOAD STARTING ADDRESS INTO R5
5825 045206 004737 031330      JSR PC,CLRRREG    ;CLEAR 16 REGISTERS POINTED TO BY R5
5826 045212 012701 177636      MOV #UDPDR7,R1    ;PUT UDPDR7 ADDRESS IN R1
5827 045216 000240              NOP               ;THIS IS A SYNC POINT FOR SCOPING
5828 045220 012710 177777      MOV #-1,(R0)      ;LOAD REGISTER UNDER TEST
5829 045224 005037 001172      4$: CLR $TMP0      ;FLAG TO INDICATE THERE WAS A MATCH
5830 045230 011102              MOV (R1),R2       ;READ ALL REGISTERS
5831 045232 001406              BEQ 6$            ;BRANCH IF REGISTER IS 0
5832 045234 020001              CMP R0,R1         ;IS ADDRESS OF NON-ZERO REGISTER SAME
5833                          ;AS THAT OF REGISTER UNDER TEST
5834 045236 001402              BEQ 5$            ;BRANCH IF ADDRESSES MATCH
5835 045240 004737 031524      JSR PC,DUALADR    ;LOG AND REPORT ERRORS
5836 045244 005237 001172      5$: INC $TMP0      ;SET FLAG WHEN ADDRESSES MATCH
5837 045250 162701 000002      6$: SUB #2,R1      ;POINT TO NEXT REGISTER
5838 045254 022701 177600      CMP #UIPDRO,R1   ;SEE IF ALL REGISTERS HAVE BEEN READ
5839 045260 101761              BLOS 4$          ;BRANCH IF MORE TO READ
5840 045262 062700 000002      ADD #2,R0        ;NOW LOAD NEXT REGISTER
5841 045266 022700 177636      CMP #UDPDR7,R0   ;SEE IF THERE ARE MORE REGISTERS TO TEST
5842 045272 103343              BHIS 3$          ;BRANCH IF MORE REGISTERS TO TEST
5843 045274 012737 045120 001112      MOV #20$,SLPERR  ;SET LOOP ON ERROR POINTER TO START OF TEST
5844 045302 005737 001302      TST ERRCNT       ;SEE IF THERE WERE ANY ERRORS
5845 045306 001404              BEQ TST31        ;BRANCH TO NEXT TEST IF NO ERRORS
5846 045310 013737 001302 001204      MOV ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR PDR OUT
5847 045316 104021              ERROR 21        ;SUMMARY OF DUAL ADDRESSING ERRORS
    
```

```

5848
5849      .SBTTL          TEST FOR BAD READ/WRITE BITS IN P.A.R.'S & P.D.R.'S
5850      * THESE NEXT SIX (6) TESTS CHECK FOR BAD BITS IN THE MEMORY CHIPS
5851      * THAT MAKE UP THE PAR'S AND PDR'S. THE REGISTERS ARE LOADED WITH
5852      * ZERO AND MODIFIED BY '401' UNTIL 177777 IS REACHED IN EACH ONE
5853      * (THE NUMBER 401 WAS CHOSEN FOR FASTER RUN-TIME). THE BITS THAT
5854      * ARE NOT IMPLEMENTED OR THAT ARE NOT READ/WRITE ARE MASKED OUT
5855      * OF THE DATA COMPARE. A LOG OF MULTIPLE ERRORS IS KEPT FOR EACH
5856      * GROUP AND IT IS REPORTED AT THE CONCLUSION OF EACH TEST.
    
```

```

5857      *
5858      * *****
5859      * TEST 31      COUNT PATTERN IN KERNEL PAGE ADDRESS REGISTERS
5860      *
5861      * THIS TEST RUNS A COUNT PATTERN THRU THE KERNEL PAGE ADDRESS
5862      * REGISTERS. SINCE ALL THE BITS ARE IMPLEMENTED, NONE NEED
5863      * TO BE MASKED OUT OF THE COMPARE. IF THE COUNT PATTERN DOES
5864      * NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS, DATA
5865      * PATTERN, AND BAD DATA ARE REPORTED. AT THE END OF THE TEST A
5866      * SUMMARY OF THE ERRORS IS GIVEN.
5867      *
5868      * ALL ERRORS FOUND BY THIS TEST ARE REPORTED AND ANALYZED BY
5869      * SUBROUTINE 'PARCOUNT'.
    
```

```

5870      *
5871      * *****
5872 045320      TST31:
5873 045320 000004      SCOPE
    
```

```

5874 045322 012737 045450 001316      MOV      #TST32,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
5875                                     ;TEST FOR ESCAPE ON PARITY ERRORS
5876 045330 012737 000012 001206      MOV      #12,$TIMES        ;DO 12 ITERATIONS
5877 045336 004737 031346      JSR      PC,CLEANUP        ;INITIALIZE ERROR LOCATIONS
5878 045342 012737 045356 001112 20$:   MOV      #2$, $LPERR       ;SET LOOP ON ERROR POINTER TO 1$
5879 045350 005001                CLR      R1                ;CLEAR REGISTER TO HOLD COUNT PATTERN
5880 045352 012700 172340      1$:   MOV      #KIPARO,RO      ;PUT ADDRESS OF FIRST REGISTER IS RO
5881 045356 000240      2$:   NOP                    ;THIS IS A SYNC POINT FOR SCOPING
5882 045360 010110                MOV      R1,(R0)          ;LOAD COUNT INTO REGISTER
5883 045362 011002                MOV      (R0),R2         ;READ REGISTER BACK TO R2
5884 045364 010104                MOV      R1,R4           ;PUT PATTERN IS R4
5885 045366 020402                CMP      R4,R2           ;SEE IF DATA MATCHES PATTERN
5886 045370 001402                BEQ      3$              ;BRANCH IF DATA IS GOOD
5887 045372 004737 031600      JSR      PC,PARCOUNT     ;LOG AND REPORT COUNT ERROR
5888 045376 062700 000002      3$:   ADD      #2,RO          ;POINT TO NEXT REGISTER
5889 045402 022700 172376      CMP      #KDPAR7,RO      ;SEE IF YOU PASSED THE KDPAR7 PAR
5890 045406 103363                BHIS    2$              ;BRANCH IF MORE PAR'S TO TEST
5891 045410 022701 177777      CMP      #177777,R1      ;SEE IF COUNT HAS REACHED 177777
5892 045414 001403                BEQ      4$              ;BRANCH IF COUNT IS 177777
5893 045416 062701 000401      ADD      #401,R1         ;INCREASE COUNT PATTERN
5894 045422 000753                BR       1$              ;BRANCH TO CONTINUE TEST
5895 045424 012737 045336 001112 4$:   MOV      #20$, $LPERR     ;SET LOOP POINTER TO START OF TEST
5896 045432 005737 001302      TST      ERRCNT          ;SEE IF THERE WERE ANY ERRORS
5897 045436 001404                BEQ      TST32           ;BRANCH TO NEXT TEST IF NO ERRORS
5898 045440 013737 001302 001204      MOV      ERRCNT,$TMP5    ;SAVE NUMBER OF ERRORS FOR TYPEOUT
5899 045446 104022                ERROR  22                ;SUMMARY OF COUNT PATTERN FAILURES

```

```

5900
5901
5902 *****
5903 *TEST 32      COUNT PATTERN IN SUPERVISOR PAGE ADDRESS REGISTERS
5904
5905 THIS TEST RUNS A COUNT PATTERN THRU THE SUPERVISOR PAGE ADDRESS
5906 REGISTERS. SINCE ALL THE BITS ARE IMPLEMENTED, NONE NEED
5907 TO BE MASKED OUT OF THE COMPARE. IF THE COUNT PATTERN DOES
5908 NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS, DATA
5909 PATTERN, AND BAD DATA ARE REPORTED. AT THE END OF THE TEST A
5910 SUMMARY OF THE ERRORS IS GIVEN.
5911
5912 ALL ERRORS FOUND BY THIS TEST ARE REPORTED AND ANALYZED BY
5913 SUBROUTINE 'PARCOUNT'.
5914 *****

```

```

5915 045450
5916 045450 000004
5917 045452 012737 045600 001316      TST32: SCOPE
5918                                     MOV      #TST33,NXTTST    ;SAVE STARTING ADDRESS OF NEXT
5919                                     ;TEST FOR ESCAPE ON PARITY ERRORS
5920 045460 012737 000012 001206      MOV      #12,$TIMES        ;DO 12 ITERATIONS
5921 045466 004737 031346      JSR      PC,CLEANUP        ;INITIALIZE ERROR LOCATIONS
5922 045472 012737 045506 001112 20$:   MOV      #2$, $LPERR       ;SET LOOP ON ERROR POINTER TO 1$
5923 045500 005001                CLR      R1                ;CLEAR REGISTER TO HOLD COUNT PATTERN
5924 045502 012700 172240      1$:   MOV      #SIPARO,RO      ;PUT ADDRESS OF FIRST REGISTER IS RO
5925 045506 000240      2$:   NOP                    ;THIS IS A SYNC POINT FOR SCOPING
5926 045510 010110                MOV      R1,(R0)          ;LOAD COUNT INTO REGISTER
5927 045512 011002                MOV      (R0),R2         ;READ REGISTER BACK TO R2
5928 045514 010104                MOV      R1,R4           ;PUT PATTERN IS R4
5929 045516 020402                CMP      R4,R2           ;SEE IF DATA MATCHES PATTERN
5929 045520 001402                BEQ      3$              ;BRANCH IF DATA IS GOOD

```



```

5930 045522 004737 031600 JSR PC,PARCOUNT ;LOG AND REPORT COUNT ERROR
5931 045526 062700 000002 3$: ADD #2,R0 ;POINT TO NEXT REGISTER
5932 045532 022700 172276 CMP #SDPAR7,R0 ;SEE IF YOU PASSED THE SDPAR7 PAR
5933 045536 103363 BHIS 2$ ;BRANCH IF MORE PAR'S TO TEST
5934 045540 022701 177777 CMP #177777,R1 ;SEE IF COUNT HAS REACHED 177777
5935 045544 001403 BEQ 4$ ;BRANCH IF COUNT IS 177777
5936 045546 062701 000401 ADD #401,R1 ;INCREASE COUNT PATTERN
5937 045552 000753 BR 1$ ;BRANCH TO CONTINUE TEST
5938 045554 012737 045466 001112 4$: MOV #20$,$LPERR ;SET LOOP POINTER TO START OF TEST
5939 045562 005737 001302 TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
5940 045566 001404 BEQ TST33 ;BRANCH TO NEXT TEST IF NO ERRORS
5941 045570 013737 001302 001204 MOV ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
5942 045576 104022 ERROR 22 ;SUMMARY OF COUNT PATTERN FAILURES
5943
5944
5945
5946
5947
5948
5949
5950
5951
5952
5953
5954
5955
5956
5957

```

```

*****
*TEST 33 COUNT PATTERN IN USER PAGE ADDRESS REGISTERS
*
* THIS TEST RUNS A COUNT PATTERN THRU THE USER PAGE ADDRESS
* REGISTERS. SINCE ALL THE BITS ARE IMPLEMENTED, NONE NEED
* TO BE MASKED OUT OF THE COMPARE. IF THE COUNT PATTERN DOES
* NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS, DATA
* PATTERN, AND BAD DATA ARE REPORTED. AT THE END OF THE TEST A
* SUMMARY OF THE ERRORS IS GIVEN.
*
* ALL ERRORS FOUND BY THIS TEST ARE REPORTED AND ANALYZED BY
* SUBROUTINE 'PARCOUNT'.
*
*****

```

```

5958 045600 TST33:
5959 045600 000004 SCOPE
5960 045602 012737 045730 001316 MOV #TST34,NXTTST ;SAVE STARTING ADDRESS OF NEXT
5961 ;TEST FOR ESCAPE ON PARITY ERRORS
5962 045610 012737 000012 001206 MOV #12,$TIMES ;DO 12 ITERATIONS
5963 045616 004737 031346 20$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
5964 045622 012737 045636 001112 MOV #2$,$LPERR ;SET LOOP ON ERROR POINTER TO 1$
5965 045630 005001 CLR R1 ;CLEAR REGISTER TO HOLD COUNT PATTERN
5966 045632 012700 177640 1$: MOV #UIPAR0,R0 ;PUT ADDRESS OF FIRST REGISTER IS R0
5967 045636 000240 2$: NOP ;THIS IS A SYNC POINT FOR SCOPING
5968 045640 010110 MOV R1,(R0) ;LOAD COUNT INTO REGISTER
5969 045642 011002 MOV (R0),R2 ;READ REGISTER BACK TO R2
5970 045644 010104 MOV R1,R4 ;PUT PATTERN IS R4
5971 045646 020402 CMP R4,R2 ;SEE IF DATA MATCHES PATTERN
5972 045650 001402 BEQ 3$ ;BRANCH IF DATA IS GOOD
5973 045652 004737 031600 JSR PC,PARCOUNT ;LOG AND REPORT COUNT ERROR
5974 045656 062700 000002 3$: ADD #2,R0 ;POINT TO NEXT REGISTER
5975 045662 022700 177676 CMP #UDPAR7,R0 ;SEE IF YOU PASSED THE UDPAR7 PAR
5976 045666 103363 BHIS 2$ ;BRANCH IF MORE PAR'S TO TEST
5977 045670 022701 177777 CMP #177777,R1 ;SEE IF COUNT HAS REACHED 177777
5978 045674 001403 BEQ 4$ ;BRANCH IF COUNT IS 177777
5979 045676 062701 000401 ADD #401,R1 ;INCREASE COUNT PATTERN
5980 045702 000753 BR 1$ ;BRANCH TO CONTINUE TEST
5981 045704 012737 045616 001112 4$: MOV #20$,$LPERR ;SET LOOP POINTER TO START OF TEST
5982 045712 005737 001302 TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
5983 045716 001404 BEQ TST34 ;BRANCH TO NEXT TEST IF NO ERRORS
5984 045720 013737 001302 001204 MOV ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
5985 045726 104022 ERROR 22 ;SUMMARY OF COUNT PATTERN FAILURES

```

5986
 5987
 5988
 5989
 5990
 5991
 5992
 5993
 5994
 5995
 5996
 5997
 5998
 5999
 6000
 6001
 6002
 6003
 6004
 6005
 6006
 6007
 6008
 6009
 6010
 6011
 6012
 6013
 6014
 6015
 6016
 6017
 6018
 6019
 6020
 6021
 6022
 6023
 6024
 6025
 6026
 6027
 6028
 6029
 6030
 6031
 6032
 6033
 6034

045730
 045730 000004
 045732 012737 046106 001316
 045740 012737 000012 001206
 045746 004737 031346
 045752 012737 045766 001112
 045760 005001
 045762 012700 172300
 045766 000240
 045770 010110
 045772 011002
 045774 010104
 045776 105737 001362
 046002 001003
 046004 005737 001360
 046010 001403
 046012 012737 000360 046022
 046020 042704 100360
 046024 020402
 046026 001402
 046030 004737 031600
 046034 062700 000002
 046040 022700 172336
 046044 103350
 046046 022701 177777
 046052 001403
 046054 062701 000401
 046060 000740
 046062 012737 045746 001112
 046070 005737 001302
 046074 001404
 046076 013737 001302 001204
 046104 104022

```

*****
*TEST 34      COUNT PATTERN IN KERNEL PAGE DESCRIPTOR REGISTERS
*
*   THIS TEST RUNS A COUNT PATTERN THRU THE KERNEL PAGE DESCRIPTOR
*   REGISTERS.  SINCE BITS <15> AND <05:04> 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 THE ERRORS IS GIVEN.
*****
    
```

```

TST34:
SCOPE
MOV      #TST35,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
                                ;TEST FOR ESCAPE ON PARITY ERRORS
MOV      #12,$TIMES        ;DO 12 ITERATIONS
JSR      PC,CLEANUP        ;INITIALIZE ERROR LOCATIONS
MOV      #2,$SLPERR        ;SET LOOP ON ERROR POINTER TO 1$
CLR      R1                ;CLEAR REGISTER TO HOLD COUNT PATTERN
MOV      #KIPDR0,R0        ;PUT ADDRESS OF FIRST REGISTER IS R0
NOP                               ;THIS IS A SYNC POINT FOR SCOPING
MOV      R1,(R0)           ;LOAD COUNT INTO REGISTER
MOV      (R0),R2           ;READ REGISTER BACK TO R2
MOV      R1,R4             ;PUT PATTERN IS R4
TSTB    KB11CM             ;IS THIS A MODIFIED CPU?
BNE     6$                ;YES
TST     KB11E              ;IS IT A KB11E OR KB11EM?
BEQ     5$                ;BR IF NOT
MOV     #360,5$+2         ;DIDDLE 100360
BIC     #100360,R4        ;CLEAR BITS NOT FOUND IN REGISTER.
CMP     R4,R2             ;SEE IF DATA MATCHES PATTERN
BEQ     3$                ;BRANCH IF DATA IS GOOD
JSR     PC,PARCOUNT     ;LOG AND REPORT COUNT ERROR
ADD     #2,R0              ;POINT TO NEXT REGISTER
CMP     #KDPDR7,R0        ;SEE IF YOU PASSED THE KDPDR7 PDR
BHS     2$                ;BRANCH IF MORE PDR'S TO TEST
CMP     #177777,R1        ;SEE IF COUNT HAS REACHED 177777
BEQ     4$                ;BRANCH IF COUNT IS 177777
ADD     #401,R1           ;INCREASE COUNT PATTERN
BR      1$                ;BRANCH TO CONTINUE TEST
MOV     #20$,$LPERR       ;SET LOOP POINTER TO START OF TEST
TST     ERRCNT            ;SEE IF THERE WERE ANY ERRORS
BEQ     TST35             ;BRANCH TO NEXT TEST IF NO ERRORS
MOV     ERRCNT,$TMP5      ;SAVE NUMBER OF ERRORS FOR TYPEOUT
ERROR   22                ;SUMMARY OF COUNT PATTERN FAILURES
    
```

6035
 6036
 6037
 6038
 6039
 6040
 6041
 6042
 6043
 6044
 6045
 6046
 6047
 6048
 6049 046106
 6050 046106 000004
 6051 046110 012737 046264 001316
 6052
 6053 046116 012737 000012 001206
 6054 046124 004737 031346 20\$:
 6055 046130 012737 046144 001112
 6056 046136 005001
 6057 046140 012700 172200 1\$:
 6058 046144 000240 2\$:
 6059 046146 010110
 6060 046150 011002
 6061 046152 010104
 6062 046154 105737 001362
 6063 046160 001003
 6064 046162 005737 001360
 6065 046166 001403
 6066 046170 012737 000360 046200 6\$:
 6067 046176 042704 100360 5\$:
 6068 046202 020402
 6069 046204 001402
 6070 046206 004737 031600
 6071 046212 062700 000002 3\$:
 6072 046216 022700 172236
 6073 046222 103350
 6074 046224 022701 177777
 6075 046230 001403
 6076 046232 062701 000401
 6077 046236 000740
 6078 046240 012737 046124 001112 4\$:
 6079 046246 005737 001302
 6080 046252 001404
 6081 046254 013737 001302 001204
 6082 046262 104022
 6083
 6084
 6085
 6086
 6087
 6088
 6089
 6090

 *TEST 35 COUNT PATTERN IN SUPERVISOR PAGE DESCRIPTOR REGISTERS

 THIS TEST RUNS A COUNT PATTERN THRU THE SUPERVISOR PAGE DESCRIPTOR
 REGISTERS. SINCE BITS <15> AND <05:04> 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 THE ERRORS IS GIVEN.

 TST35:
 SCOPE
 MOV #TST36,NXTTST ;SAVE STARTING ADDRESS OF NEXT
 ;TEST FOR ESCAPE ON PARITY ERRORS
 MOV #12,\$TIMES ;DO 12 ITERATIONS
 JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
 MOV #2,\$SLPERR ;SET LOOP ON ERROR POINTER TO 1\$
 CLR R1 ;CLEAR REGISTER TO HOLD COUNT PATTERN
 MOV #SIPDR0,R0 ;PUT ADDRESS OF FIRST REGISTER IS R0
 NOP ;THIS IS A SYNC POINT FOR SCOPING
 MOV R1,(R0) ;LOAD COUNT INTO REGISTER
 MOV (R0),R2 ;READ REGISTER BACK TO R2
 MOV R1,R4 ;PUT PATTERN IS R4
 TSTB KB11CM ;IS THIS A MODIFIED CPU?
 BNE 6\$;YES
 TST KB11E ;IS IT A KB11E OR KB11EM?
 BEQ 5\$;BR IF NOT
 MOV #360,5\$+2 ;DIDDLE 100360
 BIC #100360,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
 ADD #2,R0 ;POINT TO NEXT REGISTER
 CMP #SDPDR7,R0 ;SEE IF YOU PASSED THE SDPDR7 PDR
 BHS 2\$;BRANCH IF MORE PDR'S TO TEST
 CMP #177777,R1 ;SEE IF COUNT HAS REACHED 177777
 BEQ 4\$;BRANCH IF COUNT IS 177777
 ADD #401,R1 ;INCREASE COUNT PATTERN
 BR 1\$;BRANCH TO CONTINUE TEST
 MOV #20\$,SLPERR ;SET LOOP POINTER TO START OF TEST
 TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
 BEQ TST36 ;BRANCH TO NEXT TEST IF NO ERRORS
 MOV ERRCNT,\$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
 ERROR 22 ;SUMMARY OF COUNT PATTERN FAILURES

 *TEST 36 COUNT PATTERN IN USER PAGE DESCRIPTOR REGISTERS

 THIS TEST RUNS A COUNT PATTERN THRU THE USER PAGE DESCRIPTOR
 REGISTERS. SINCE BITS <15> AND <05:04> 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

6091 :
6092 :
6093 :
6094 :
6095 :
6096 :
6097 :
6098 :
6099 :
6100 :
6101 :
6102 :
6103 :
6104 :
6105 :
6106 :
6107 :
6108 :
6109 :
6110 :
6111 :
6112 :
6113 :
6114 :
6115 :
6116 :
6117 :
6118 :
6119 :
6120 :
6121 :
6122 :
6123 :
6124 :
6125 :
6126 :
6127 :
6128 :
6129 :
6130 :
6131 :
6132 :
6133 :
6134 :
6135 :
6136 :
6137 :
6138 :
6139 :
6140 :
6141 :
6142 :
6143 :
6144 :
6145 :
6146 :

DESCRIPTION REGISTER TO FIND BAD ETCHES. IF THE COUNT PATTERN DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS, DATA PATTERN, AND BAD DATA ARE REPORTED. AT THE END OF THE TEST A SUMMARY OF THE ERRORS IS GIVEN.

```
*****
TST36:
SCOPE
MOV #TST37,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
MOV #12,$TIMES ;DO 12 ITERATIONS
JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
MOV #2$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
CLR R1 ;CLEAR REGISTER TO HOLD COUNT PATTERN
MOV #UIPDR0,R0 ;PUT ADDRESS OF FIRST REGISTER IS R0
NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV R1,(R0) ;LOAD COUNT INTO REGISTER
MOV (R0),R2 ;READ REGISTER BACK TO R2
MOV R1,R4 ;PUT PATTERN IS R4
TSTB KB11CM ;IS THIS A MODIFIED CPU?
BNE 6$ ;YES
TST KB11E ;IS IT A KB11E OR KB11EM?
BEQ 5$ ;BR IF NOT
MOV #360,5$+2 ;DIDDLE 100360
BIC #100360,R4 ;CLEAR BITS NOT FOUND IN REGISTER.
CMP R4,R2 ;SEE IF DATA MATCHES PATTERN
BEQ 3$ ;BRANCH IF DATA IS GOOD
JSR PC,PARCOUNT ;LOG AND REPORT COUNT ERROR
ADD #2,R0 ;POINT TO NEXT REGISTER
CMP #UDPDR7,R0 ;SEE IF YOU PASSED THE UDPDR7 PDR
BHS 2$ ;BRANCH IF MORE PDR'S TO TEST
CMP #177777,R1 ;SEE IF COUNT HAS REACHED 177777
BEQ 4$ ;BRANCH IF COUNT IS 177777
ADD #401,R1 ;INCREASE COUNT PATTERN
BR 1$ ;BRANCH TO CONTINUE TEST
MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
BEQ TST37 ;BRANCH TO NEXT TEST IF NO ERRORS
MOV ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
ERROR 22 ;SUMMARY OF COUNT PATTERN FAILURES
```

.SBTTL TEST FOR CORRECT BYTE ADDRESSING OF P.A.R.'S & P.D.R.'S
* THESE NEXT SIX (6) TESTS ARE USED TO TEST THE LOGIC THAT
* ALLOWS BYTE ADDRESSING OF THE PAR'S AND PDR'S. IN EACH
* CASE THE I-SPACE REGISTER 0 IS USED, SINCE IT IS REALLY
* THE WRITE PULSE TO THE REGISTER SET THAT IS BEING TESTED.
* IT IS ASSUMED THAT IF ONE REGISTER OF A GROUP FUNCTIONS
* PROPERLY THAT THEY ALL WILL, SINCE ALL REGISTERS HAVE BEEN
* BIT TESTED EARLIER.

*TEST 37 BYTE ADDRESSING OF KERNEL PAGE ADDRESS REGISTERS

* THIS TEST CHECKS THE 'SAPA WRITE LOBYTE' AND 'SAPA WRITE HIBYTE'
* SIGNAL GENERATION FOR KERNEL PAGE ADDRESS REGISTERS.
* BOTH SIGNALS ARE CONTROLLED BY THE LOGIC ON 'SAPK' THAT

```

6147      ;*      GENERATES 'SAPK LO BYTE B H' AND 'SAPK HI BYTE B H'.
6148      ;*
6149      ;*
6150      ;*****
6151      TST37:
6152      046442 000004 046560 001316      SCOPE
6153      046444 012737      MOV      #TST40,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
6154      046452 012737 046470 001112 20$:  MOV      #11$,SLPERR      ;TEST FOR ESCAPE ON PARITY ERRORS
6155      046460 012702 000015      MOV      #15,R2           ;SET LOOP ON ERROR POINTER TO 11$
6156      046464 012700 172340      MOV      #KIPARO,R0       ;PUT EXPECTED DATA IN R2
6157      046470 005037 172340      11$:  CLR      KIPARO          ;PUT ADDRESS OF REGISTER IN R0
6158      046474 000240      NOP                       ;CLEAR THE REGISTER UNDER TEST
6159      046476 112710 172015      MOV      #172015,(R0)     ;THIS IS A SYNC POINT FOR SCOPING
6160      046502 013701 172340      MOV      KIPARO,R1       ;LOAD LOWER BYTE OF REGISTER
6161      046506 020102      CMP      R1,R2           ;READ REGISTER INTO R1
6162      046510 001401      BEQ      1$              ;SEE IF ONLY LOWER BYTE WAS WRITTEN
6163      046512 104023      ERROR   23              ;BRANCH IF DATA MATCHES
6164      046514 012737 046532 001112 1$:  MOV      #12$,SLPERR      ;DIDN'T LOAD CORRECT BYTE
6165      046522 012700 172341      MOV      #KIPARO+1,R0    ;SET LOOP ON ERROR POINTER TO 12$
6166      046526 012702 052015      MOV      #052015,R2      ;POINT TO UPPER BYTE OF REGISTER
6167      046532 000240      12$:  NOP                       ;LOAD EXPECTED DATA IN R2
6168      046534 112710 000124      MOV      #124,(R0)       ;THIS IS A SYNC POINT FOR SCOPING
6169      046540 013701 172340      MOV      KIPARO,R1       ;WRITE THE UPPER BYTE OF REGISTER
6170      046544 020102      CMP      R1,R2           ;READ REGISTER INTO R1
6171      046546 001401      BEQ      2$              ;SEE IF REGISTER HOLDS CORRECT DATA
6172      046550 104023      ERROR   23              ;BRANCH TO EXIT IF DATA RIGHT
6173      046552 012737 046452 001112 2$:  MOV      #20$,SLPERR     ;DIDN'T LOAD CORRECT BYTE
6174      ;*
6175      ;*
6176      ;*****
6177      ;*TEST 40      BYTE ADDRESSING OF SUPERVISOR PAGE ADDRESS REGISTERS
6178      ;*
6179      ;*
6180      ;*      THIS TEST CHECKS THE 'SAPB WRITE LOBYTE' AND 'SAPB WRITE HIBYTE'
6181      ;*      SIGNAL GENERATION FOR SUPERVISOR PAGE ADDRESS REGISTERS.
6182      ;*      BOTH SIGNALS ARE CONTROLLED BY THE LOGIC ON 'SAPK' THAT
6183      ;*      GENERATES 'SAPK LO BYTE B H' AND 'SAPK HI BYTE B H'.
6184      ;*
6185      ;*****
6186      TST40:
6187      046560 000004 046676 001316      SCOPE
6188      046562 012737      MOV      #TST41,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
6189      046570 012737 046606 001112 20$:  MOV      #11$,SLPERR     ;TEST FOR ESCAPE ON PARITY ERRORS
6190      046576 012702 000015      MOV      #15,R2           ;SET LOOP ON ERROR POINTER TO 11$
6191      046602 012700 172240      MOV      #SIPARO,R0       ;PUT EXPECTED DATA IN R2
6192      046606 005037 172240      11$:  CLR      SIPARO          ;PUT ADDRESS OF REGISTER IN R0
6193      046612 000240      NOP                       ;CLEAR THE REGISTER UNDER TEST
6194      046614 112710 172015      MOV      #172015,(R0)     ;THIS IS A SYNC POINT FOR SCOPING
6195      046620 013701 172240      MOV      SIPARO,R1       ;LOAD LOWER BYTE OF REGISTER
6196      046624 020102      CMP      R1,R2           ;READ REGISTER INTO R1
6197      046626 001401      BEQ      1$              ;SEE IF ONLY LOWER BYTE WAS WRITTEN
6198      046630 104023      ERROR   23              ;BRANCH IF DATA MATCHES
6199      046632 012737 046650 001112 1$:  MOV      #12$,SLPERR     ;DIDN'T LOAD CORRECT BYTE
6200      046640 012700 172241      MOV      #SIPARO+1,R0    ;SET LOOP ON ERROR POINTER TO 12$
6201      046644 012702 052015      MOV      #052015,R2      ;POINT TO UPPER BYTE OF REGISTER
6202      046650 000240      12$:  NOP                       ;LOAD EXPECTED DATA IN R2
6202      ;*
6202      ;*      THIS IS A SYNC POINT FOR SCOPING
    
```



```

6259 047024 012737 047042 001112 20$: MOV #11$,SLPERR ;SET LOOP ON ERROR POINTER TO 11$
6260 047032 012702 000015 MOV #15,R2 ;PUT EXPECTED DATA IN R2
6261 047036 012700 172300 MOV #KIPDRO,R0 ;PUT ADDRESS OF REGISTER IN R0
6262 047042 005037 172300 11$: CLR KIPDRO ;CLEAR THE REGISTER UNDER TEST
6263 047046 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
6264 047050 112710 172015 MOVB #172015,(R0) ;LOAD LOWER BYTE OF REGISTER
6265 047054 013701 172300 MOV KIPDRO,R1 ;READ REGISTER INTO R1
6266 047060 020102 CMP R1,R2 ;SEE IF ONLY LOWER BYTE WAS WRITTEN
6267 047062 001401 BEQ 1$ ;BRANCH IF DATA MATCHES
6268 047064 104023 ERROR 23 ;DIDN'T LOAD CORRECT BYTE
6269 047066 012737 047104 001112 1$: MOV #12$,SLPERR ;SET LOOP ON ERROR POINTER TO 12$
6270 047074 012700 172301 MOV #KIPDRO+1,R0 ;POINT TO UPPER BYTE OF REGISTER
6271 047100 012702 052015 MOV #052015,R2 ;LOAD EXPECTED DATA IN R2
6272 047104 000240 12$: NOP ;THIS IS A SYNC POINT FOR SCOPING
6273 047106 112710 000124 MOVB #124,(R0) ;WRITE THE UPPER BYTE OF REGISTER
6274 047112 013701 172300 MOV KIPDRO,R1 ;READ REGISTER INTO R1
6275 047116 020102 CMP R1,R2 ;SEE IF REGISTER HOLDS CORRECT DATA
6276 047120 001401 BEQ 2$ ;BRANCH TO EXIT IF DATA RIGHT
6277 047122 104023 ERROR 23 ;DIDN'T LOAD CORRECT BYTE
6278 047124 012737 047024 001112 2$: MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
    
```

```

6280
6281 ;*****
6282 ;*TEST 43 BYTE ADDRESSING OF SUPERVISOR PAGE DESCRIPTOR REGISTERS
6283 ;
6284 ; THIS TEST CHECKS THE 'SAPE WRITE LOBYTE' AND 'SAPE WRITE HIBYTE'
6285 ; SIGNAL GENERATION FOR SUPERVISOR PAGE ADDRESS REGISTERS.
6286 ; BOTH SIGNALS ARE CONTROLLED BY THE LOGIC ON 'SAPK' THAT
6287 ; GENERATES 'SAPK LO BYTE B H' AND 'SAPK HI BYTE B H'.
6288 ;
6289 ;*****
    
```

```

6290 047132 TST43:
6291 047132 000004 SCOPE
6292 047134 012737 047250 001316 MOV #TST44,NXTTST ;SAVE STARTING ADDRESS OF NEXT
6293 ;TEST FOR ESCAPE ON PARITY ERRORS
6294 047142 012737 047160 001112 20$: MOV #11$,SLPERR ;SET LOOP ON ERROR POINTER TO 11$
6295 047150 012702 000015 MOV #15,R2 ;PUT EXPECTED DATA IN R2
6296 047154 012700 172200 MOV #SIPDRO,R0 ;PUT ADDRESS OF REGISTER IN R0
6297 047160 005037 172200 11$: CLR SIPDRO ;CLEAR THE REGISTER UNDER TEST
6298 047164 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
6299 047166 112710 172015 MOVB #172015,(R0) ;LOAD LOWER BYTE OF REGISTER
6300 047172 013701 172200 MOV SIPDRO,R1 ;READ REGISTER INTO R1
6301 047176 020102 CMP R1,R2 ;SEE IF ONLY LOWER BYTE WAS WRITTEN
6302 047200 001401 BEQ 1$ ;BRANCH IF DATA MATCHES
6303 047202 104023 ERROR 23 ;DIDN'T LOAD CORRECT BYTE
6304 047204 012737 047222 001112 1$: MOV #12$,SLPERR ;SET LOOP ON ERROR POINTER TO 12$
6305 047212 012700 172201 MOV #SIPDRO+1,R0 ;POINT TO UPPER BYTE OF REGISTER
6306 047216 012702 052015 MOV #052015,R2 ;LOAD EXPECTED DATA IN R2
6307 047222 000240 12$: NOP ;THIS IS A SYNC POINT FOR SCOPING
6308 047224 112710 000124 MOVB #124,(R0) ;WRITE THE UPPER BYTE OF REGISTER
6309 047230 013701 172200 MOV SIPDRO,R1 ;READ REGISTER INTO R1
6310 047234 020102 CMP R1,R2 ;SEE IF REGISTER HOLDS CORRECT DATA
6311 047236 001401 BEQ 2$ ;BRANCH TO EXIT IF DATA RIGHT
6312 047240 104023 ERROR 23 ;DIDN'T LOAD CORRECT BYTE
6313 047242 012737 047142 001112 2$: MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
6314
    
```

6315
6316
6317
6318
6319
6320
6321
6322
6323
6324
6325
6326
6327
6328
6329
6330
6331
6332
6333
6334
6335
6336
6337
6338
6339
6340
6341
6342
6343
6344
6345
6346
6347
6348
6349
6350
6351
6352
6353
6354
6355
6356
6357
6358
6359
6360
6361
6362
6363
6364
6365
6366
6367
6368
6369
6370

 *TEST 44 BYTE ADDRESSING OF USER PAGE DESCRIPTOR REGISTERS

THIS TEST CHECKS THE 'SAPF WRITE LOBYTE' AND 'SAPF WRITE HIBYTE'
 SIGNAL GENERATION FOR USER PAGE ADDRESS REGISTERS.
 BOTH SIGNALS ARE CONTROLLED BY THE LOGIC ON 'SAPK' THAT
 GENERATES 'SAPK LO BYTE B H' AND 'SAPK HI BYTE B H'.

TST44:

SCOPE	MOV	#TST45,NXTTST	:	SAVE STARTING ADDRESS OF NEXT
			:	TEST FOR ESCAPE ON PARITY ERRORS
20\$:	MOV	#11\$, \$LPERR	:	SET LOOP ON ERROR POINTER TO 11\$
	MOV	#15, R2	:	PUT EXPECTED DATA IN R2
	MOV	#UIPDRO, R0	:	PUT ADDRESS OF REGISTER IN R0
11\$:	CLR	UIPDRO	:	CLEAR THE REGISTER UNDER TEST
	NOP		:	THIS IS A SYNC POINT FOR SCOPING
	MOVB	#172015, (R0)	:	LOAD LOWER BYTE OF REGISTER
	MOV	UIPDRO, R1	:	READ REGISTER INTO R1
	CMP	R1, R2	:	SEE IF ONLY LOWER BYTE WAS WRITTEN
	BEQ	1\$:	BRANCH IF DATA MATCHES
	ERROR	23	:	DIDN'T LOAD CORRECT BYTE
1\$:	MOV	#12\$, \$LPERR	:	SET LOOP ON ERROR POINTER TO 12\$
	MOV	#UIPDRO+1, R0	:	POINT TO UPPER BYTE OF REGISTER
	MOV	#052015, R2	:	LOAD EXPECTED DATA IN R2
12\$:	NOP		:	THIS IS A SYNC POINT FOR SCOPING
	MOVB	#124, (R0)	:	WRITE THE UPPER BYTE OF REGISTER
	MOV	UIPDRO, R1	:	READ REGISTER INTO R1
	CMP	R1, R2	:	SEE IF REGISTER HOLDS CORRECT DATA
	BEQ	2\$:	BRANCH TO EXIT IF DATA RIGHT
	ERROR	23	:	DIDN'T LOAD CORRECT BYTE
2\$:	MOV	#20\$, \$LPERR	:	SET LOOP POINTER TO START OF TEST

.SBTTL DUAL ADDRESSING BETWEEN GROUPS OF REGISTERS

 *TEST 45 DUAL ADDRESSING FOR ALL PAR'S AND 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 KERNEL
 I-SPACE PAR YOU ARE REALLY REFERENCING THAT PAR. FIRST EACH
 I-SPACE PARO OR PDRO IS LOADED WITH A UNIQUE NUMBER 0-12 AND
 THEN THEY ARE EACH CHECKED FOR THE CORRECT DATA.
 EACH GROUP HAS ALREADY BEEN CHECKED FOR DUAL ADDRESSING WITHIN
 ITS OWN GROUP, SO ONLY ONE REGISTER FROM EACH GROUP NEEDS TO
 BE TESTED HERE.

TST45:

SCOPE	MOV	#TST46,NXTTST	:	SAVE STARTING ADDRESS OF NEXT
			:	TEST FOR ESCAPE ON PARITY ERRORS
20\$:	MOV	#1\$, \$LPERR	:	SET LOOP ON ERROR POINTER TO 1\$
	CLR	R0	:	THIS WILL HOLD THE INDEX OF EACH REGISTER


```

6371
6372 047406 012704 000006
6373 047412 010070 001322
6374 047416 062700 000002
6375 047422 077405
6376 047424 012704 000006
6377 047430 012737 047442 001112
6378 047436 162700 000002
6379 047442 017001 001322
6380 047446 020001
6381 047450 001403
6382 047452 016002 001322
6383 047456 104036
6384 047460 077412
6385 047462 012737 047376 001112
6386
6387
6388
6389
6390
6391
6392
6393
6394
6395
6396
6397
6398
6399
6400
6401
6402
6403
6404
6405
6406 047470
6407 047470 000004
6408 047472 012737 051104 001316
6409
6410
6411
6412
6413
6414
6415 047500
6416 047500 012737 050714 001110
6417 047506 012737 050714 001112
6418 047514 012737 000046 001102
6419 047522 013737 001102 177570
6420 047530 012705 172300
6421 047534 004737 031330
6422 047540 012705 172340
6423 047544 004737 031330
6424 047550 012705 172200
6425 047554 004737 031330
6426 047560 012705 172240
    
```

```

1S: MOV #6,R4 ;AND THE COUNT LOADED
    MOV R0,@PARTAB(R0) ;THIS IS THE NUMBER OF TIMES TO DO THIS LOOP
    ADD #2,R0 ;LOAD PAR OR PDR WITH INDEX NUMBER
    SOB R4,1S ;CHANGE INDEX TO POINT TO NEXT REGISTER
    MOV #6,R4 ;BRANCH BACK 5 TIMES
    MOV #10$,SLPERR ;DO NEXT LOOP 6 TIMES ALSO
2S: SUB #2,R0 ;SET LOOP ON ERROR POINTER TO 10$
10$: MOV @PARTAB(R0),R1 ;ADJUST INDEX FOR REGISTER DESIRED
    CMP R0,R1 ;READ PAR OR PDR INTO R1
    BEQ 3S ;SEE IF INDEX EQUALS DATA
    MOV PARTAB(R0),R2 ;BRANCH IFCORRECT REGISTER WAS READ.
    ERROR 36 ;SAVE ADDRESS OF BAD REGISTER
    SOB R4,2S ;DUAL ADDRESSING
    MOV #20$,SLPERR ;BRANCH BACK 5 TIMES
    ;SET LOOP ON ERROR POINTER TO START OF TEST
    
```

```

.SBTTL ***** ENTRY POINT 4 --- STARTING ADDRESS 214 *****
.SBTTL ***** RELOCATION AND ADDER TESTS *****
:
: THIS GROUP OF TESTS CHECKS MEMORY MANAGEMENT'S RELOCATION ADDER
: FIRST USING DESTINATION ONLY RELOCATION, THEN WITH FULL 18-BIT
: RELOCATION AND FINALLY WITH 22-BIT RELOCATION.
:
    
```

```

:*****
:*TEST 46 18-BIT MAPPING ADDER TESTING
:
: THIS TEST USES 'DESTINATION ONLY' RELOCATION TO CHECK THE
: FULL ADD PROPERTIES OF THE RELOCATION ADDER. TWELVE PAIRS
: OF NUMBERS ARE ADDED, GENERATING PHYSICAL ADDRESSES ABOVE
: 16K.
:*****
    
```

```

TST46:
    SCOPE
    MOV #TST47,NXTTST ;SAVE STARTING ADDRESS OF NEXT
    ;TEST FOR ESCAPE ON PARITY ERRORS
    
```

```

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

```

ENTPT4:
    MOV #20$,SLPADR ;SET LOOP ADDRESS POINTER TO 20$
    MOV #20$,SLPERR ;SET LOOP ON ERROR POINTER TO 20$
    MOV #46,$TSTNM ;LOAD TEST NUMBER INTO MEMORY
    MOV $TSTNM,DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
    MOV #KIPDR0,R5 ;PUT ADDRESS OF KIPDR0 IN R5
    JSR PC,CLRRÉG ;CLEAR KERNEL PDR'S
    MOV #KIPAR0,R5 ;PUT ADDRESS OF KIPAR0 IN R5
    JSR PC,CLRRÉG ;CLEAR KERNEL PAR'S
    MOV #SIPDR0,R5 ;PUT ADDRESS OF SIPDR0 IN R5
    JSR PC,CLRRÉG ;CLEAR SUPERVISOR PDR'S
    MOV #SIPAR0,R5 ;PUT ADDRESS OF SIPAR0 IN R5
    
```

6427	047564	004737	031330			JSR	PC,CLRREG	:CLEAR SUPERVISOR PAR'S
6428	047570	012705	177600			MOV	#UIPDRO,R5	:PUT ADDRESS OF UIPDRO IN R5
6429	047574	004737	031330			JSR	PC,CLRREG	:CLEAR USER PDR'S
6430	047600	012705	177640			MOV	#UIPAR0,R5	:PUT ADDRESS OF UIPAR0 IN R5
6431	047604	004737	031330			JSR	PC,CLRREG	:CLEAR USER PAR'S
6432	047610	012700	077406	30\$:		MOV	#77406,R0	:MAKE KERNEL I PAGES 4K, R/W, EXPAND UP
6433	047614	012702	000010			MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
6434	047620	012701	172300			MOV	#KIPDRO,R1	:PUT ADDRESS OF FIRST PDR IN R1
6435	047624	010021		64\$:		MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
6436	047626	077202				SOB	R2,64\$:BRANCH BACK TO 64\$ IF R2 IS NOT ZERO
6437	047630	012737	000000	172340		MOV	#000,KIPAR0	:MAP PAGE 0 TO 0 - 4K
6438	047636	012737	000200	172342		MOV	#200,KIPAR1	:MAP PAGE 1 TO 4K - 8K
6439	047644	012737	000400	172344		MOV	#400,KIPAR2	:MAP PAGE 2 TO 8K - 12K
6440	047652	012737	000600	172346		MOV	#600,KIPAR3	:MAP PAGE 3 TO 12K - 16K
6441	047660	012737	177600	172356		MOV	#177600,KIPAR7	:MAP PAGE 7 TO I/O PAGE
6442	047666	012737	047720	001112		MOV	#1\$,SLPERR	:SET LOOP ON ERROR POINTER TO 1\$
6443	047674	013737	100000	001172		MOV	@#100000,\$TMP0	:SAVE DATA AT TEST LOCATION
6444	047702	012737	001000	172350		MOV	#1000,@#KIPAR4	:LOAD PAR4 WITH 1000
6445	047710	012700	100000			MOV	#100000,R0	:PUT VIRTUAL ADDRESS IN R0
6446	047714	012701	125200			MOV	#125200,R1	:PUT DATA PATTERN IN R1
6447	047720	052737	000400	177572	1\$:	BIS	#BIT8,@#MMRO	:TURN ON DESTINATION ONLY RELOCATION
6448	047726	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6449	047730	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 100000
6450	047732	013702	100000			MOV	@#100000,R2	:READ (100000) INTO R2
6451	047736	000005				RESET		:CLEAR MMRO
6452	047740	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6453	047742	001401				BEQ	2\$:BRANCH IF DATA MATCHES
6454	047744	104037				ERROR	37	:RELOCATION FAILED
6455	047746	013737	001172	100000	2\$:	MOV	\$TMP0,@#100000	:RESTORE ORIGINAL DATA TO TEST LOCATION
6456	047754	012737	050006	001112		MOV	#3\$,SLPERR	:SET LOOP ON ERROR POINTER TO 3\$
6457	047762	013737	137776	001172		MOV	@#137776,\$TMP0	:SAVE DATA AT TEST LOCATION
6458	047770	012737	001252	172350		MOV	#1252,@#KIPAR4	:LOAD PAR4 WITH 1252
6459	047776	012700	112576			MOV	#112576,R0	:PUT VIRTUAL ADDRESS IN R0
6460	050002	012701	125201			MOV	#125201,R1	:PUT DATA PATTERN IN R1
6461	050006	052737	000400	177572	3\$:	BIS	#BIT8,@#MMRO	:TURN ON DESTINATION ONLY RELOCATION
6462	050014	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6463	050016	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 137776
6464	050020	013702	137776			MOV	@#137776,R2	:READ (137776) INTO R2
6465	050024	000005				RESET		:CLEAR MMRO
6466	050026	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6467	050030	001401				BEQ	4\$:BRANCH IF DATA MATCHES
6468	050032	104037				ERROR	37	:RELOCATION FAILED
6469	050034	013737	001172	137776	4\$:	MOV	\$TMP0,@#137776	:RESTORE ORIGINAL DATA TO TEST LOCATION
6470	050042	012737	050074	001112		MOV	#5\$,SLPERR	:SET LOOP ON ERROR POINTER TO 5\$
6471	050050	013737	117776	001172		MOV	@#117776,\$TMP0	:SAVE DATA AT TEST LOCATION
6472	050056	012737	001125	172350		MOV	#1125,@#KIPAR4	:LOAD PAR4 WITH 1125
6473	050064	012700	105276			MOV	#105276,R0	:PUT VIRTUAL ADDRESS IN R0
6474	050070	012701	125202			MOV	#125202,R1	:PUT DATA PATTERN IN R1
6475	050074	052737	000400	177572	5\$:	BIS	#BIT8,@#MMRO	:TURN ON DESTINATION ONLY RELOCATION
6476	050102	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6477	050104	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 117776
6478	050106	013702	117776			MOV	@#117776,R2	:READ (117776) INTO R2
6479	050112	000005				RESET		:CLEAR MMRO
6480	050114	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6481	050116	001401				BEQ	6\$:BRANCH IF DATA MATCHES
6482	050120	104037				ERROR	37	:RELOCATION FAILED

6483	050122	013737	001172	117776	6\$:	MOV	\$TMP0,@#117776	:RESTORE ORIGINAL DATA TO TEST LOCATION
6484	050130	012737	050162	001112		MOV	#7\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 7\$
6485	050136	013737	102000	001172		MOV	@#102000,\$TMP0	:SAVE DATA AT TEST LOCATION
6486	050144	012737	001010	172350		MOV	#1010,@#KIPAR4	:LOAD PAR4 WITH 1010
6487	050152	012700	101000			MOV	#101000,R0	:PUT VIRTUAL ADDRESS IN R0
6488	050156	012701	125203			MOV	#125203,R1	:PUT DATA PATTERN IN R1
6489	050162	052737	000400	177572	7\$:	BIS	#BIT8,@#MRO	:TURN ON DESTINATION ONLY RELOCATION
6490	050170	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6491	050172	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 102000
6492	050174	013702	102000			MOV	@#102000,R2	:READ (102000) INTO R2
6493	050200	000005				RESET		:CLEAR MRO
6494	050202	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6495	050204	001401				BEQ	8\$:BRANCH IF DATA MATCHES
6496	050206	104037				ERROR	37	:RELOCATION FAILED
6497	050210	013737	001172	102000	8\$:	MOV	\$TMP0,@#102000	:RESTORE ORIGINAL DATA TO TEST LOCATION
6498	050216	012737	050250	001112		MOV	#9\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 9\$
6499	050224	013737	142000	001172		MOV	@#142000,\$TMP0	:SAVE DATA AT TEST LOCATION
6500	050232	012737	001314	172350		MOV	#1314,@#KIPAR4	:LOAD PAR4 WITH 1314
6501	050240	012700	110400			MOV	#110400,R0	:PUT VIRTUAL ADDRESS IN R0
6502	050244	012701	125204			MOV	#125204,R1	:PUT DATA PATTERN IN R1
6503	050250	052737	000400	177572	9\$:	BIS	#BIT8,@#MRO	:TURN ON DESTINATION ONLY RELOCATION
6504	050256	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6505	050260	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 142000
6506	050262	013702	142000			MOV	@#142000,R2	:READ (142000) INTO R2
6507	050266	000005				RESET		:CLEAR MRO
6508	050270	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6509	050272	001401				BEQ	10\$:BRANCH IF DATA MATCHES
6510	050274	104037				ERROR	37	:RELOCATION FAILED
6511	050276	013737	001172	142000	10\$:	MOV	\$TMP0,@#142000	:RESTORE ORIGINAL DATA TO TEST LOCATION
6512	050304	012737	050336	001112		MOV	#11\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 11\$
6513	050312	013737	122000	001172		MOV	@#122000,\$TMP0	:SAVE DATA AT TEST LOCATION
6514	050320	012737	001104	172350		MOV	#1104,@#KIPAR4	:LOAD PAR4 WITH 1104
6515	050326	012700	111400			MOV	#111400,R0	:PUT VIRTUAL ADDRESS IN R0
6516	050332	012701	125205			MOV	#125205,R1	:PUT DATA PATTERN IN R1
6517	050336	052737	000400	177572	11\$:	BIS	#BIT8,@#MRO	:TURN ON DESTINATION ONLY RELOCATION
6518	050344	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6519	050346	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 122000
6520	050350	013702	122000			MOV	@#122000,R2	:READ (122000) INTO R2
6521	050354	000005				RESET		:CLEAR MRO
6522	050356	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6523	050360	001401				BEQ	12\$:BRANCH IF DATA MATCHES
6524	050362	104037				ERROR	37	:RELOCATION FAILED
6525	050364	013737	001172	122000	12\$:	MOV	\$TMP0,@#122000	:RESTORE ORIGINAL DATA TO TEST LOCATION
6526	050372	012737	050424	001112		MOV	#13\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 13\$
6527	050400	013737	142000	001172		MOV	@#142000,\$TMP0	:SAVE DATA AT TEST LOCATION
6528	050406	012737	001356	172350		MOV	#1356,@#KIPAR4	:LOAD PAR4 WITH 1356
6529	050414	012700	104200			MOV	#104200,R0	:PUT VIRTUAL ADDRESS IN R0
6530	050420	012701	125206			MOV	#125206,R1	:PUT DATA PATTERN IN R1
6531	050424	052737	000400	177572	13\$:	BIS	#BIT8,@#MRO	:TURN ON DESTINATION ONLY RELOCATION
6532	050432	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6533	050434	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 142000
6534	050436	013702	142000			MOV	@#142000,R2	:READ (142000) INTO R2
6535	050442	000005				RESET		:CLEAR MRO
6536	050444	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6537	050446	001401				BEQ	14\$:BRANCH IF DATA MATCHES
6538	050450	104037				ERROR	37	:RELOCATION FAILED

6539	050452	013737	001172	142000	14\$:	MOV	\$TMP0, @#142000	:RESTORE ORIGINAL DATA TO TEST LOCATION
6540	050460	012737	050512	001112		MOV	#15\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 15\$
6541	050466	013737	142000	001172		MOV	@#142000, \$TMP0	:SAVE DATA AT TEST LOCATION
6542	050474	012737	001242	172350		MOV	#1242, @#KIPAR4	:LOAD PAR4 WITH 1242
6543	050502	012700	115600			MOV	#115600, R0	:PUT VIRTUAL ADDRESS IN R0
6544	050506	012701	125207			MOV	#125207, R1	:PUT DATA PATTERN IN R1
6545	050512	052737	000400	177572	15\$:	BIS	#BIT8, @#MMRO	:TURN ON DESTINATION ONLY RELOCATION
6546	050520	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6547	050522	010110				MOV	R1, (R0)	:TRY TO LOAD DATA PATTERN INTO 142000
6548	050524	013702	142000			MOV	@#142000, R2	:READ (142000) INTO R2
6549	050530	000005				RESET		:CLEAR MMRO
6550	050532	020102				CMP	R1, R2	:SEE IF DATA MATCHES PATTERN
6551	050534	001401				BEQ	16\$:BRANCH IF DATA MATCHES
6552	050536	104037				ERROR	37	:RELOCATION FAILED
6553	050540	013737	001172	142000	16\$:	MOV	\$TMP0, @#142000	:RESTORE ORIGINAL DATA TO TEST LOCATION
6554	050546	012737	050600	001112		MOV	#17\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 17\$
6555	050554	013737	142000	001172		MOV	@#142000, \$TMP0	:SAVE DATA AT TEST LOCATION
6556	050562	012737	001377	172350		MOV	#1377, @#KIPAR4	:LOAD PAR4 WITH 1377
6557	050570	012700	102100			MOV	#102100, R0	:PUT VIRTUAL ADDRESS IN R0
6558	050574	012701	125210			MOV	#125210, R1	:PUT DATA PATTERN IN R1
6559	050600	052737	000400	177572	17\$:	BIS	#BIT8, @#MMRO	:TURN ON DESTINATION ONLY RELOCATION
6560	050606	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6561	050610	010110				MOV	R1, (R0)	:TRY TO LOAD DATA PATTERN INTO 142000
6562	050612	013702	142000			MOV	@#142000, R2	:READ (142000) INTO R2
6563	050616	000005				RESET		:CLEAR MMRO
6564	050620	020102				CMP	R1, R2	:SEE IF DATA MATCHES PATTERN
6565	050622	001401				BEQ	18\$:BRANCH IF DATA MATCHES
6566	050624	104037				ERROR	37	:RELOCATION FAILED
6567	050626	013737	001172	142000	18\$:	MOV	\$TMP0, @#142000	:RESTORE ORIGINAL DATA TO TEST LOCATION
6568	050634	012737	050666	001112		MOV	#19\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 19\$
6569	050642	013737	142000	001172		MOV	@#142000, \$TMP0	:SAVE DATA AT TEST LOCATION
6570	050650	012737	001221	172350		MOV	#1221, @#KIPAR4	:LOAD PAR4 WITH 1221
6571	050656	012700	117700			MOV	#117700, R0	:PUT VIRTUAL ADDRESS IN R0
6572	050662	012701	125211			MOV	#125211, R1	:PUT DATA PATTERN IN R1
6573	050666	052737	000400	177572	19\$:	BIS	#BIT8, @#MMRO	:TURN ON DESTINATION ONLY RELOCATION
6574	050674	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6575	050676	010110				MOV	R1, (R0)	:TRY TO LOAD DATA PATTERN INTO 142000
6576	050700	013702	142000			MOV	@#142000, R2	:READ (142000) INTO R2
6577	050704	000005				RESET		:CLEAR MMRO
6578	050706	020102				CMP	R1, R2	:SEE IF DATA MATCHES PATTERN
6579	050710	001401				BEQ	20\$:BRANCH IF DATA MATCHES
6580	050712	104037				ERROR	37	:RELOCATION FAILED
6581	050714	013737	001172	142000	20\$:	MOV	\$TMP0, @#142000	:RESTORE ORIGINAL DATA TO TEST LOCATION
6582	050722	012737	050754	001112		MOV	#21\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 21\$
6583	050730	013737	140000	001172		MOV	@#140000, \$TMP0	:SAVE DATA AT TEST LOCATION
6584	050736	012737	001377	172350		MOV	#1377, @#KIPAR4	:LOAD PAR4 WITH 1377
6585	050744	012700	100100			MOV	#100100, R0	:PUT VIRTUAL ADDRESS IN R0
6586	050750	012701	125212			MOV	#125212, R1	:PUT DATA PATTERN IN R1
6587	050754	052737	000400	177572	21\$:	BIS	#BIT8, @#MMRO	:TURN ON DESTINATION ONLY RELOCATION
6588	050762	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6589	050764	010110				MOV	R1, (R0)	:TRY TO LOAD DATA PATTERN INTO 140000
6590	050766	013702	140000			MOV	@#140000, R2	:READ (140000) INTO R2
6591	050772	000005				RESET		:CLEAR MMRO
6592	050774	020102				CMP	R1, R2	:SEE IF DATA MATCHES PATTERN
6593	050776	001401				BEQ	22\$:BRANCH IF DATA MATCHES
6594	051000	104037				ERROR	37	:RELOCATION FAILED

```

6595 051002 013737 001172 140000 22$: MOV $TMP0,#140000 ;RESTORE ORIGINAL DATA TO TEST LOCATION
6596 051010 012737 051042 001112 MOV #23$,SLPERR ;SET LOOP ON ERROR POINTER TO 23$
6597 051016 013737 140000 001172 MOV #140000,$TMP0 ;SAVE DATA AT TEST LOCATION
6598 051024 012737 001370 172350 MOV #1370,$KIPAR4 ;LOAD PAR4 WITH 1370
6599 051032 012737 101000 MOV #101000,R0 ;PUT VIRTUAL ADDRESS IN R0
6600 051036 012701 125213 MOV #125213,R1 ;PUT DATA PATTERN IN R1
6601 051042 052737 000400 177572 23$: BIS #BIT8,$MPPRO ;TURN ON DESTINATION ONLY RELOCATION
6602 051050 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
6603 051052 010110 MOV R1,(R0) ;TRY TO LOAD DATA PATTERN INTO 140000
6604 051054 013702 140000 MOV #140000,R2 ;READ (140000) INTO R2
6605 051060 000005 RESET ;CLEAR MPPRO
6606 051062 020102 CMP R1,R2 ;SEE IF DATA MATCHES PATTERN
6607 051064 001401 BEQ 24$ ;BRANCH IF DATA MATCHES
6608 051066 104037 ERROR 37 ;RELOCATION FAILED
6609 051070 013737 001172 140000 24$: MOV $TMP0,#140000 ;RESTORE ORIGINAL DATA TO TEST LOCATION
6610 051076 012737 047610 001112 MOV #30$,SLPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
    
```

```

*****
*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 00750000 ON 2K BOUNDARIES, IT WILL
* REPORT ANY HOLES THAT IT FINDS IN MEMORY UP TO THE SIZE
* JUMPERS. THE INFORMATION GIVEN WILL BE THE ADDRESS WHERE
* THE HOLE WAS DISCOVERED AND THE FIRST GOOD ADDRESS AFTER THE
* HOLE.
*****
    
```

```

6623 051104 TST47: SCOPE
6624 051104 000004 MOV #TST50,NXTTST ;SAVE STARTING ADDRESS OF NEXT
6625 051106 012737 051734 001316 MOV ;TEST FOR ESCAPE ON PARITY ERRORS
6626 ;MAKE SURE HOLE FLAG STARTS AT 0
6627 051114 005037 001304 20$: CLR HOLFLG
6628 051120 012737 051550 000004 MOV #10$,ERRVEC ;SET ERRVEC POINTER TO 10$
6629 051126 012737 051714 000114 MOV #30$,CACHVEC ;SET UP CACHE VECTOR POINTER FOR SIZE TO HIGH
6630 051134 012737 000777 172350 MOV #777,$KIPAR4 ;LOAD PAR4 WITH STARTING BASE
6631 051142 012737 001000 172352 MOV #1000,$KIPAR5 ;START ADDRESSING WITH 16K
6632 051150 012700 100100 MOV #100100,R0 ;LOAD VIRTUAL ADDR FOR PAR4 IN R0
6633 051154 012701 120000 MOV #120000,R1 ;LOAD VIRTUAL ADDR FOR PAR5 IN R1
6634 051160 012702 001000 MOV #1000,R2 ;LOAD DATA PATTERN INTO R2
6635 051164 012737 051254 001112 MOV #2$,SLPERR ;SET LOOP ON ERROR POINTER TO 2$
6636 051172 012737 000001 177572 MOV #1,$MPPRO ;TURN ON 18-BIT MAPPING
6637 ;
6638 ;FULL RELOCATION STARTS HERE AND CONTINUES FOR REST OF PROGRAM
6639 ;
6640 051200 012737 051254 001112 1$: MOV #2$,SLPERR ;SET LOOP ON ERROR POINTER TO 2$
6641 051206 005037 001260 CLR PCPUER ;CLEAR CPU ERROR FLAG
6642 051212 011137 001172 MOV (R1),$TMP0 ;SAVE DATA AT TEST LOCATION
6643 051216 005737 001260 TST PCPUER ;SEE IF THERE WAS A CPU TRAP
6644 051222 001014 BNE 2$ ;BRANCH IF TRAP OCCURRED
6645 051224 005737 001304 TST HOLFLG ;SEE IF A HOLE WAS FOUND IN MEMORY
6646 051230 001411 BEQ 2$ ;BRANCH IF NO HOLE WAS FOUND
6647 051232 013737 172352 001176 MOV $KIPAR5,$TMP2 ;SAVE PAR THAT POINTS TO END OF HOLE
6648 051240 012737 051114 001112 MOV #20$,SLPERR ;SET LOOP ON ERROR POINTER TO 20$
6649 051246 104040 ERROR 40 ;HOLE IN MEMORY FROM $TMP1 TO $TMP2
6650 051250 005037 001304 CLR HOLFLG ;CLEAR HOLE FLAG IN CASE THERE ARE MORE
    
```

```

6651 051254 000240      2$:  NOP                ;THIS A IS SYNC POINT FOR SCOPING
6652 051256 010210      MOV      R2,(R0)        ;LOAD TEST PATTERN INTO TEST LOCATION
6653 051260 011103      MOV      (R1),R3        ;READ TEST LOCATION VIA DIFFERENT V. A.
6654 051262 020203      CMP      R2,R3          ;SEE IF CORRECT LOCATION WAS REFERENCED
6655 051264 001401      BEQ      3$             ;BRANCH IF CORRECT DATA WAS OBTAINED
6656 051266 104042      ERROR   42             ;BAD RELOCATION
6657 051270 013711 001172 3$:  MOV      $TMP0,(R1)     ;RESTORE ORIGINAL DATA TO TEST LOCATION
6658 051274 062737 000100 172350 ADD      #100,KIPAR4    ;CHANGE BASE ADDRESS
6659 051302 062737 000100 172352 ADD      #100,KIPAR5    ;CHANGE BASE ADDRESS
6660 051310 005202      INC      R2             ;CHANGE DATA PATTERN
6661 051312 022737 007500 172352 CMP      #7500,KIPAR5   ;SEE IF PAST LAST ADDRESS
6662 051320 103327      BHS     1$             ;BRANCH IF NOT PAST LAST ADDRESS
6663 051322 005737 001304      TST     HOLFLG          ;SEE IF YOU ARE IN THE MIDDLE OF A HOLE
6664 051326 001401      BEQ     4$             ;BRANCH IF NOT IN MIDDLE OF A HOLE
6665 051330 104041      ERROR   41             ;IN MIDDLE OF A HOLE IN MEMORY
6666                                     ;HOLFLG HAS NO. OF TIME OUTS STMP1
6667                                     ;HAS PAR OF FIRST TIMEOUT
6668                                     .SBTTL      TEST 'SAPN UNIBUS ADRS L' IN 18-BIT MAPPING
6669 :*:
6670 :*:
6671 :* ADDRESS 760000 IS GENERATED BY CARRY PROPAGATION WHILE THE
6672 :* PROGRAM IS EXPECTING A TIME OUT. (760000 IS THE FIRST I/O
6673 :* PAGE ADDRESS AND IS NOT IMPLEMENTED EXCEPT FOR DIAGNOSTICS)
6674 :* KIPAR4 WILL CONTAIN 007577 AND R0 HAS 100100 (WHICH
6675 :* SELECTS KIPAR4 AND GENERATES ADDRESS 760000).
6676 :* IF THE WRONG BIT IN THE CPU ERROR REGISTER IS SET OR IF
6677 :* NO TRAP TO 'ERRVEC' OCCURS AN ERROR IS REPORTED.
6678 :*
6679 :*
6680 051332 012737 031670 000004 4$:  MOV      #CPUER,ERRVEC  ;RESTORE NORMAL ROUTINE FOR TRAPS THRU 4
6681 051340 012737 031774 000114      MOV      #MEMER,CACHVEC ;RESTORE NORMAL CACHE PARITY ERROR POINTER
6682 051346 012737 051400 001112      MOV      #40$,SLPERR    ;SET LOOP ON ERROR POINTER TO 40$
6683 051354 012737 007577 172350      MOV      #7577,KIPAR4   ;LOAD KIPAR4 WITH 7577
6684 051362 012702 007600      MOV      #7600,R2       ;LOAD DATA PATTERN INTO R2
6685 051366 012737 000020 001224      MOV      #20,CPUEXP     ;EXPECTING UNIBUS TIMEOUT
6686 051374 005037 001260      CLR     PCPUER          ;CLEAR CPU TRAP FLAG
6687 051400 000240      40$:  NOP                ;THIS A IS SYNC POINT FOR SCOPING
6688 051402 010210      MOV      R2,(R0)        ;LOAD 760000 THRU PAGE 4
6689                                     ;THIS INSTRUCTION SHOULD TIME OUT
6690                                     ;OVER THE UNIBUS.
6691 051404 005737 001260      TST     PCPUER          ;SEE IF TRAP OCCURRED
6692 051410 001001      BNE     6$             ;BRANCH IF TRAP
6693 051412 104043      ERROR   43             ;NO CPU TRAP
6694
6695 :*.SBTTL      TEST 'SAPN NOT CACHE ADRS H' 18-BIT MAPPING
6696 :*:
6697 :* ADDRESS 000000 IS GENERATED BY CARRY PROPAGATION, THIS WILL
6698 :* CAUSE '18BIT WRAPAROUND' TO BE ASSERTED, KNOCKING DOWN
6699 :* 'NOT CACHE ADRS'. THEN, IF THERE IS LESS THAN 120K OF MEMORY
6700 :* ON THE SYSTEM, THE SIZE REGISTER IS USED AS A PAR AND A
6701 :* CARRY IS PROPAGATED TO CAUSE '18BIT OVERFLOW' TO BE ASSERTED
6702 :* WHICH SHOULD GENERATE 'NOT CACHE ADRS'.
6703 :*
6704 :*
6705 051414 012737 051440 001112 6$:  MOV      #16$,SLPERR    ;SET LOOP ON ERROR POINTER TO 16$
6706 051422 005037 001224      CLR     CPUEXP          ;NO TRAPS THRU ERRVEC EXPECTED HERE
    
```

6707	051426	012737	007777	172350		MOV	#7777,KIPAR4	:LOAD PAR 4 WITH HIGHEST VALUE POSSIBLE
6708	051434	005037	000000			CLR	#000000	:CLEAR ADDRESS ZERO
6709	051440	000240			16\$:	NOP		:THIS A IS SYNC POINT FOR SCOPING
6710	051442	013701	100100			MOV	#100100,R1	:THIS SHOULD READ ADDRESS ZERO INTO R1
6711	051446	005701				TST	R1	:SEE IF YOU REALLY READ ADDRESS ZERO
6712	051450	001401				BEQ	7\$:BRANCH IF YOU READ ADDRESS ZERO
6713	051452	104044				ERROR	44	:DIDN'T READ ADDRESS ZERO
6714	051454	012737	051512	001112	7\$:	MOV	#17\$,SLPERR	:SET LOOP ON ERROR POINTER TO 17\$
6715	051462	022737	000170	031142		CMP	#170,\$LSTBK	:IS THERE 120K ON THE SYSTEM?
6716	051470	101421				BLOS	8\$:BRANCH IF MORE THAN 120K ON SYSTEM
6717	051472	012737	000040	001224		MOV	#40,CPUEXP	:EXPECTING CACHE NON-EXISTANT MEMORY
6718	051500	005037	001260			CLR	PCPUER	:CLEAR TRAP THRU ERRVEC FLAG
6719	051504	013737	177760	172350		MOV	SIZELO,KIPAR4	:GET READY TO GENERATE NON-EXIST ADDR.
6720	051512	000240			17\$:	NOP		:THIS A IS SYNC POINT FOR SCOPING
6721	051514	013701	100100			MOV	#100100,R1	:READ FROM NON-EXISTANT ADDRESS
6722	051520	005737	001260			TST	PCPUER	:SEE IF TRAP THRU ERRVEC OCCURRED
6723	051524	001003				BNE	8\$:BRANCH TO EXIT IF TRAP HAPPENED
6724	051526	012700	100100			MOV	#100100,R0	:SAVE VIRTUAL ADDRESS FOR ERROR TYPE OUT
6725	051532	104045				ERROR	45	:NO TRAP THRU ERRVEC
6726	051534	005037	001224		8\$:	CLR	CPUEXP	:NO CPU TRAPS EXPECTED
6727	051540	012737	051114	001112		MOV	#20\$,SLPERR	:SET LOOP POINTER TO START OF TEST
6728	051546	000472				BR	TST50	:BRANCH TO NEXT TEST
6729								
6730								::***** TRAP TO HERE THRU ERRVEC *****
6731								
6732	051550	012637	001306		10\$:	MOV	(KSP)+,OLDPC	:SAVE RETURN ADDRESS
6733	051554	012637	001310			MOV	(KSP)+,OLDPS	:SAVE OLD PROCESSOR STATUS
6734	051560	013737	177766	001260		MOV	CPUERR,PCPUER	:SAVE CPU ERROR REGISTER
6735	051566	022737	000040	001260		CMP	#40,PCPUER	:WAS TRAP NON-EXISTANT MEMORY
6736	051574	001012				BNE	12\$:BRANCH IF MEMORY EXISTS
6737	051576	023737	172350	177760		CMP	KIPAR4,SIZELO	:SEE IF PAR 4 MATCHES SIZE REGISTER
6738	051604	001004				BNE	11\$:BRANCH IF NO MATCH, POSSIBLE ERROR
6739								:IN COMPARE CIRCUIT
6740	051606	012737	051332	001306		MOV	#4\$,OLDPC	:CHANGE RETURN ADDRESS IF AT TOP OF MEM
6741	051614	000430				BR	14\$:BRANCH TO EXIT
6742	051616	104050			11\$:	ERROR	50	:COMPARE CIRCUIT FOR SIZE JUMPERS BAD
6743	051620	000426				BR	14\$:BRANCH TO EXIT & CONTINUE
6744								
6745								::***** COME HERE IF YOU DON'T GET CACHE NON-EXISTANT MEMORY ERROR
6746								::***** THERE MIGHT BE A HOLE IN MEMORY.
6747								
6748	051622	022737	051114	001260	12\$:	CMP	#20\$,PCPUER	:SEE IF ADDRESS TIMED OUT
6749	051630	001011				BNE	13\$:BRANCH IF NO TIME OUT, UNEXPECTED ERROR
6750	051632	005737	001304			TST	HOLFLG	:HAS THIS HAPPENED BEFORE?
6751	051636	001003				BNE	15\$:BRANCH IF NOT FIRST TIMEOUT
6752	051640	013737	172352	001174		MOV	KIPAR5,\$TMP1	:SAVE PAR WHEN HOLE FIRST DISCOVERED
6753	051646	005237	001304		15\$:	INC	HOLFLG	:KEEP COUNT OF CONSECUTIVE TIMEOUTS
6754	051652	000411				BR	14\$:BRANCH TO EXIT AND CONTINUE TEST
6755	051654	013737	001306	001262	13\$:	MOV	OLDPC,BADPC	:MOVE PC OF UNEXPECTED ERROR FOR TYPE OUT
6756	051662	104002				ERROR	2	:UNEXPECTED TRAP THRU ERRVEC
6757	051664	013737	001110	001306		MOV	\$LPADR,OLDPC	:RETURN TO START OF TEST
6758	051672	005037	001304			CLR	HOLFLG	:CLEAR HOLE FLAG IN CASE IT WAS SET
6759	051676	005037	177766		14\$:	CLR	CPUERR	:CLEAR THE CPU ERROR REGISTER
6760	051702	013746	001310			MOV	OLDPS,-(KSP)	:PUSH OLD PROCESSOR STATUS ON STACK
6761	051706	013746	001306			MOV	OLDPC,-(KSP)	:PUSH RETURN ADDRESS ON STACK
6762	051712	000002				RTI		:RETURN TO TEST AND CONTINUE

```

6763
6764
6765
6766 051714 012637 001306
6767 051720 012637 001310
6768 051724 012737 051332 001306
6769 051732 000761
6770
6771
6772
6773
6774
6775
6776
6777
6778
6779
6780
6781
6782
6783
6784 051734
6785 051734 000004
6786 051736 012737 052610 001316
6787
6788
6789
6790
6791
6792 051744 022737 007377 177760
6793 051752 003105
6794 051754 012737 000020 172516
6795 051762 012737 052424 000004
6796 051770 012737 052570 000114
6797 051776 005037 001304
6798 052002 012700 100100
6799 052006 012701 120000
6800 052012 012737 007377 172350
6801 052020 012737 007400 172352
6802 052026 012702 007400
6803 052032 012737 052106 001112
6804 052040 005037 001260
6805 052044 011137 001172
6806 052050 005737 001260
6807 052054 001014
6808 052056 005737 001304
6809 052062 001411
6810 052064 013737 172352 001176
6811 052072 012737 051762 001112
6812 052100 104125
6813 052102 005037 001304
6814 052106 000240
6815 052110 010210
6816 052112 011103
6817 052114 020203
6818 052116 001401

:***** COME HERE IF TRAP TO 114 (SIZELO HIGHER THAN ACTUAL MEMORY)
30$: MOV (KSP)+,OLDPC
MOV (KSP)+,OLDPS
MOV #4$,OLDPC
BR 14$ :BRANCH TO EXIT

:*****
*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
* ALSO SCANS MEMORY THIS TIME FROM ADDRESS 00740000
* TO 16740000 OR THE SIZE JUMPERS, ON 8K BOUNDARIES. AGAIN
* IF ANY HOLES ARE FOUND THE ADDRESS WHERE THEY ARE DISCOVERED
* AND THE FIRST GOOD ADDRESS AFTER THE HOLE WILL BE REPORTED
:*****
TST50: SCOPE
MOV #TST51,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS

:
: 22-BIT MAPPING STARTS HERE AND CONTINUES FOR THE REST OF
: THE PROGRAM.
:
:
: CMP #7377,SIZELO ;IS THERE AT LEAST 120K ON SYSTEM?
BGT 4$ ;BRANCH IF LESS THAN 120K
MOV #BIT4,MMR3 ;ENABLE 22-BIT MAPPING
20$: MOV #10$,ERRVEC ;SET ERRVEC POINTER TO 10$
MOV #30$,CACHVEC ;SET UP CACHE VECTOR POINTER FOR SIZELO TO HIGH (KB11-EM)
CLR HOLFLG ;START HOLE FLAG AT ZERO
MOV #100100,R0 ;LOAD VIRTUAL ADDRESS FOR PAGE 4
MOV #120000,R1 ;LOAD VIRTUAL ADDRESS FOR PAGE 5
MOV #7377,KIPAR4 ;LOAD BASE ADDRESS INTO PAR 4
MOV #7400,KIPAR5 ;LOAD BASE ADDRESS +100 INTO PARS
MOV #7400,R2 ;LOAD DATA PATTERN INTO R2
1$: MOV #2$,SLPERR ;SET LOOP ON ERROR POINTER TO 2$
CLR PCPUER ;CLEAR CPU TRAP FLAG
MOV (R1),STMP0 ;SAVE DATA AT TEST LOCATION USING PAGE 5
TST PCPUER ;SEE IF CPU TRAP OCCURRED
BNE 2$ ;BRANCH IF TRAP OCCURED
TST HOLFLG ;SEE IF A HOLE IN MEMORY WAS FOUND
BEQ 2$ ;BRANCH IF NO HOLE WAS FOUND
MOV KIPAR5,STMP2 ;SAVE PAR THAT POINTS TO END OF HOLE
MOV #20$,SLPERR ;SET LOOP ON ERROR POINTER TO 20$
ERROR 125 ;HOLE IN MEMORY FROM STMP1 TO STMP2
CLR HOLFLG ;CLEAR FLAG IN CASE OF MORE HOLES
2$: NOP ;THIS A IS SYNC POINT FOR SCOPING
MOV R2,(R0) ;WRITE DATA PATTERN INTO TEST LOCATION
MOV (R1),R3 ;READ TEST LOCATION VIA DIFFERENT VIRT.ADDR
CMP R2,R3 ;SEE IF DATA MATCHES
BEQ 3$ ;BRANCH IF DATA IS GOOD
    
```



```

6819 052120 104046
6820 052122 013711 001172
6821 052126 062702 000400
6822 052132 062737 000400 172350
6823 052140 062737 000400 172352
6824 052146 022737 167400 172352
6825 052154 103326
6826 052156 005737 001304
6827 052162 001401
6828 052164 104126
6829
6830
6831
6832
6833
6834
6835
6836
6837 052166 012737 000020 172516
6838 052174 012737 031670 000004
6839 052202 012737 031774 000114
6840 052210 012737 000020 001224
6841 052216 012737 167777 172350
6842 052224 012737 000000 172352
6843 052232 012702 017000
6844 052236 012737 052250 001112
6845 052244 011037 001172
6846 052250 000240
6847 052252 010210
6848 052254 011103
6849 052256 013710 001172
6850 052262 020203
6851 052264 001401
6852 052266 104047
6853
6854
6855
6856
6857
6858
6859
6860
6861
6862
6863 052270 012737 052314 001112
6864 052276 005037 001224
6865 052302 012737 177777 172350
6866 052310 005037 000000
6867 052314 000240
6868 052316 013701 100100
6869 052322 005701
6870 052324 001401
6871 052326 104044
6872 052330 012737 052366 001112
6873 052336 022737 167777 177760
6874 052344 101421

ERROR 46 :BAD RELOCATION 22-BIT MAPPING
MOV $TMP0,(R1) :RESTORE ORIGINAL DATA USING PAGE 5
ADD #400,R2 :CHANGE DATA PATTERN
ADD #400,KIPAR4 :GET READY TO TEST NEXT BIT
ADD #400,KIPAR5 :NEW PHYSICAL ADDRESS
CMP #167400,KIPAR5 :MAKE SURE YOU DON'T GET ON THE UNIBUS
BHS 1$ :BRANCH IF PAR 5 IS NOT PAST LIMIT
TST HOLFLG :SEE IF MEMORY ENDS WITH A HOLE
BEQ 4$ :BRANCH IF NO HOLE AT END OF MEMORY
ERROR 126 :HOLE AT END OF MEMORY

.SBTTL TEST 'SAPN UNIBUS ADRS L' IN 22-BIT MAPPING
:*
:* UNIBUS ADDRESS 000000 IS GENERATED BY CARRY PROPAGATION
:* SINCE THE MAP IS DISABLED THIS SHOULD REFERENCE PHYSICAL
:* ADDRESS 000000.
:*

4$: MOV #BIT4,MMR3 :ENABLE 22-BIT MAPPING
MOV #CPUER,ERRVEC :RESTORE NORMAL CPU TRAP ROUTINE
MOV #MEMER,CACHVEC :RESTORE NORMAL CACHE PARITY ERROR POINTER
MOV #20,CPUEXP :POSSIBLE U.B. TIME OUT
MOV #167777,KIPAR4 :GET READY TO TEST U.B. ADDRESS 0
MOV #0,KIPAR5 :SHOULD GO TO PHYSICAL ADDRESS 0
MOV #17000,R2 :LOAD DATA PATTERN INTO R2
MOV #5$,SLPERR :SET LOOP ON ERROR POINTER TO 6$
MOV (R0),$TMP0 :SAVE DATA IN LOCATION 0 USING PAGE 4
5$: NOP :THIS A IS SYNC POINT FOR SCOPING
MOV R2,(R0) :LOAD DATA PATTERN INTO TEST LOCATION
MOV (R1),R3 :READ TEST LOCATION VIA DIFFERENT V. A.
MOV $TMP0,(R0) :RESTORE ORIGINAL DATA USING PAGE 4
CMP R2,R3 :SEE IF DATA MATCHES
BEQ 6$ :BRANCH IF DATA IS GOOD
ERROR 47 :BAD RELOCATION, UNIBUS ADDRESS

.SBTTL TEST 'SAPN NOT CACHE ADRS H' 22-BIT MAPPING
:*
:* ADDRESS 000000 IS GENERATED BY CARRY PROPAGATION, THIS WILL
:* CAUSE '22BIT WRAPAROUND' TO BE ASSERTED, KNOCKING DOWN
:* 'NOT CACHE ADRS'. THEN THE SIZE REGISTER IS USED AS A PAR AND A
:* CARRY IS PROPAGATED TO CAUSE 'ADRS OVERFLOW' TO BE ASSERTED
:* WHICH SHOULD GENERATE 'NOT CACHE ADRS'.
:*

6$: MOV #16$,SLPERR :SET LOOP ON ERROR POINTER TO 16$
CLR CPUEXP :NO TRAPS THRU ERRVEC EXPECTED HERE
MOV #177777,KIPAR4 :LOAD PAR 4 WITH HIGHEST VALUE POSSIBLE
CLR #000000 :CLEAR ADDRESS ZERO
16$: NOP :THIS IS A SYNC POINT FOR SCOPING
MOV #100100,R1 :THIS SHOULD READ ADDRESS ZERO INTO R1
TST R1 :SEE IF YOU REALLY READ ADDRESS ZERO
BEQ 7$ :BRANCH IF YOU READ ADDRESS ZERO
ERROR 44 :DIDN'T READ ADDRESS ZERO
7$: MOV #17$,SLPERR :SET LOOP ON ERROR POINTER TO 17$
CMP #167777,SIZELO :IS SIZE REGISTER L.E. TO 167777
BLOS 8$ :BRANCH IF MAXIMUM MEMORY IS ON SYSTEM
    
```

```

6875 052346 012737 000040 001224      MOV      #40,CPUEXP      ;EXPECTING CACHE NON-EXISTANT MEMORY
6876 052354 005037 001260      CLR      PCPUER        ;CLEAR TRAP THRU ERRVEC FLAG
6877 052360 013737 177760 172350      MOV      SIZELO,KIPAR4 ;GET READY TO GENERATE NON-EXIST ADDR.
6878 052366 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
6879 052370 013701 100100 17$:      MOV      @#100100,R1    ;READ FROM NON-EXISTANT ADDRESS
6880 052374 005737 001260      TST      PCPUER        ;SEE IF TRAP THRU ERRVEC OCCURRED
6881 052400 001003      BNE      8$            ;BRANCH TO EXIT IF TRAP HAPPENED
6882 052402 012700 100100      MOV      #100100,R0    ;SAVE VIRTUAL ADDRESS FOR ERROR TYPE OUT
6883 052406 104045      ERROR   45            ;NO TRAP THRU ERRVEC
6884 052410 005037 001224 8$:      CLR      CPUEXP        ;NO CPU TRAPS EXPECTED
6885 052414 012737 051762 001112      MOV      #20$, $LPERR  ;SET LOOP POINTER TO START OF TEST
6886 052422 000472      BR       TST51        ;:BRANCH TO NEXT TEST
6887
6888
6889      ;:***** TRAP TO HERE THRU ERRVEC *****
6890
6891 052424 012637 001306 10$:      MOV      (KSP)+,OLDPC  ;SAVE RETURN ADDRESS
6892 052430 012637 001310      MOV      (KSP)+,OLDPS  ;SAVE OLD PROCESSOR STATUS
6893 052434 013737 177766 001260      MOV      CPUERR,PCPUER ;SAVE CPU ERROR REGISTER FOR TYPING
6894 052442 022737 000040 001260      CMP      #40,PCPUER    ;WAS TRAP NON-EXISTANT MEMORY
6895 052450 001012      BNE      12$          ;BRANCH IF MEMORY EXISTS
6896 052452 023737 172350 177760      CMP      KIPAR4,SIZELO ;SEE IF PAR 4 MATCHES SIZE REGISTER
6897 052460 001004      BNE      11$          ;BRANCH IF NO MATCH, POSSIBLE ERROR
6898                                ;IN COMPARE CIRCUIT
6899 052462 012737 052166 001306      MOV      #4$,OLDPC     ;CHANGE RETURN ADDRESS IF AT TOP OF MEM
6900 052470 000430      BR       14$          ;BRANCH TO EXIT
6901 052472 104050 11$:      ERROR   50            ;COMPARE CIRCUIT FOR SIZE JUMPERS BAD
6902 052474 000426      BR       14$          ;BRANCH TO EXIT & CONTINUE
6903
6904      ;:***** COME HERE IF YOU DON'T GET CACHE NON-EXISTANT MEMORY ERROR
6905      ;:***** THERE MIGHT BE A HOLE IN MEMORY.
6906
6907 052476 022737 051762 001260 12$:      CMP      #20$,PCPUER  ;SEE IF ADDRESS TIMED OUT
6908 052504 001011      BNE      13$          ;BRANCH IF NO TIME OUT, UNEXPECTED ERROR
6909 052506 005737 001304      TST      HOLFLG        ;HAS THIS HAPPENED BEFORE?
6910 052512 001003      BNE      15$          ;BRANCH IF NOT FIRST TIMEOUT
6911 052514 013737 172352 001174      MOV      KIPAR5,$TMP1  ;SAVE PAR WHEN HOLE FIRST DISCOVERED
6912 052522 005237 001304 15$:      INC      HOLFLG        ;KEEP COUNT OF CONSECUTIVE TIMEOUTS
6913 052526 000411      BR       14$          ;BRANCH TO EXIT AND CONTINUE TEST
6914 052530 013737 001306 001262 13$:      MOV      OLDPC,BADPC  ;MOVE PC OF UNEXPECTED ERROR FOR TYPE OUT
6915 052536 104002      ERROR   2            ;UNEXPECTED TRAP THRU ERRVEC
6916 052540 013737 001110 001306      MOV      $LPADR,OLDPC  ;RETURN TO START OF TEST
6917 052546 005037 001304      CLR      HOLFLG        ;CLEAR HOLE FLAG IN CASE IT WAS SET
6918 052552 005037 177766 14$:      CLR      CPUERR        ;CLEAR THE CPU ERROR REGISTER
6919 052556 013746 001310      MOV      OLDPS,-(KSP)  ;PUSH OLD PROCESSOR STATUS ON STACK
6920 052562 013746 001306      MOV      OLDPC,-(KSP)  ;PUSH RETURN ADDRESS ON STACK
6921 052566 000002      RTI                    ;RETURN TO TEST AND CONTINUE
6922
6923      ;:***** COME HERE IF TRAP TO 114 (SIZELO HIGHER THAN ACTUAL MEMORY)
6924
6925 052570 012637 001306 30$:      MOV      (KSP)+,OLDPC  ;
6926 052574 012637 001310      MOV      (KSP)+,OLDPS  ;
6927 052600 012737 052166 001306      MOV      #4$,OLDPC    ;
6928 052606 000761      BR       14$          ;BRANCH TO EXIT
6929
6930
    
```

6931
 6932
 6933
 6934
 6935
 6936
 6937
 6938
 6939
 6940
 6941
 6942
 6943
 6944
 6945
 6946
 6947
 6948
 6949
 6950
 6951
 6952
 6953
 6954
 6955 052610
 6956 052610 000004
 6957 052612 012737 053142 001316
 6958
 6959 052620 012737 000024 001206
 6960 052626
 6961 052626 012737 052766 001110
 6962 052634 012737 052766 001112
 6963 052642 012737 000051 001102
 6964 052650 013737 001102 177570
 6965 052656 012737 077406 172300
 6966 052664 012737 077406 172302
 6967 052672 012737 077406 12304
 6968 052700 012737 077406 172306
 6969 052706 012737 077406 172316
 6970 052714 012737 000000 172340
 6971 052722 012737 000200 172342
 6972 052730 012737 000400 172344
 6973 052736 012737 000600 172346
 6974 052744 012737 177600 172356
 6975 052752 012737 000001 177572
 6976 052760 012737 000020 172516
 6977 052766 012737 000006 172310
 6978 052774 012737 001000 172350
 6979 053002 012700 172311
 6980 053006 012737 053040 001112
 6981 053014 005037 001250
 6982 053020 012702 100000
 6983 053024 010203
 6984 053026 072327 177772
 6985
 6986 053032 042703 177600

.SBTTL ***** ENTRY POINT 5 --- STARTING ADDRESS 220 *****
 .SBTTL ***** MEMORY MANAGEMENT ABORTS AND TRAPS LOGIC TESTS *****
 :
 :*
 :* THIS GROUP OF TESTS CHECKS OUT THE MEMORY MANAGEMENT ABORT
 :* AND TRAP LOGIC ON PAGES 'SAPL', 'SSRC', AND 'SSRD'. IT WILL
 :* ALSO CHECK OUT BITS <07:01> OF MMR0, AND ALL BITS OF MMR2.
 :* IT THEN CHECKS THAT KERNEL MODE IS ALWAYS CLOCKED DURING
 :* A TRAP SEQUENCE SO THAT THE VECTOR IS PICKED UP FROM KERNEL
 :* SPACE.
 :*

 :TEST 51 PAGE LENGTH FAULTS - UPWARD EXPANSION
 :
 :* THIS TEST CHECKS OUT THE PAGE LENGTH COMPARATORS ON PAGE 'SAPL'
 :* AND THE LOGIC THAT GENERATES 'SAPL LENGTH FAULT'. IT TRIES
 :* EVERY PAGE LENGTH FIELD FROM 1 BLOCK TO 200(8) BLOCKS. THE
 :* TEST THEN TRIES A REFERENCE AT EVERY 100 BYTES UNTIL AN ABORT
 :* OCCURS. IF THE ABORT HAPPENS TOO SOON OR IF NO ABORT HAPPENS AT
 :* THE CORRECT BLOCK NUMBER AN ERROR IS REPORTED.
 :*

 :TST51:
 : SCOPE
 : MOV #TST52,NXTTST ;SAVE STARTING ADDRESS OF NEXT
 : ;TEST FOR ESCAPE ON PARITY ERRORS
 : ;DO 24 ITERATIONS
 : ENTPTS: MOV #24,STIMES
 : MOV #20\$,SLPADR ;SET LOOP ADDRESS POINTER TO 20\$
 : MOV #20\$,SLPERR ;SET LOOP ON ERROR POINTER TO 20\$
 : MOV #51,\$STSTNM ;LOAD TEST NUMBER INTO MEMORY
 : MOV \$STSTNM,DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
 : MOV #77406,KIPDR0 ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
 : MOV #77406,KIPDR1 ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
 : MOV #77406,KIPDR2 ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
 : MOV #77406,KIPDR3 ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
 : MOV #77406,KIPDR7 ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
 : MOV #000,KIPAR0 ;MAP KERNEL I PAGE 0 TO 0 - 4K
 : MOV #200,KIPAR1 ;MAP KERNEL I PAGE 1 TO 4K - 8K
 : MOV #400,KIPAR2 ;MAP KERNEL I PAGE 2 TO 8K - 12K
 : MOV #600,KIPAR3 ;MAP KERNEL I PAGE 3 TO 12K - 16K
 : MOV #177600,KIPAR7 ;MAP KERNEL I PAGE 7 TO THE I/O PAGE
 : MOV #BIT0,MMR0 ;ENABLE 18-BIT RELOCATION IF NOT ON
 : MOV #BIT4,MMR3 ;ENABLE 22-BIT RELOCATION IF NOT ON
 : 20\$: MOV #000006,\$KIPDR4 ;LOAD PDR4 FOR PAGE LENGTH OF 1
 : MOV #1000,KIPAR4 ;MAP PAGE 4 TO 16K
 : MOV #KIPDR4+1,R0 ;PUT ADDRESS OF PDR 4'S UPPER BYTE IN R0
 : 1\$: MOV #3\$,SLPERR ;SET LOOP ON ERROR POINTER TO 3\$
 : CLR MMR0 ;CLEAR LOCATION THAT HOLDS MMR0
 : MOV #100000,R2 ;PUT VIRTUAL ADDRESS INTO R2
 : 2\$: MOV R2,R3 ;PUT VIRTUAL ADDRESS INTO R3 TOO
 : ASH #6,R3 ;RIGHT SHIFT 6 BITS SO IT WILL
 : ;MATCH THE PLF
 : BIC #177600,R3 ;CLEAR BITS THAT ARE SET IN UPPER BYTE

J 11

```

7043 053240 000240      NOP
7044 053242 011204      MOV      (R2),R4
7045 053244 005037 001226  CLR      MMEXP
7046 053250 005737 001250  TST      MPRO
7047 053254 001405      BEQ      4$
7048 053256 005203      INC      R3
7049 053260 020103      CMP      R1,R3
7050 053262 001414      BEQ      6$
7051 053264 104051      ERROR   51
7052 053266 000412      BR       6$
7053 053270 020103 4$:  CMP      R1,R3
7054 053272 101402      BLOS    5$
7055 053274 104052      ERROR   52
7056 053276 000406      BR       6$
7057 053300 120327 000000 5$:  CMPB    R3,#000
7058 053304 001403      BEQ      6$
7059 053306 162702 000100  SUB     #100,R2
7060 053312 000741      BR       2$
7061 053314 122710 000000 6$:  CMPB    #000,(R0)
7062 053320 001402      BEQ      7$
7063 053322 105310      DECB    (R0)
7064 053324 000730      BR       1$
7065 053326 012737 053160 001112 7$:  MOV     #20$,$LPERR
    
```

```

:THIS IS A SYNC POINT FOR SCOPING
:READ USING V.A. IN R2
:CLEAR EXPECTED ABORT CONDITION
:SEE IF ABORT OCCURRED YET
:BRANCH IF NO ABORT YET, CHANGE V.A.
:MAKE R3 EQUAL TO PLF
:SEE IF PDR 4'S PLF=R3
:BRANCH IF ABORT HAPPENS AT RIGHT PLACE
:ABORT WRONG PLACE
:BRANCH TO CHANGE PLF
:SEE IF PAGE LENGTH FAULT SHOULD HAVE OCCURRED
:BRANCH IF NO PAGE LENGTH FAULT CONDITION
:NO ABORT, IT SHOULD HAVE HAPPENED THIS TIME
:BRANCH TO CHANGE PLF
:SEE IF V.A. IS 000
:BRANCH TO CHECK PLF
:CHANGE VIRTUAL ADDRESS
:GO TRY THE NEXT VIRTUAL ADDRESS
:SEE IF PDR 4'S PLF IS 000
:BRANCH TO EXIT IF PLF IS 000
:STEP PLF OF PDR 4 UP BY 1
:BRANCH TO START WITH V.A. OF 0
:SET LOOP POINTER TO START OF TEST
    
```

```

*****
:TEST 53      ACCESS CONTROL FIELD = 0, 3, OR 7 (ABORT ALL ACCESSES)
:
: THESE A.C.F.'S ARE ALL NON-RESIDENT, ANY REFERENCE (READ
: OR WRITE) TO A NON-RESIDENT PAGE SHOULD SET BIT 15 IN MPRO.
: BITS <06:01> IN MPRO WILL LOCK UP ALL AVAILABLE INFORMATION
: ON THAT PAGE. IN THIS CASE THE PAGE THAT CAUSES THE ABORT
: IS KERNEL I-SPACE PAGE 5. THE EXPECTED ERROR CODE IS 100013.
:
*****
    
```

```

7078 053334
7079 053334 000004
7080 053336 012737 053620 001316
7081
7082 053344 012737 077406 172310 20$: MOV     #77406,@MKIPDR4
7083 053352 012737 077400 172312  MOV     #77400,@MKIPDR5
7084
7085 053360 012737 001000 172350  MOV     #1000,@MKIPAR4
7086 053366 012737 001000 172352  MOV     #1000,@MKIPAR5
7087 053374 012700 100000      MOV     #100000,R0
7088 053400 012701 120000      MOV     #120000,R1
7089 053404 012702 161457      MOV     #161457,R2
7090 053410 012737 053420 001112 11$: MOV     #1$,$LPERR
7091 053416 005010      CLR     (R0)
7092 053420 012737 100013 001226 1$:  MOV     #100013,MMEXP
7093
7094 053426 000240      NOP
7095 053430 010211 8$:  MOV     R2,(R1)
7096
7097 053432 005037 001226      CLR     MMEXP
7098
    
```

```

:SCOPE
:SAVE STARTING ADDRESS OF NEXT
:TEST FOR ESCAPE ON PARITY ERRORS
:LOAD ACF 6 INTO PDR 4
:LOAD ACF 0 INTO PDR5
:LOAD 16K INTO PAR4
:LOAD 16K INTO PAR5
:LOAD VIRTUAL ADDRESS FOR PAR4 INTO R0
:LOAD VIRTUAL ADDRESS FOR PAR5 INTO R1
:LOAD DATA PATTERN INTO R2
:SET LOOP ON ERROR POINTER TO 1$
:CLEAR MEMORY LOCATION 100000
:LOAD EXPECTED ABORT CONDITION: NON-RESIDENT,
:KERNEL, I-SPACE, PAGE 5, FULL RELOCATION
:THIS IS A SYNC POINT FOR SCOPING
:WRITE TO NON-RESIDENT OR UNUSED ACF
:SHOULD CAUSE ABORT
:CLEAR EXPECTED ABORT CONDITION
    
```

```

7099 053436 005710 TST (R0) ;SEE IF (100000) IS STILL ZERO
7100 053440 001401 BEQ 2$ ;BRANCH IF (100000) IS ZERO
7101 053442 104053 ERROR 53 ;ABORT DID NOT HAPPEN
7102 053444 012737 053456 001112 2$: MOV #4$,SLPERR ;SET LOOP ON ERROR POINTER TO 4$
7103 053452 010210 MOV R2,(R0) ;LOAD DATA PATTERN INTO 100000
7104 053454 005004 CLR R4 ;CLEAR REGISTER TO RECEIVE DATA
7105 053456 012737 100013 001226 4$: MOV #100013,MMEXP ;LOAD EXPECTED ABORT CONDITION: NON-RESIDENT,
7106 ;KERNEL, I-SPACE, PAGE 5, FULL RELOCATION
7107 053464 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
7108 053466 011104 9$: MOV (R1),R4 ;TRY TO READ (100000) INTO R4
7109 ;THIS SHOULD ABORT
7110 053470 005037 001226 CLR MMEXP ;EXPECTED ABORT CONDITION
7111
7112 053474 005704 TST R4 ;MAKE SURE R4 IS STILL 0
7113 053476 001401 BEQ 5$ ;BRANCH IF R4 IS 0
7114 053500 104053 ERROR 53 ;ABORT DID NOT HAPPEN
7115 053502 023727 172312 077407 5$: CMP KIPDR5,#77407 ;SEE IF PDR 5'S ACF=7
7116 053510 001414 BEQ 12$ ;TEST OVER IF ACF=7
7117 053512 023727 172312 077403 CMP KIPDR5,#77403 ;SEE IF PDR 5'S ACF=3
7118 053520 001004 BNE 10$ ;BRANCH TO MAKE ACF=3 IF NOT 3
7119 053522 012737 077407 172312 MOV #77407,KIPDR5 ;MAKE ACF=7 IF ALREADY 3
7120 053530 000727 BR 11$ ;REPEAT TEST WITH ACF=7
7121 053532 012737 077403 172312 10$: MOV #77403,KIPDR5 ;MAKE PDR 5'S ACF=3
7122 053540 000723 BR 11$ ;REPEAT TEST WITH ACF=3
7123
7124 :::
7125 :::
7126 :::
7127 053542 012737 053570 001112 12$: MOV #13$,SLPERR ;SET LOOP ON ERROR POINTER TO 13$
7128 053550 012737 177600 172352 MOV #177600,KIPARS ;MAP PAGE 5 TO I/O PAGE
7129 053556 012701 132350 MOV #132350,R1 ;LOAD VIRTUAL ADDRESS TO REFERENCE
7130 ;KIPAR4
7131 053562 012737 100013 001226 MOV #100013,MMEXP ;EXPECTING NON-RESIDENT ABORT
7132 ;KERNEL I PAGE 5
7133 053570 000240 13$: NOP ;THIS IS A SYNC POINT FOR SCOPING
7134 053572 005011 CLR (R1) ;THIS INSTRUCTION SHOULD ABORT
7135 ;DURING THE ABORT 'SSRC INH T3' IS
7136 ;ASSERTED TO STOP THE CLOCKING OF THE
7137 ;FLIP/FLOPS ON SCCX
7138 053574 022737 001000 172350 CMP #1000,KIPAR4 ;KIPAR4 SHOULD STILL BE 1000
7139 053602 001401 BEQ 14$ ;BRANCH IF KIPAR4 IS STILL 1000
7140 053604 104127 ERROR 127 ;ABORT DID NOT STOP CLRING OF KIPAR4
7141 053606 012737 053344 001112 14$: MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
7142 053614 005037 001226 CLR MMEXP ;NOT EXPECTING ANY TRAPS
7143
7144
7145
7146
7147
7148
7149
7150
7151
7152
7153
7154

```

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

```

*****
*TEST 54 ACCESS CONTROL FIELD = 2 (ABORT ON WRITE)
*
* THIS IS A READ ONLY A.C.F., ANY WRITE ATTEMPT TO THIS PAGE
* WILL ABORT AND SET BIT 13 OF MMRO.
* BITS <06:01> IN MMRO WILL LOCK UP ALL AVAILABLE INFORMATION
* ON THAT PAGE. IN THIS CASE THE WRITE ATTEMPT WILL BE TO
* KERNEL I-SPACE PAGE 4. THE EXPECTED ERROR CODE IS 020011.
*****

```

```

7155 053620          TST54:
7156 053620 000004          SCOPE
7157 053622 012737 053750 001316      MOV      #TST55,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
7158                                     ;TEST FOR ESCAPE ON PARITY ERRORS
7159 053630          20$:
7160 053630 012737 001000 172352      MOV      #1000,KIPAR5      ;MAP PAGE 5 TO 16K
7161 053636 012737 001000 172350      MOV      #.000,KIPAR4      ;MAP PAGE 4 TO 16K
7162 053644 012737 077406 172312      MOV      #77406,KIPDR5     ;PAGE 5 IS 200 BLOCKS LONG,
7163                                     ;EXPANDS UPWARD, AND IS READ/WRITE
7164                                     ;WITH NO TRAPPING
7165 053652 012737 077402 172310      MOV      #77402,KIPDR4     ;LOAD ACF 2 INTO PDR 4
7166 053660 005037 001226                CLR      MMEXP              ;NOT EXPECTING ANY TRAPS OR ABORTS YET
7167 053664 012737 002222 120000      MOV      #2222,@#120000    ;LOAD DATA INTO 16K
7168 053672 013700 100000                MOV      @#100000,R0        ;READ DATA THRU PAGE 4
7169 053676 012737 053710 001112      1$:  MOV      #11$,SLPERR       ;SET LOOP ON ERROR POINTER TO 11$
7170 053704 005037 120000                CLR      @#120000          ;CLEAR TEST LOCATION THRU PAGE 5
7171 053710 012737 020011 001226      11$: MOV      #20011,MMEXP      ;EXPECTING READ ONLY FAULT, PAGE 4
7172 053716 000240                NOP                          ;THIS IS A SYNC POINT FOR SCOPING
7173 053720 012737 017777 100000      MOV      #17777,@#100000   ;TRY TO WRITE THRU PAGE 4
7174 053726 005037 001226                CLR      MMEXP              ;NO MORE TRAPS EXPECTED
7175 053732 005737 120000                TST      @#120000          ;SEE IF TEST LOCATION IS STILL ZERO
7176 053736 001401                BEQ      2$                 ;BRANCH IF WORD IS STILL ZERO
7177 053740 104054                ERROR   54                  ;NO ABORT ON PAGE 4
7178 053742 012737 053630 001112      2$:  MOV      #20$,SLPERR      ;SET LOOP POINTER TO START OF TEST
7179
7180
7181
7182
7183
7184
7185
7186
7187
7188
7189
7190
7191
7192
7193
7194
7195
7196
7197
7198
7199
    
```

```

:*****
:*TEST 55      ACCESS CONTROL FIELD = 1 (ABORT ON WRITE, TRAP ON READ)
:*
:*      THIS IS ANOTHER READ ONLY A.C.F., ALL WRITES TO THIS PAGE WILL
:*      ABORT AND SET BIT 13 OF MMRO.
:*      BITS <06:01> IN MMRO WILL LOCK UP ALL AVAILABLE INFORMATION
:*      ON THAT PAGE.  IN THIS CASE THE PAGE THAT CAUSES THE ABORT
:*      IS KERNEL I-SPACE PAGE 4.  THE EXPECTED ERROR CODE IS 020011.
:*
:*      IF BIT 09 OF MMRO (ENABLE MEMORY MANAGEMENT TRAPS) IS SET
:*      THEN ALL READS TO THIS PAGE WILL TRAP, AFTER THE INSTRUCTION
:*      IS COMPLETED, SETTING BIT 12 OF MMRO.
:*
:*      AFTER THE ABORT ON WRITE IS TESTED, A READ FROM THIS PAGE
:*      WITH BIT 9 CLEAR WILL BE DONE TO ENSURE THAT TRAPPING DOESN'T
:*      TAKE PLACE WHEN NOT ENABLED.  THEN BIT 09 IS SET AND ANOTHER
:*      READ IS DONE (THIS TIME IT SHOULD TRAP TO PAGE 1 KERNEL MODE).
:*
    
```

```

7200 053750          TST55:
7201 053750 000004          SCOPE
7202 053752 012737 054214 001316      MOV      #TST56,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
7203                                     ;TEST FOR ESCAPE ON PARITY ERRORS
7204 053760          20$:
7205 053760 012737 001000 172352      MOV      #1000,KIPAR5      ;MAP PAGE 5 TO 16K
7206 053766 012737 001000 172350      MOV      #1000,KIPAR4      ;MAP PAGE 4 TO 16K
7207 053774 012737 077406 172312      MOV      #77406,KIPDR5     ;PAGE 5 IS 200 BLOCKS LONG,
7208                                     ;EXPANDS UPWARD, AND IS READ/WRITE
7209                                     ;WITH NO TRAPPING
7210 054002 012737 077401 172310      MOV      #77401,KIPDR4     ;SET ACF = 1 FOR PAGE 4
    
```

```

CEKBEE0 11/70 MEM MGMT MACY11 30A(1052) 02-APR-80 09:15 PAGE 133 M 11
CEKBEE.P11 02-APR-80 08:48 T55 ACCESS CONTROL FIELD = 1 (ABORT ON WRITE, TRAP ON READ) SEQ 0142

7211 054010 012737 054022 001112 MOV #10$, $LPERR ;SET LOOP ON ERROR POINTER TO 10$
7212 054016 005037 001250 CLR PPMRO ;CLEAR M.M. ABORT FLAG
7213 054022 012737 020011 001226 10$: MOV #20011, MMEXP ;READ ONLY ABORT, PAGE 4
7214 054030 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
7215 054032 012737 017777 100000 MOV #17777, @#100000 ;TRY TO WRITE THRU PAGE 4
7216 054040 005037 001226 CLR MMEXP ;NO MORE TRAPS EXPECTED
7217 054044 005737 001250 TST PPMRO ;SEE IF M.M. ABORT HAPPENED
7218 054050 001001 BNE 1$ ;BRANCH IF ABORT HAPPENED
7219 054052 104054 ERROR 54 ;NO ABORT ON PAGE 4 A.C.F.=1
7220 054054 012737 054072 001112 1$: MOV #11$, $LPERR ;SET LOOP ON ERROR POINTER TO 11$
7221 054062 012700 012547 MOV #12547, R0 ;PUT DATA PATTERN INTO R0
7222 054066 010037 120000 MOV R0, @#120000 ;LOAD DATA PATTERN THRU PAGE 5
7223 054072 000240 11$: NOP ;THIS IS A SYNC POINT FOR SCOPING
7224 054074 013701 100000 MOV @#100000, R1 ;READ TEST LOCATION THRU PAGE 4
7225 054100 020100 CMP R1, R0 ;SEE IF DATA READ TOOK PLACE
7226 054102 001401 BEQ 2$ ;BRANCH IF READ CORRECT
7227 054104 104055 ERROR 55 ;INCORRECT READ, BIT09 (MMRO) IS CLEAR
7228 054106 013737 177572 001250 2$: MOV MMRO, PPMRO ;SAVE MMRO IN CASE OF ERROR
7229 054114 032737 010000 001250 BIT #BIT12, PPMRO ;BIT 12 SHOULD SET EVEN IF YOU DON'T
7230 ;TAKE THE TRAP DUE TO BIT09 BEING CLR
7231 054122 001001 BNE 3$ ;BRANCH IF BIT12 IS SET
7232 054124 104066 ERROR 66 ;BIT 12 WAS NOT SET
7233 054126 012737 054136 001112 3$: MOV #12$, $LPERR ;SET LOOP ON ERROR POINTER TO 12$
7234 054134 005001 CLR R1 ;CLEAR REG 1, RECEIVES DATA ON FETCH
7235 054136 005037 001250 12$: CLR PPMRO ;CLEAR M.M. TRAP FLAG
7236 054142 012737 001001 177572 MOV #1001, MMRO ;ENABLE TRAPS, FULL RELOCATION, CLR BIT 12
7237 054150 012737 011003 001226 MOV #11003, MMEXP ;EXPECTED TRAP CONDITION
7238 ;TRAP, TRAPS ENABLED, RELOCATION
7239 054156 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
7240 054160 013701 100000 MOV @#100000, R1 ;READ THRU PAGE 4
7241 054164 005037 001226 CLR MMEXP ;NO MORE TRAPS EXPECTED
7242 054170 005737 001250 TST PPMRO ;SEE IF TRAP OCCURED
7243 054174 001001 BNE 4$ ;BRANCH IF TRAP OCCURRED
7244 054176 104056 ERROR 56 ;NO TRAP
7245 054200 020001 4$: CMP R0, R1 ;SEE IF DATA READ WAS CORRECT
7246 054202 001401 BEQ 5$ ;BRANCH IF READ CORRECT
7247 054204 104057 ERROR 57 ;INCORRECT READ, BIT09 (MMRO) WAS SET
7248 054206 012737 053760 001112 5$: MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
7249
7250
7251 ;*****
7252 ;*TEST 56 ACCESS CONTROL FIELD = 4 (TRAP ON READ OR WRITE)
7253 ;*
7254 ;* THIS A.C.F. IS READ/WRITE BUT ALL REFERENCES TO THIS PAGE
7255 ;* WILL TRAP TO VECTOR 250 SETTING BIT 12 OF MMRO IF BIT 9
7256 ;* (ENABLE MEMORY MANAGEMENT TRAPS) IS SET.
7257 ;*
7258 ;* SINCE I HAVE ALREADY TESTED THE FACT THAT BIT 9 OF MMRO DOES
7259 ;* INDEED ENABLE M.M. TRAPS I WILL JUST SET BIT 9 AND VERIFY THAT
7260 ;* BOTH A READ AND A WRITE TO PAGE 4 A.C.F. = 4 TRAP CORRECTLY
7261 ;*
7262 ;*****
7263 054214 TST56:
7264 054214 000004 SCOPE
7265 054216 012737 054434 001316 MOV #TST57, NXTTST ;SAVE STARTING ADDRESS OF NEXT
7266 ;TEST FOR ESCAPE ON PARITY ERRORS

```



```

20$:
7267 054224
7268 054224 012737 001000 172352 MOV #1000,KIPAR5 ;MAP PAGE 5 TO 16K
7269 054232 012737 001000 172350 MOV #1000,KIPAR4 ;MAP PAGE 4 TO 16K
7270 054240 012737 077406 172312 MOV #77406,KIPDR5 ;PAGE 5 IS 200 BLOCKS LONG,
7271 ;EXPANDS UPWARD, AND IS READ/WRITE
7272 ;WITH NO TRAPPING
7273 054246 012737 077404 172310 MOV #77404,KIPDR4 ;SET ACF = 4 IN PDR 4
7274 054254 012700 013451 MOV #13451,R0 ;LOAD DATA PATTERN INTO R0
7275 054260 010037 120000 MOV R0,#120000 ;LOAD DATA INTO TEST LOCATION
7276 054264 005001 CLR R1 ;WHAT DO YOU THINK
7277 054266 012737 054274 001112 MOV #10$,SLPERR ;SET LOOP ON ERROR POINTER TO 10$
7278 054274 005037 001250 CLR PMPRO ;CLEAR FLAG LOCATION
7279 054300 012737 011003 001226 10$: MOV #11003,MPEXP ;EXPECTING TRAP WITH TRAPS ENABLED
7280 ;KERNEL I PAGE 1, FULL RELOCATION
7281 054306 012737 001001 177572 MOV #1001,MPRO ;ENABLE M. M. TRAPS
7282 054314 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
7283 054316 013701 100000 MOV #100000,R1 ;TRY TO READ THRU PAGE 4
7284 054322 005037 001226 CLR MPEXP ;NO MORE TRAPS EXPECTED
7285 054326 005737 001250 TST PMPRO ;SEE IF TRAP OCCURRED
7286 054332 001001 BNE 1$ ;BRANCH IF TRAP
7287 054334 104056 ERROR 56 ;NO TRAP
7288 054336 020001 1$: CMP R0,R1 ;SEE IF READ WAS CORRECT
7289 054340 001401 BEQ 2$ ;BRANCH IF CORRECT
7290 054342 104057 ERROR 57 ;INCORRECT READ, BIT09 (MPRO) WAS SET
7291 054344 012737 054352 001112 2$: MOV #12$,SLPERR ;SET LOOP ON ERROR POINTER TO 12$
7292 054352 005037 001250 12$: CLR PMPRO ;CLEAR FLAG LOCATION
7293 054356 012737 001001 177572 MOV #1001,MPRO ;ENABLE TRAPPING
7294 054364 012737 011003 001226 MOV #11003,MPEXP ;EXPECTING TRAP WITH TRAPS ENABLED
7295 ;KERNEL I PAGE 1, FULL RELOCATION
7296 054372 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
7297 054374 012737 000000 100000 MOV #0,#100000 ;TRY TO WRITE INTO PAGE 4
7298 054402 005037 001226 CLR MPEXP ;NOT EXPECTING ANY TRAPS
7299 054406 005737 001250 TST PMPRO ;SEE IF TRAP OCCURRED
7300 054412 001001 BNE 3$ ;BRANCH IF TRAP
7301 054414 104056 ERROR 56 ;NO TRAP
7302 054416 005737 120000 3$: TST #120000 ;SEE IF WRITE OCCURED
7303 054422 01401 BEQ 4$ ;BRANCH IF WRITE HAPPENED
7304 054424 104060 ERROR 60 ;NO WRITE, BIT09 (MPRO) WAS SET
7305 054426 012737 054224 001112 4$: MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
7306
7307
7308
7309
7310
7311
7312
7313
7314
7315
7316
7317
7318
7319
7320
7321
7322

```

```

*****
*TEST 57 ACCESS CONTROL FIELD = 5 (TRAP ON WRITE)
*
* THIS TEST IS RUN WITH THE ENABLE M.M. TRAPS BIT (BIT09) SET.
* THE A.C.F. FOR PAGE 4 IS SET TO 5 (TRAP ON WRITE).
* A READ FROM PAGE 4 IS TRIED EXPECTING NO TRAP AND THEN A WRITE
* TO PAGE 4 IS TRIED EXPECTING A M.M. TRAP. THE TRAP IS VERIFIED
* BY THE USE OF A TRAP FLAG WHICH IS SET IN THE TRAP ROUTINE.
*
*****

```

```

*****
TST57:
MOV #TST60,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS

```

20\$:

7327	054444	012737	001000	172352		MOV	#1000,KIPAR5	:MAP PAGE 5 TO 16K
7324	054452	012737	001000	172350		MOV	#1000,KIPAR4	:MAP PAGE 4 TO 16K
7325	054460	012737	077406	172312		MOV	#77406,KIPDR5	:PAGE 5 IS 200 BLOCKS LONG,
7326								:EXPANDS UPWARD, AND IS READ/WRITE
7327								:WITH NO TRAPPING
7328	054466	012737	077405	172310		MOV	#77405,KIPDR4	:SET ACF = 5 IN PDR 4
7329	054474	012700	012345			MOV	#12345,R0	:SET DATA PATTERN INTO R0
7330	054500	010037	120000			MOV	R0,@#120000	:LOAD TEST LOCATION WITH DATA
7331	054504	005037	001226			CLR	MMEXP	:NO TRAP EXPECTED
7332	054510	005001				CLR	R1	:CLEAR REGISTER 1
7333	054512	012737	054520	001112		MOV	#10\$,SLPERR	:SET LOOP ON ERROR POINTER TO 10\$
7334	054520	012737	001001	177572	10\$:	MOV	#1001,MMRO	:ENABLE M M TRAPS
7335	054526	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
7336	054530	013701	100000			MOV	@#100000,R1	:READ TEST LOCATION THRU PAGE 4
7337	054534	020001				CMR	R0,R1	:SEE IF READ WAS CORRECT
7338	054536	001401				BEQ	1\$:BRANCH IF READ CORRECT
7339	054540	104057				ERROR	57	:INCORRECT READ, BIT09 (MMRO) WAS SET
7340	054542	012737	054550	001112	1\$:	MOV	#11\$,SLPERR	:SET LOOP ON ERROR POINTER TO 11\$
7341	054550	005037	001250		11\$:	CLR	MMRO	:CLEAR FLAG LOCATION
7342	054554	012737	001001	177572		MOV	#1001,MMRO	:ENABLE TRAPS
7343	054562	012737	011003	001276		MOV	#1100\$,MMEXP	:EXPECTING M.M. TRAP
7344								
7345	054570	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
7346	054572	012737	000000	100000		MOV	#0,@#100000	:TRY TO WRITE INTO PAGE 4
7347	054600	005037	001226			CLR	MMEXP	:NO MORE TRAPS EXPECTED
7348	054604	005737	001250			TST	MMRO	:SEE IF TRAP OCCURRED
7349	054610	001001				BNE	2\$:BRANCH IF TRAP
7350	054612	104056				ERROR	56	:NO TRAP
7351	054614	005737	120000		2\$:	TST	@#120000	:SEE IF WRITE OCCURRED
7352	054620	001401				BEQ	3\$:BRANCH IF WRITE HAPPENED
7353	054622	104060				ERROR	60	:NO WRITE, BIT09 (MMRO) WAS SET
7354	054624	012737	054444	001112	3\$:	MOV	#20\$,SLPERR	:SET LOOP POINTER TO START OF TEST

```

*****
*TEST 60      NO TRAP WHEN TRAP BIT IS SET
*
*      THIS TEST VERIFIES THE LOGIC (ON 'SSRD') THAT PREVENTS A M.M.
*      TRAP AS LONG AS BIT12 OF MMRO (M.M. TRAP BIT) IS SET.  THE
*      TEST SETS BITS 12, 09, & 00 OF MMRO AND TRIES A REFERENCE TO
*      PAGE 4 WHOSE A.C.F.= 4 (TRAP ON ALL REFERENCES).  NO TRAP IS
*      EXPECTED, SO IF THE LOGIC FAILS AN UNEXPECTED M.M. TRAP WILL
*      OCCUR.
*****

```

```

TST60:
SCOPE
MOV      #TS161,NXTTST      :SAVE STARTING ADDRESS OF NEXT
                                :TEST FOR ESCAPE ON PARITY ERRORS
20$:
MOV      #1000,KIPAR5      :MAP PAGE 5 TO 16K
MOV      #1000,KIPAR4      :MAP PAGE 4 TO 16K
MOV      #77406,KIPDR5     :PAGE 5 IS 200 BLOCKS LONG,
                                :EXPANDS UPWARD, AND IS READ/WRITE
                                :WITH NO TRAPPING
MOV      #77404,KIPDR4     :SET ACF = 4 IN PDR 4
CLR      MMEXP              :NO TRAPS EXPECTED

```

7379	054676	012737	017777	120000		MOV	#17777,#120000	:LOAD DATA INTO TEST LOCATION
7380	054704	012737	054712	001112		MOV	#10\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 10\$
7381	054712	012737	011001	177572	10\$:	MOV	#11001, MPRO	:ENABLE TRAPS AND SET TRAP BIT (12)
7382	054720	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
7383	054722	012737	000000	100000		MOV	#0, #100000	:TRY TO WRITE THRU PAGE 4
7384	054730	012737	000001	177572		MOV	#1, MPRO	:CLEAR BITS 9 & 12 OF MPRO
7385	054736	005737	120000			TST	#120000	:SEE IF WORD WAS CHANGED
7386	054742	001401				BEQ	1\$:BRANCH IF LOCATION IS CLEAR
7387	054744	104060				ERROR	60	:NO WRITE, BIT09 (MPRO) WAS SET
7388	054746	012737	054642	001112	1\$:	MOV	#20\$, \$LPERR	:SET LOOP POINTER TO START OF TEST

7389
7390
7391
7392
7393

```

*****
*TEST 61      NO TRAPPING WHEN REFERENCING A MEMORY MANAGEMENT REG
*
*      THIS VERIFIES THE LOGIC THAT PREVENTS A M.M. TRAP WHEN
*      REFERENCING A M.M. REGISTER.  PAGE 7 IS MAPPED TO THE I/O
*      PAGE AND ITS A.C.F.= 4 (TRAP ON ALL REFERENCES).  EACH STATUS
*      REGISTER AND EACH PAR7 AND PDR7 IS READ THRU PAGE 7.  IF ANY
*      TRAPS OCCUR AN UNEXPECTED M.M. TRAP IS REPORTED.  THEN A MAP
*      REGISTER (170200) IS REFERENCED AND THE CORRECT TRAP IS VERIFIED
*      TO INSURE THAT A TRAP CAN OCCUR ON PAGE 7.
*      THE SIGNAL UNDER TEST IS 'SCCC INT REG B L'.
*****

```

7404	054754					TST61:	SCUPE		
7405	054754	000004					MOV	#TST62, NXTTST	:SAVE STARTING ADDRESS OF NEXT
7406	054756	012737	055204	001316					:TEST FOR ESCAPE ON PARITY ERRORS
7407									
7408	054764				20\$:				
7409	054764	012700	012754			MOV	#12754, R0	:LOAD DATA PATTERN INTO R0	
7410	054770	010037	170200			MOV	R0, MAPLO	:LOAD MAP REGISTER 0	
7411	054774	005037	001226			CLR	MPEXP	:NOT EXPECTING ANY TRAPS	
7412	055000	012737	077404	172316		MOV	#77404, KIPDR7	:SET ACF = 4 IN PAGE 7	
7413	055006	012737	001001	177572		MOV	#1001, MPRO	:ENABLE MEMORY MANAGEMENT TRAPS	
7414	055014	013701	172356			MOV	KIPAR7, R1	:READ KERNEL PAR 7	
7415	055020	013701	172256			MOV	SIPAR7, R1	:READ SUPERVISOR PAR 7	
7416	055024	013701	177656			MOV	UIPAR7, R1	:READ USER PAR 7	
7417	055030	013701	172316			MOV	KIPDR7, R1	:READ KERNEL PDR 7	
7418	055034	013701	172216			MOV	SIPDR7, R1	:READ SUPERVISOR PDR 7	
7419	055040	013701	177616			MOV	UIPDR7, R1	:READ USER PDR 7	
7420	055044	013701	177572			MOV	MPRO, R1	:READ MPRO	
7421	055050	013701	177574			MOV	MPR1, R1	:READ MPR1	
7422	055054	013701	177576			MOV	MPR2, R1	:READ MPR2	
7423	055060	013701	172516			MOV	MPR3, R1	:READ MPR3	
7424	055064	012737	001001	177572		MOV	#1001, MPRO	:MAKE SURE TRAPS ARE ENABLED	
7425	055072	005037	001250			CLR	MPRO	:CLEAR M.M. TRAP FLAG	
7426	055076	012737	011003	001226		MOV	#11003, MPEXP	:EXPECTING M.M. TRAP ON NEXT REFERENCE	
7427	055104	013701	170200			MOV	MAPLO, R1	:TRY TO READ MAP REGISTER 0	
7428	055110	005037	001226			CLR	MPEXP	:NOT EXPECTING ANY M.M. TRAPS	
7429	055114	005737	001250			TST	MPRO	:SEE IF TRAP OCCURRED	
7430	055120	001001				BNE	1\$:BRANCH IF TRAP OCCURED	
7431	055122	104061				ERROR	61	:NO TRAP	
7432	055124	020001			1\$:	CMP	R0, R1	:SEE IF DATA WAS READ CORRECTLY	
7433	055126	001401				BEQ	2\$:BRANCH IF DATA IS RIGHT	
7434	055130	104062				ERROR	62	:INCORRECT READ ON I/O PAGE	

```

7435 055132 012737 001000 177572 2$: MOV #BIT9,MMRO ;ENABLE TRAPS, NO RELOCATION
7436 055140 005037 001250 CLR PPMRO ;CLEAR M.M. TRAPS FLAG
7437 055144 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
7438 055146 013701 170200 MOV MAPLO,R1 ;READ MAP REGISTER 0
7439 ;THIS READ SHOULD NOT TRAP SINCE
7440 ;THERE IS NO RELOCATION ENABLED.
7441 055152 005737 001250 TST PPMRO ;SEE IF TRAP OCCURRED
7442 055156 001401 BEQ 3$ ;BRANCH IF NO TRAP
7443 055160 104063 ERROR 63 ;TRAPPED WHEN NO RELOCATION ENABLED
7444 055162 012737 000001 177572 3$: MOV #BIT0,MMRO ;ENABLE FULL RELOCATION
7445 055170 012737 077406 172316 MOV #77406,KIPDR7 ;SET PDR 7 TO NO TRAPPING ACF
7446 055176 012737 054764 001112 MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
7447
7448
7449
7450 *****
7451 *TEST 62 ONLY ONE VECTOR TAKEN IF TRAP AND ABORT
7452 *
7453 * IF THERE IS A M.M. TRAP CONDITION AND A M.M. ABORT ON THE
7454 * SAME INSTRUCTION ONLY ONE VECTOR TO 000250 SHOULD BE TAKEN.
7455 * 'SSRC KT ABORT FLG L' AND 'SSRC ABT FLG (0) H' WILL KNOCK
7456 * DOWN 'SSRD MEM MGMT TRAP L' SO THAT ONLY THE ABORT VECTOR WILL
7457 * BE TAKEN.
7458 * THIS TEST SETS THE VECTOR TO THE CODE AT 10$ AND, IF TWO VECTORS
7459 * ARE TAKEN ERROR 64 IS CALLED. PPMRO SHOULD REPORT BOTH THE TRAP
7460 * AND THE ABORT CONDITIONS WHICH ARE: PAGE LENGTH, KERNEL I-SPACE
7461 * PAGE 5, AND BIT12 OF PPMRO. (051013)
7462 *****
7463 TST62:
7464 055204 000004 SCOPE
7465 055206 012737 055402 001316 MOV #TST63,NXTTST ;SAVE STARTING ADDRESS OF NEXT
7466 ;TEST FOR ESCAPE ON PARITY ERRORS
7467 055214 012737 055260 001112 20$: MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
7468 055222 012737 055302 000250 MOV #10$,MMVEC ;SET M.M. VECTOR TO 10$
7469 055230 012737 001000 172350 MOV #1000,KIPAR4 ;MAP PAGE 4 TO 16K - 20K
7470 055236 012737 001000 172352 MOV #1000,KIPAR5 ;MAP PAGE 5 TO 16K- 20K
7471 055244 012737 000006 172312 MOV #000006,KIPDR5 ;SET PAGE LENGTH TO ONE FOR PAGE 5
7472 055252 012737 077404 172310 MOV #77404,KIPDR4 ;SET TRAP ON READ OR WRITE FOR PAGE 4
7473 055260 012706 001100 1$: MOV #KERSTK,KSP ;MAKE SURE KERN STK PTR IS SETUP IN
7474 ;CASE YOU LOOP ON ERROR
7475 055264 012737 001001 177572 MOV #1001,MMRO ;ENABLE M.M. TRAPS
7476 055272 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
7477 055274 013737 100000 120100 MOV @#100000,@#120100 ;TRY TO READ THRU PAGE 4 AND
7478 ;WRITE THRU PAGE 5 (PAGE 4 TRAP &
7479 ;PAGE 5 ABORT PAGE LENGTH)
7480 055302 020627 001074 10$: CMP KSP,#1074 ;HAS THE KERNEL STACK ONLY BEEN
7481 ;PUSHED ONCE BY THE PREVIOUS INSTRUCTION
7482 055306 001404 BEQ 12$ ;BRANCH IF IT HAS BEEN PUSHED
7483 ;ONLY ONE TIME
7484 055310 010637 001172 MOV KSP,STMP0 ;SAVE THE KERNEL STACK POINTER FOR TYPE OUT
7485 055314 104064 ERROR 64 ;TWO PUSHES WHEN ONLY ONE SHOULD HAPPEN
7486 055316 000413 BR 15$ ;BRANCH TO EXIT TEST
7487 055320 013737 177572 001250 12$: MOV PPMRO,PPMRO ;SAVE PPMRO FOR CHECK
7488 055326 012737 051013 001226 MOV #51013,MMEXP ;PPMRO SHOULD HAVE PAGE LENGTH,
7489 ;TRAP, ENABLE TRAP, PAGE 5, RELOCATING
7490 055334 023737 001250 001226 CMP PPMRO,MMEXP ;SEE IF ABORT CONDITION IS CORRECT
    
```

```

7491 055342 001401          BEQ      15$          ;BRANCH TO EXIT IF CORRECT
7492 055344 104065          ERROR    65          ;INCORRECT ABORT CONDITION
7493 055346 012716 055354      15$:    MOV      #16$, (KSP) ;CHANGE RETURN ADDRESS TO 16$
7494 055352 000006          RTT          ;RETURN TO 16$ AND CONTINUE PROGRAM
7495 055354 012737 032226 000250 16$:    MOV      #MMTRAP,MMVEC ;RESTORE TRAP HANDLER
7496 055362 012737 000001 177572      MOV      #BIT0,MMRO  ;CLEAR OUT MMRO, BUT LEAVE RELOC ON
7497 055370 005037 001226          CLR      MMEXP      ;NO! EXPECTING ANY M.M. TRAPS
7498 055374 012737 055214 001112      MOV      #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
7499
7500
7501
7502
7503
7504
7505
7506
7507
7508
7509
7510
7511
7512 055402
7513 055402 000004
7514 055404 012737 055554 001316          MOV      #TST64,NXTTST ;SAVE STARTING ADDRESS OF NEXT
7515
7516 055412 012737 000000 172350 20$:    MOV      #000,KIPAR4 ;TEST FOR ESCAPE ON PARITY ERRORS
7517 055420 012737 077404 172310      MOV      #77404,KIPDR4 ;MAP PAGE 4 TO 0 - 4K
7518 055426 012737 055450 001112      MOV      #1$,SLPERR  ;TRAP ALL REFERENCES
7519 055434 012700 001356          MOV      #ENMMTR,RO  ;SET LOOP ON ERROR POINTER TO 1$
7520
7521 055440 052700 100000          BIS      #BIT15,RO   ;PUT ADDRESS OF ENABLE M.M. TRAPS
7522 055444 005037 001226          CLR      MMEXP      ;WORD INTO RO
7523 055450 000240          1$:    NOP          ;MAKE ADDRESS IN RO USE PAGE 4
7524 055452 051037 177572      11$:   BIS      (RO),MMRO  ;NOT EXPECTING ANY TRAPS ON THIS REF.
7525
7526
7527 055456 013737 177572 001250      MOV      MMRO,MMRO  ;THIS IS A SYNC POINT FOR SCOPING
7528 055464 032737 010000 001250      BIT      #BIT12,MMRO ;WERE ON DURING SOURCE MODE
7529 055472 001001          BNE      2$          ;READ MMRO FOR CHECK ON BIT 12
7530 055474 104066          ERROR    66          ;SEE IF BIT12 IS SET BY INST. AT 11$
7531 055476 012737 055512 001112 2$:    MOV      #3$,SLPERR  ;BRANCH IF IT IS SET
7532 055504 012737 010003 001226      MOV      #10003,MMEXP ;BIT 12 NOT SET IN MMRO
7533
7534
7535 055512 012737 001001 177572 3$:    MOV      #1001,MMRO  ;SET LOOP ON ERROR POINTER TO 3$
7536 055520 005037 001250          CLR      MMRO      ;EXPECTING TRAP, BUT ENABLE TRAPS BIT
7537 055524 000240          NOP          ;SHOULD BE CLEAR BEFORE END OF INST.
7538 055526 041037 177572          BIC      (RO),MMRO  ;THAT CAUSES THE TRAP
7539
7540
7541 055532 005737 001250          TST      MMRO      ;ENABLE M.M. TRAPS, FULL RELOCATION
7542 055536 001001          BNE      4$          ;CLEAR M.M. TRAP FLAG
7543 055540 104067          ERROR    67          ;THIS IS A SYNC POINT FOR SCOPING
7544 055542 005037 001226 4$:    CLR      MMEXP      ;CLEAR ENABLE M.M. TRAP BIT AT END
7545 055546 012737 055412 001112      MOV      #20$,SLPERR ;OF INSTRUCTION, TRAP FLIP/FLOP SHOULD
7546

```

```

*****
*TEST 63      PROPER TIMING OF MEMORY MANAGEMENT TRAPS
*
* IF THE INSTRUCTION SETTING BIT09 OF MMRO SATISFIES A M.M. TRAP
* CONDITION, NO TRAP SHOULD OCCUR SINCE BIT09 WILL NOT BE SET
* WHEN THE TRAP CONDITION IS SATISFIED.
* THE SECOND HALF OF THIS TEST VERIFIES THAT IF THE M.M. TRAP
* CONDITION IS MET DURING THE INSTRUCTION WHICH CLEARS BIT 09
* OF MMRO THE TRAP WILL OCCUR ANYWAY.
*****

```

TST63:

7547
7548
7549
7550
7551
7552
7553
7554
7555
7556
7557
7558
7559
7560
7561
7562
7563
7564
7565
7566
7567
7568
7569
7570
7571
7572
7573
7574
7575
7576
7577
7578
7579
7580
7581
7582
7583
7584
7585
7586
7587
7588
7589
7590
7591
7592
7593
7594
7595
7596
7597
7598
7599
7600
7601
7602

*TEST 64 ABORT ON ILLEGAL MODE

* IF THE MODE SET IN BITS <15:14> OF THE PROCESSOR STATUS IS
* <10> THE MODE IS ILLEGAL AND THE NEXT INSTRUCTION FETCH WILL
* SELECT NO PAR/PDR PAIR TO CONTROL THE REFERENCE. THE PDR
* LINES WILL ALL BE READ AS ONES. THE A.C.F. = 7 (NON-RESIDENT),
* THE EXPANSION DIRECTION = DOWN, THE P.L.F. = 177 OR 1 BLOCK.
* THE M.M. ABORT WILL BE NON-RESIDENT, PAGE LENGTH (IF THE
* VIRTUAL ADDRESS HAS A BLOCK NUMBER OF 176 OR LESS), MODE <10>,
* PAGE 2 (SINCE THE CODE IS ON PAGE 2).

TST64:

7562	055554					SCOPE		
7563	055554	000004				MOV	#TST65,NXTTST	:SAVE STARTING ADDRESS OF NEXT
7564	055556	012737	055666	001316				:TEST FOR ESCAPE ON PARITY ERRORS
7565								:SET M.M. VECTOR TO 10\$
7566	055564	012737	055612	000250	20\$:	MOV	#10\$,MVEC	:SET LOOP ON ERROR POINTER TO 1\$
7567	055572	012737	055600	001112		MOV	#1\$,SLPERR	:THIS IS A SYNC POINT FOR SCOPING
7568	055600	000240			1\$:	NOP		:SET ILLEGAL MODE IN PROCESSOR STATUS
7569	055602	052737	100000	177776		BIS	#BIT15,PSW	:INSTRUCTION FETCH DIDN'T ABORT
7570	055610	104071				ERROR	71	:THIS INSTRUCTION FETCH SHOULD ABORT,
7571								:NON-RESIDENT & PAGE LENGTH FAULT,
7572								:ILLEGAL MODE, PAGE 1.
7573								
7574								
7575								
7576	055612	013737	177572	001250	10\$:	MOV	MPIRO,MPPIRO	:READ MPIRO FOR COMPARE
7577	055620	012716	055634			MOV	#16\$, (KSP)	:CHANGE RETURN ADDRESS TO 16\$
7578	055624	042766	100000	000002		BIC	#BIT15,2(KSP)	:CLEAR ILLEGAL MODE BIT IN PSW ON STACK
7579	055632	000006				RTT		:RETURN TO 16\$ AND CONTINUE PROGRAM
7580	055634	012701	140105		16\$:	MOV	#140105,R1	:LOAD EXPECTED ABORT CONDITION IN R1:
7581								:NON-RESIDENT, PAGE FAULT, MODE=<10>,
7582								:PAGE 2.
7583	055640	020137	001250			CMP	R1,MPPIRO	:DID YOU GET THE EXPECTED CONDITION
7584	055644	001401				BEQ	11\$:BRANCH IF CONDITION IS CORRECT
7585	055646	104072				ERROR	72	:WRONG ERROR CONDITION
7586	055650	000237			11\$:	SPL	7	:MAKE THE PRIORITY LEVEL 7
7587	055652	042737	177776	177572		BIC	#177776,MPPIRO	:CLEAR ALL ERROR CONDITIONS
7588	055660	012737	055564	001112		MOV	#20\$,SLPERR	:SET LOOP POINTER TO START OF TEST

*TEST 65 MEMORY MANAGEMENT REGISTERS ONLY CLOKED ONCE IF MPIRO NOT CLEARED

* AS LONG AS 'SSRC NO ERROR (1) H' IS NOT ASSERTED (THAT IS AFTER
* AN ABORT AND UNTIL MPIRO BITS <15:13> ARE CLEARED) MPIRO, MPIR1,
* AND MPIR2 SHOULD NOT BE CLOKED. THIS TEST CAUSES A NON-RESIDENT
* ABORT, SAVES THE STATUS REGISTERS, CHANGES THE VECTOR TO 10\$,
* AND THEN CAUSES A PAGE LENGTH ABORT. AT THE SECOND ABORT THE
* STATUS REGISTERS ARE COMPARED WITH THEIR FIRST CONDITIONS. IF ANY
* OF THEM CHANGE, THE OLD AND THE NEW CONDITIONS WILL BE REPORTED.

```

7603 055666 TS.05:
7604 055666 000004 SCOPE
7605 055670 012737 056130 001316 MOV #TST66,NXTTST ;SAVE STARTING ADDRESS OF NEXT
7606 ;TEST FOR ESCAPE ON PARITY ERRORS
7607 055676 012737 000000 172310 20$: MOV #000000,KIPDR4 ;MAP PAGE 4 NON-RESIDENT, AND PAGE
7608 ;LENGTH OF 1 BLOCK
7609 055704 012737 055746 000250 MOV #5$,MMVEC ;SET M.M. TRAP VECTOR TO 5$
7610 055712 012737 000340 000252 MOV #340,MMVEC+2 ;SET PRIORITY TO 7 GOTO KERNEL MODE
7611 055720 012737 055726 001112 MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
7612 055726 013700 100000 1$: MOV @#100000,R0 ;TRY TO READ THRU PAGE 4
7613 ;THIS PAGE NON-RESIDENT SHOULD CAUSE
7614 ;ABORT AND TRAP TO 5$.
7615 055732 012737 055776 000250 2$: MOV #10$,MMVEC ;SET M.M. TRAP VECTOR TO 10$
7616 055740 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
7617 055742 013700 100100 MOV @#100100,R0 ;TRY TO READ FROM BLOCK 2 OF PAGE 4
7618 ;THIS SHOULD ABORT AGAIN BUT NONE
7619 ;OF THE MEMORY MANAGEMENT STATUS
7620 ;REGISTERS SHOULD BE CLOKED THIS TIME.
7621
7622
7623 055746 013737 177572 001250 5$: MOV MPRO,MPRO ;READ MEMORY MANAGEMENT REGISTER 0
7624 055754 013737 177574 001252 MOV MPR1,MPR1 ;READ MEMORY MANAGEMENT REGISTER 1
7625 055762 013737 177576 001254 MOV MPR2,MPR2 ;READ MEMORY MANAGEMENT REGISTER 2
7626 055770 012716 055732 MOV #2$, (KSP) ;CHANGE RETURN ADDRESS TO 2$
7627 055774 000006 RTT ;GO BACK TO 2$ AND CAUSE PAGE FAULT
7628
7629
7630 055776 012716 056004 10$: MOV #16$, (KSP) ;CHANGE RETURN ADDRESS TO 16$
7631 056002 000006 RTT ;RETURN TO 16$ AND CONTINUE PROGRAM
7632 056004 005037 001200 16$: CLR $TMP3 ;ERROR COUNTER, 3 POSSIBLE CMP FAILURES
7633 056010 013737 177572 001172 MOV MPRO,$TMP0 ;READ MEMORY MANAGEMENT REGISTER 0
7634 056016 013737 177574 001174 MOV MPR1,$TMP1 ;READ MEMORY MANAGEMENT REGISTER 1
7635 056024 013737 177576 001176 MOV MPR2,$TMP2 ;READ MEMORY MANAGEMENT REGISTER 2
7636 056032 023737 001176 001254 CMP $TMP2,MPR2 ;SEE IF MPR2 CHANGED
7637 056040 001402 BEQ 11$ ;BRANCH IF MPR2 DIDN'T CHANGE
7638 056042 005237 001200 INC $TMP3 ;ONE COMPARE FAILURE
7639 056046 023737 001174 001252 11$: CMP $TMP1,MPR1 ;SEE IF MPR1 CHANGED
7640 056054 001402 BEQ 12$ ;BRANCH IF MPR1 DIDN'T CHANGE
7641 056056 005237 001200 INC $TMP3 ;ANOTHER COMPARE FAILURE
7642 056062 023737 001172 001250 12$: CMP $TMP0,MPRO ;SEE IF MPRO CHANGED
7643 056070 001402 BEQ 13$ ;BRANCH IF MPRO DIDN'T CHANGE
7644 056072 005237 001200 INC $TMP3 ;ANOTHER COMPARE FAILURE
7645 056076 005737 001200 13$: TST $TMP3 ;WERE THERE ANY ERRORS ON THIS TEST
7646 056102 001401 BEQ 19$ ;BRANCH IF NO ERRORS
7647 056104 104073 ERROR 73 ;AT LEAST ONE M.M. REG CHANGED
7648 056106 042737 177776 177572 19$: BIC #177776,MPRO ;CLEAR ALL ERROR BITS IN MPRO
7649 056114 012737 032226 000250 MOV #MMTRAP,MMVEC ;PUT BACK REGULAR M.M. TRAP ROUTINE
7650 056122 012737 055676 001112 MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
7651
7652
7653
7654
7655
7656
7657
7658

```

```

*****
*TEST 66 SUPERVISOR MODE, ABORT VECTOR FROM KERNEL SPACE
*
* THIS TEST DOES AN ABORT FROM SUPERVISOR MODE. THE VECTOR
* SHOULD BE PICKED UP FROM KERNEL I-SPACE DUE TO 'ROM OUT06'
* FORCING KERNEL MODE ON 'SSRB' DURING THE ABORT SEQUENCE.

```



```

7715 056422 000000 HALT ;EVER BE REACHED
7716 056424 000000 HALT ;EVER BE REACHED
7717 056426 000000 HALT ;EVER BE REACHED
7718 056430 000000 HALT ;EVER BE REACHED
7719 056432 000000 HALT ;EVER BE REACHED
7720 056434 012737 032226 000250 2$: MOV #MMTRAP,MMVEC ;RESTORE NORMAL M.M. TRAP ROUTINE
7721 056442 012737 056142 001112 MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
7722 056450 000431 BR .ST67 ;BRANCH TO NEXT TEST
7723 056452 000000 HALT ;THE NEXT 'SEVERAL' HALTS SHOULDN'T
7724 056454 000000 HALT ;EVER BE REACHED
7725 056456 000000 HALT ;EVER BE REACHED
7726 056460 000000 HALT ;EVER BE REACHED
7727 056462 000000 HALT ;EVER BE REACHED
7728 056464 012737 032456 000150 3$: MOV #KERVEC,<MMVEC-100> ;SET UP KERNEL SPACE VECTOR
7729 056472 012737 000340 000152 MOV #340,<MMVEC+2-100> ;KERNEL SPACE PSW = 340
7730 056500 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
7731 056502 013700 100000 MOV @#100000,R0 ;READ FROM PAGE 4, SUPERVISOR
7732 056506 022737 100051 001150 CMP #100051,<MMR0-100> ;EXPECTING NON-RESIDENT PAGE 4
7733 ;ABORT IN SUPERVISOR MODE
7734 056514 001401 BEQ 10$ ;BRANCH IF CORRECT COND
7735 056516 104075 ERROR 75 ;ABORT CONDITION INCORRECT
7736 056520 042737 177776 177572 10$: BIC #177776,MMR0 ;CLEAR MMR0 FOR NEXT ABORT
7737 056526 012737 000340 177776 MOV #340,PSW ;GO BACK INTO KERNEL MODE NOW
7738 ;THE NEXT INSTRUCTION TO BE EXECUTED
7739 ;IS AT LABEL 2$
7740
7741
7742
7743
7744
7745
7746
7747
7748
7749
7750
7751
7752
7753
7754
7755
7756
7757
7758
7759

```

```

*****
*TEST 67 M.M. ABORT DURING AN ODD ADDRESS ABORT SEQUENCE
*
* THIS TEST VERIFIES BIT07 OF MMR0 (INSTRUCTION COMPLETE) AND
* 'SSRA PS RESTORE (1) H'.
* THE TEST CAUSES AN 'ODD ADDRESS' ABORT THRU 'ERRVEC' WHICH
* PUTS THE PROCESSOR INTO SUPERVISOR MODE. THE SUPERVISOR
* STACK POINTER IS AT 1104 AND ITS PAGE 0 IS 11 BLOCKS LONG,
* SO WHEN THE OLD PS AND PC ARE PUSHED ON THE STACK A M.M.
* PAGE LENGTH ABORT OCCURS. THE OLD PS SHOULD BE RESTORED SO
* THAT THE PROPER RECOVERY CAN BE MADE, AND THE M.M. ABORT
* HAPPENS.
* MMR0 HAS PAGE LENGTH FAULT, SUPERVISOR I-SPACE, PAGE 0
* MMR1 HAS R6 DECREMENTED BY 2 TWICE
* MMR2 HAS 000004 (ADDRESS OF 'ERRVEC' WHERE M.M. ABORT OCCURRED)
*****

```

```

7760 056534 TST67: SCOPE
7761 056534 000004 MOV #TST70,NXTTST ;SAVE STARTING ADDRESS OF NEXT
7762 056536 012737 057120 001316 ;TEST FOR ESCAPE ON PARITY ERRORS
7763 ;BIT 4 OF P.S. IS T-BIT TRAPPING BIT
7764 .EQUIV BIT4,TBIT
7765 056544 012737 040340 000006 20$: MOV #40340,ERRVEC+2 ;SET PRIORITY OF ERRVEC TO 7
7766 ;AND MAKE NEW MODE SUPERVISOR
7767 056552 012737 000000 172240 MOV #000,SIPAR0 ;MAP SUPERVISOR PAGE 0 TO PHYS. 0
7768 056560 012737 000200 172242 MOV #200,SIPAR1 ;MAP SUPERVISOR PAGE 1 TO 4K -8K
7769 056566 012737 000400 172244 MOV #400,SIPAR2 ;MAP SUPERVISOR PAGE 2 TO 8K - 12K
7770 056574 012737 000600 172246 MOV #600,SIPAR3 ;MAP SUPERVISOR PAGE 3 TO 12K - 16K

```



```

7827 057044 104070          ERROR 70          ;AT LEAST ONE COMPARE FAILED
7828 057046 000237          SPL      7          ;NORMAL PRIORITY IS 7
7829 057050 052737 040000 177776 15$:  BIS      #BIT14,PSW ;GO TO SUPERVISOR MODE TO RESET PTR
7830 057056 012706 000700          MOV      #700,SSP   ;RESTORE SUPER STK PTR TO 700
7831 057062 042737 040000 177776          BIC      #BIT14,PSW ;RETURN TO KERNEL MODE
7832 057070 012737 000340 000006          MOV      #340,ERRVEC+2 ;RESTORE CORRECT PS TO ERROR VECTOR
7833 057076 012737 056544 001112          MOV      #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
7834 057104 042737 177776 177572          BIC      #177776,MMRO ;CLEAR ERROR CONDITION IN MMRO
7835 057112 012737 032226 000250          MOV      #MMTRAP,MMVEC ;RESTORE NORMAL M.M. VECTOR
    
```

```

7836
7837
7838
7839
7840
7841
7842
7843
7844
7845
7846
7847
7848
7849
7850
7851
7852
7853
7854
7855
    
```

 *TEST 70 USER MODE, ABORT VECTOR FROM KERNEL SPACE
 *
 * THIS TEST DOES AN ABORT FROM USER MODE. THE VECTOR
 * SHOULD BE PICKED UP FROM KERNEL I-SPACE DUE TO 'ROM OUT06'
 * FORCING KERNEL MODE ON 'SSRB' DURING THE ABORT SEQUENCE.
 *
 * THE 'HALTS' IN THIS TEST ARE SPACE FILLERS AND SHOULD NEVER BE
 * REACHED, IF USER MODE IS ENABLED PROPERLY ON 'SSRB', 'SAPE',
 * 'SAPB', 'SAPC', AND 'SAPF'.
 *
 * IT SHOULD BE NOTED THAT IF THIS TEST CODE IS EXECUTED IN SINGLE
 * INSTRUCTION, THE ABORT WILL PUSH THE PS & PC ONTO THE SUPERVISOR
 * STACK INSTEAD OF THE KERNEL STACK. THIS IS DUE TO A FLAW IN
 * THE CPU ROM AND IS A CARRY OVER FROM THE PDP-11/45. IF YOU NEED
 * TO SINGLE INSTRUCTION THIS TEST, SINGLE BUS CYCLE THRU THE ABORT
 * SEQUENCE FOR PROPER OPERATION OF THE STACKS.

```

7856
7857 057120
7858 057120 000004
7859 057122 012737 057522 001316
7860
7861 057130 005037 001226          20$: CLR      MMEXP
7862 057134 012737 077400 172202          MOV      #77400,SIPDR1 ;LOAD SUPERVISOR PAGE 1 NON-RESIDENT
7863 057142 012737 077400 172204          MOV      #77400,SIPDR2 ;LOAD SUPERVISOR PAGE 2 NON-RESIDENT
7864 057150 012737 077400 172206          MOV      #77400,SIPDR3 ;LOAD SUPERVISOR PAGE 3 NON-RESIDENT
7865 057156 012737 077400 172216          MOV      #77400,SIPDR7 ;LOAD SUPERVISOR PAGE 7 NON-RESIDENT
7866 057164 012700 077406          MOV      #77406,RO     ;PAGE LENGTH-200 BLOCKS EXPAND UP
7867
7868 057170 010037 172200          MOV      RO,SIPDR0    ;RESIDENT READ/WRITE
7869 057174 010037 177600          MOV      RO,UIPDR0   ;LOAD SUPERVISOR PAGE 0
7870 057200 010037 177602          MOV      RO,UIPDR1   ;LOAD USER PAGE 0
7871 057204 010037 177604          MOV      RO,UIPDR2   ;LOAD USER PAGE 1
7872 057210 010037 177606          MOV      RO,UIPDR3   ;LOAD USER PAGE 2
7873 057214 010037 177616          MOV      RO,UIPDR7   ;LOAD USER PAGE 3
7874 057220 012737 000002 172240          MOV      #2,SIPARO   ;LOAD USER PAGE 7
7875 057226 012737 177600 172256          MOV      #177600,SIPAR7 ;MAP SUPERVISOR PAGE 0 TO 00200
7876 057234 012737 000001 177640          MOV      #1,UIPARO   ;MAP SUPERVISOR PAGE 7 TO I/O PAGE
7877 057242 012737 000201 177642          MOV      #201,UIPAR1 ;MAP USER PAGE 0 TO 00100
7878 057250 012737 000401 177644          MOV      #401,UIPAR2 ;MAP USER PAGE 1 TO 20100
7879 057256 012737 000601 177646          MOV      #601,UIPAR3 ;MAP USER PAGE 2 TO 40100
7880 057264 012737 177600 177656          MOV      #177600,UIPAR7 ;MAP USER PAGE 3 TO 60100
7881 057272 012737 077400 177612          MOV      #77400,UIPAR5 ;MAP USER PAGE 7 TO I/O PAGE
7882 057300 012737 001000 177652          MOV      #1000,UIPAR5 ;MAKE USER PAGE 5 NON-RESIDENT
    
```

```

7883 057306 012737 032506 000450      MOV      #SUPVEC,450      ;SUPERVISOR SPACE VECTOR
7884 057314 012737 000140 000452      MOV      #140,452        ;SUPERVISOR SPACE PSW = 140
7885 057322 012737 032530 000350      MOV      #USEVEC,350     ;USER SPACE VECTOR
7886 057330 012737 000000 000352      MOV      #000,352       ;USER SPACE PSW = 000
7887 057336 012737 057344 001112      MOV      #5$, $LPERR     ;SET LOOP ON ERROR POINTER TO 5$
7888 057344 012737 140000 177776 5$:      MOV      #140000,PSW     ;GO TO USER MODE.
7889                                     ;THE NEXT INSTRUCTION EXECUTED IS AT
7890                                     ;3$. THE ADDRESS IS 100 OCTAL BYTES
7891                                     ;GREATER THAN THE ADDRESS AT 1$.
7892 057352 104076 177776 177776 1$:      ERROR   76              ;DIDN'T GO TO USER MODE
7893 057354 005037 057422 057422      CLR      PSW             ;GO BACK INTO KERNEL MODE
7894 057360 000137 057422 057422      JMP      2$              ;GO TO EXIT OF TEST
7895 057364 000000                                     HALT                                     ;THE NEXT 'SEVERAL' HALTS SHOULDN'T
7896 057366 000000                                     HALT                                     ;EVER BE REACHED
7897 057370 000000                                     HALT                                     ;EVER BE REACHED
7898 057372 000000                                     HALT                                     ;EVER BE REACHED
7899 057374 000000                                     HALT                                     ;EVER BE REACHED
7900 057376 000000                                     HALT                                     ;EVER BE REACHED
7901 057400 000000                                     HALT                                     ;EVER BE REACHED
7902 057402 000000                                     HALT                                     ;EVER BE REACHED
7903 057404 000000                                     HALT                                     ;EVER BE REACHED
7904 057406 000000                                     HALT                                     ;EVER BE REACHED
7905 057410 000000                                     HALT                                     ;EVER BE REACHED
7906 057412 000000                                     HALT                                     ;EVER BE REACHED
7907 057414 000000                                     HALT                                     ;EVER BE REACHED
7908 057416 000000                                     HALT                                     ;EVER BE REACHED
7909 057420 000000                                     HALT                                     ;EVER BE REACHED
7910 057422 012737 032226 000250 2$:      MOV      #MMTRAP,MMVEC   ;RESTORE NORMAL M.M. TRAP ROUTINE
7911 057430 012737 057130 001112      MOV      #20$, $LPERR   ;SET LOOP POINTER TO START OF TEST
7912 057436 000431                                     BR      TST71            ;BRANCH TO NEXT TEST
7913 057440 000000                                     HALT                                     ;THE NEXT 'SEVERAL' HALTS SHOULDN'T
7914 057442 000000                                     HALT                                     ;EVER BE REACHED
7915 057444 000000                                     HALT                                     ;EVER BE REACHED
7916 057446 000000                                     HALT                                     ;EVER BE REACHED
7917 057450 000000                                     HALT                                     ;EVER BE REACHED
7918 057452 012737 032456 000150 3$:      MOV      #KERVEC,<MMVEC-100> ;SET UP KERNEL SPACE VECTOR
7919 057460 012737 000340 000152      MOV      #340,<MMVEC+2-100> ;KERNEL SPACE PSW = 340
7920 057466 000240                                     NOP                                     ;THIS IS A SYNC POINT FOR SCOPING
7921 057470 013700 120000                                     MOV      @#120000,R0      ;READ FROM PAGE 5, USER SPACE
7922 057474 022737 100153 001150      CMP      #100153,<MMRO-100> ;EXPECTING NON-RESIDENT PAGE 5
7923                                     ;ABORT IN USER MODE I-SPACE
7924 057502 001401                                     BEQ      10$              ;BRANCH IF CORRECT COND
7925 057504 104077                                     ERROR   77              ;ABORT CONDITION INCORRECT
7926 057506 042737 177776 177572 10$:      BIC      #177776,MMRO    ;CLEAR MMRO FOR NEXT TEST
7927 057514 012737 000340 177776      MOV      #340,PSW       ;GO BACK INTO KERNEL MODE NOW
7928                                     ;THE NEXT INSTRUCTION TO BE EXECUTED
7929                                     ;IS AT LABEL 2$

```

```

*****
*TEST 71      COUNT PATTERN THRU MMR2, TO TEST ALL BITS
*
*      THIS TEST SETS UP ALL USER I-SPACE PAGES TO BE NON-RESIDENT
*      AND THEN TRIES ALL POSSIBLE VIRTUAL ADDRESSES AS A PROCESSOR
*      COUNTER IN USER MODE. EVERY INSTRUCTION FETCH WILL ABORT,
*      NON RESIDENT AND THE CONTENTS OF MMR2 IS TESTED ALONG

```

7930
7931
7932
7933
7934
7935
7936
7937
7938

```

7939          ;*      WITH BITS <06:01> OF MMR0.
7940          ;*
7941          ;*
7942          ;*****
7943          TST71:
7944          SCOPE
7945          MOV      #TST72,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
7946          MOV      #2,$TIMES          ;TEST FOR ESCAPE ON PARITY ERRORS
7947          MOV      #77400,UIPDR0     ;DO 2 ITERATIONS
7948          MOV      #77400,UIPDR1     ;MAP USER PAGE 0 NON-RESIDENT
7949          MOV      #77400,UIPDR2     ;MAP USER PAGE 1 NON-RESIDENT
7950          MOV      #77400,UIPDR3     ;MAP USER PAGE 2 NON-RESIDENT
7951          MOV      #77400,UIPDR4     ;MAP USER PAGE 3 NON-RESIDENT
7952          MOV      #77400,UIPDR5     ;MAP USER PAGE 4 NON-RESIDENT
7953          MOV      #77400,UIPDR6     ;MAP USER PAGE 5 NON-RESIDENT
7954          MOV      #77400,UIPDR7     ;MAP USER PAGE 6 NON-RESIDENT
7955          MOV      #2$,SLPERR        ;MAP USER PAGE 7 NON-RESIDENT
7956          MOV      #140000,R0        ;SET LOOP ON ERROR POINTER TO 2$
7957          CLR      R1                 ;THIS WILL FORCE USER MODE WHEN USED
7958          MOV      #10$,MMVEC        ;AS A PROCESSOR STATUS
7959          MOV      #340,MMVEC+2      ;R1 HOLDS USER VIRTUAL PC
7960          MOV      R0,-(KSP)         ;SET M.M. TRAP VECTOR TO 10$
7961          MOV      R1,-(KSP)         ;MAKE SURE TRAP TAKES YOU TO KERNEL
7962          NOP                       ;PUSH USER PS ON STACK
7963          RTI                        ;PUSH USER'S VIRTUAL PC ON STACK
7964          ;THIS IS A SYNC POINT FOR SCOPING
7965          ;RETURN TO USER MODE
7966          ;THE FIRST INSTRUCTION FETCH WILL
7967          ;CAUSE A NON-RESIDENT ABORT AND LOCK
7968          ;MMR2 SO THAT ALL BITS CAN BE CHECKED.
7969
7970
7971          10$:  ADD      #4,KSP          ;CLEAN UP STACK FOR NEXT TIME
7972          MOV      MMR2,PMMR2        ;READ MMR2 TO TEMP LOCATION
7973          MOV      MMR0,PMMR0        ;READ MMR0 TO TEMP LOCATION
7974          CMP      R1,PMMR2         ;SEE IF MMR2 LOCKED CORRECT V. A.
7975          BEQ      11$              ;BRANCH IF MMR2 HAS RIGHT V.A.
7976          ERROR   100                ;WRONG V.A.
7977          11$:  CLR      R2           ;R2 WILL GET THE VIRTUAL PAGE NO.
7978          MOV      R1,R3            ;COPY VIRTUAL ADDRESS INTO R3
7979          ASHC    #3,R2            ;COMBINED LEFT SHIFT <R2,R3> 3 BITS
7980          ROL     R2                ;ADJUST PAGE NUMBER
7981          BIS     #100141,R2        ;SET OTHER EXPECTED BITS IN MMR0
7982          CMP     R2,PMMR0         ;SEE IF MMR0 RECORDED CORRECT PAGE NO.
7983          BEQ     12$              ;BRANCH IF PAGE NUMBER WAS CORRECT
7984          ERROR   101                ;WRONG PAGE NO. IN MMR0
7985          12$:  BIC     #17776,MMR0  ;CLEAR ALL ERROR BITS IN MMR0
7986          ADD     #2,R1             ;TRY NEXT VIRTUAL ADDRESS
7987          BNE     2$                ;BRANCH IF NOT ALL DONE
7988          MOV     MMRTRAP,MMVEC     ;PUT BACK REGULAR M.M. TRAP ROUTINE
7989          MOV     #20$,SLPERR       ;SET LOOP POINTER TO START OF TEST
7990
7991          .SBTTL ***** ENTRY POINT 6 --- STARTING ADDRESS 224 *****
7992          .SBTTL ***** D-SPACE TESTS, CORRECT TIMING OF I & D SPACE *****
7993          ;*
7994
    
```

7995
7996
7997
7998
7999
8000
8001
8002
8003
8004
8005
8006
8007
8008
8009
8010
8011
8012
8013
8014
8015
8016
8017
8018
8019
8020
8021
8022
8023 057766
8024 057766 000004
8025 057770 012737 060562 001316
8026
8027 057776 104420
8028
8029 060000
8030 060000 012737 060140 001110
8031 060000 012737 060140 001112
8032 060001 012737 000072 001102
8033 060022 013737 001102 177570
8034 060030 012737 077406 172300
8035 060036 012737 077406 172302
8036 060044 012737 077406 172304
8037 060052 012737 077406 172306
8038 060060 012737 077406 172316
8039 060066 012737 000000 172340
8040 060074 012737 000200 172342
8041 060102 012737 000400 172344
8042 060110 012737 000600 172346
8043 060116 012737 177600 172356
8044 060124 012737 000001 177572
8045 060132 012737 000020 172516
8046 060140 012737 077400 172310
8047 060146 012737 077406 172320
8048 060154 012737 077406 172322
8049 060162 012737 077406 172330
8050 060170 012737 077400 172324

THIS GROUP OF TESTS CHECKS THE PROPER ENABLING OF D-SPACE. IT TESTS THAT I-SPACE IS FORCED DURING INSTRUCTION FETCHES AND ADDRESS, INDEX, OR OPERAND FETCHES IF THE REGISTER FIELD IS 7. IT ALSO CHECKS THAT TRAPS PICK UP THE VECTOR FROM D-SPACE, IF IT IS ENABLED.

THROUGHOUT THIS AREA OF TESTING D-SPACE PAGES 2 & 3 ARE MAPPED NON-RESIDENT, AND I-SPACE PAGE 4 IS ALSO MAPPED NON-RESIDENT. ALL OTHER PAGES IN BOTH I & D SPACE ARE MAPPED RESIDENT, 4K, READ/WRITE. IF ANY ABORTS SHOULD OCCUR DURING THESE TESTS THEY WILL VECTOR TO 'NODSPAC'. IF THE OFFENDING PAGE IS 2 OR 3 THEN THE FAULT IS THAT I-SPACE WASN'T FORCED WHEN IT SHOULD HAVE BEEN, BUT IF THE PAGE IS 4 THEN THE FAULT IS THAT I-SPACE WAS FORCED WHEN IT SHOULD NOT HAVE BEEN.

 :TEST 72 ENABLE KERNEL D-SPACE AND SEE THAT I-SPACE IS FORCED

THIS TEST SHOWS THAT I-SPACE IS FORCED BY EITHER 'SSRB I SPACEA L' OR SSRB I SPACEB L' DURING THE PROPER TIMES

ALL ERRORS FOUND IN THIS TEST ARE REPORTED WHEN THE C.P.U. ABORTS THRU 'MMVEC' TO SUBROUTINE 'NODSPAC'. THIS SUBROUTINE WILL REPORT THAT D-SPACE WAS NOT ENABLED PROPERLY.

 TST72:

SCOPE
 MOV #TST73,NXTTST ;SAVE STARTING ADDRESS OF NEXT
 ;TEST FOR ESCAPE ON PARITY ERRORS
 TBTR ;RESTORE THE T-BIT TO ITS CONDITION
 ;BEFORE THE LAST FOUR TESTS.

ENTPT6:
 MOV #20\$,SLPADR ;SET LOOP ADDRESS POINTER TO 20\$
 MOV #20\$,SLPERR ;SET LOOP ON ERROR POINTER TO 20\$
 MOV #72,\$TSTNM ;LOAD TEST NUMBER INTO MEMORY
 MOV \$TSTNM,DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
 MOV #77406,KIPDR0 ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
 MOV #77406,KIPDR1 ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
 MOV #77406,KIPDR2 ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
 MOV #77406,KIPDR3 ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
 MOV #77406,KIPDR7 ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
 MOV #000,KIPAR0 ;MAP KERNEL I PAGE 0 TO 0 - 4K
 MOV #200,KIPAR1 ;MAP KERNEL I PAGE 1 TO 4K - 8K
 MOV #400,KIPAR2 ;MAP KERNEL I PAGE 2 TO 8K - 12K
 MOV #600,KIPAR3 ;MAP KERNEL I PAGE 3 TO 12K - 16K
 MOV #177600,KIPAR7 ;MAP KERNEL I PAGE 7 TO THE I/O PAGE
 MOV MBIT0,MMR0 ;ENABLE 18-BIT RELOCATION IF NOT ON
 MOV MBIT4,MMR3 ;ENABLE 22-BIT RELOCATION IF NOT ON
 20\$:
 MOV #77400,KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
 MOV #77406,KDPDR0 ;MAKE KERNEL D PAGE 0 200 BLOCKS R/W
 MOV #77406,KDPDR1 ;MAKE KERNEL D PAGE 1 200 BLOCKS R/W
 MOV #77406,KDPDR4 ;MAKE KERNEL D PAGE 4 200 BLOCKS R/W
 MOV #77400,KDPDR2 ;MAKE KERNEL D PAGE 2 NON-RESIDENT

```

8051 060176 012737 077400 172326      MOV      #77400,KDPDR3      ;MAKE KERNEL D PAGE 3 NON-RESIDENT
8052 060204 012737 000000 172360      MOV      #000,KDPAR0      ;MAP KERNEL D PAGE 0 TO PHYSICAL 0
8053 060212 012737 000200 172362      MOV      #200,KDPAR1      ;MAP KERNEL D PAGE 1 TO 4K - 8K
8054 060220 012737 001000 172370      MOV      #1000,KDPAR4      ;MAP KERNEL D PAGE 4 TO 16K
8055 060226 012737 077406 172336      MOV      #77406,KDPDR7      ;MAP KERNEL D PAGE 7 TO 200 BLOCKS R/W
8056 060234 012737 177600 172376      MOV      #177600,KDPAR7    ;MAP KERNEL D PAGE 7 TO I/O PAGE
8057 060242 012704 172516                MOV      #MMR3,R4          ;PUT ADDRS OF MMR3 IN R4
8058 060246 012705 000004                MOV      #BIT2,R5          ;PUT KERNEL D-SPACE ENABLE BIT INTO R5
8059 060252 012700 000002                MOV      #2,R0             ;LOAD A TWO INTO R0
8060 060256 012737 032346 000250      MOV      #NODSPAC,MMVEC    ;SET M.M. VECTOR TO D-SPACE SERVICE ROUTINE
8061 060264 012737 060274 001112      MOV      #10$,SLPERR       ;SET LOOP ON ERROR POINTER TO 10$
8062 060272 050514                BIS      R5,(R4)           ;ENABLE D-SPACE MAPPING IN KERNEL MODE
8063
8064
8065
8066
8067 060274 000400      10$: BR      1$            ;BRANCH, USE FET.00
8068 060276 000244      1$: CLZ                ;CLEAR ZERO BIT IN PROCESSOR STATUS
8069 060300 001776      BEQ      1$            ;NO BRANCH, USE FET.13
8070 060302 000264      2$: SEZ                ;SET ZERO BIT IN PROC. STATUS
8071 060304 001376      BNE      2$            ;NO BRANCH, USE FET.12
8072 060306 000270      3$: SEN                ;SET NEGATIVE BIT IN PROC. STATUS
8073 060310 100376      BPL      3$            ;NO BRANCH, USE FET.11
8074 060312 000237      4$: SPL      7          ;SET PRIOR TO 7: USE SPL.10, GOTO FET.10
8075 060314 005700      TST      R0            ;USE TST.10, GOTO FET.10
8076 060316 077003      SOB      R0,4$        ;BRANCH UNTIL R0 IS ZERO:
8077
8078 060320 073002      ASHC     R2,R0         ;USE SOB.20, GOTO FET.10
8079
8080 060322 005200      INC      R0            ;COMBINED SHIFT, SHIFT COUNT ZERO
8081 060324 073002      ASHC     R2,R0         ;USE ASC.80, GOTO FET.10
8082
8083 060326 005300      DEC      R0            ;MAKE R0 POSITIVE ONE
8084 060330 005300      DEC      R0            ;MAKE R0 NEGATIVE ONE
8085 060332 073002      ASHC     R2,R0         ;LEFT COMBINED SHIFT ONE PLACE
8086
8087 060334 072001      ASH      R1,R0         ;RIGHT COMBINED SHIFT ONE PLACE
8088
8089 060336 005200      INC      R0            ;USE ASC.60, GOTO FET.10
8090 060340 005200      INC      R0            ;RIGHT SHIFT ONE PLACE
8091 060342 072001      ASH      R1,R0         ;GOTO FET.07
8092
8093 060344 070001      MUL      R1,R0         ;MAKE R0 EQUAL ZERO
8094
8095 060346 005000      CLR      R0            ;MAKE R0 POSITIVE ONE
8096 060350 071001      DIV      R1,R0         ;LEFT SHIFT ONE PLACE
8097 060352 040514      BIC      R5,(R4)       ;GO TO FET.05
8098
8099
8100
8101
8102 060354 012737 060364 001112      MOV      #11$,SLPERR      ;MULTIPLY R1 X R0
8103 060362 050514                BIS      R5,(R4)         ;USE MUL.60, GOTO FET.10
8104 060364 011700      11$: MOV      (PC),R0      ;MAKE SURE R0 IS ZERO
8105 060366 012700 000000      MOV      #0,R0           ;DIVISOR IS ZERO, GO TO FET.04
8106 060372 013700 100000      MOV      @#100000,R0     ;DISABLE KERNEL D-SPACE MAPPING

```

TEST THAT 'ROM OUT13' COMES UP WHEN SRCM = 1, 2, OR 3.
THIS IS ANDED WITH SRCF = 7

```

8107 060376 040514      BIC      R5,(R4)      ;DISABLE KERNEL D-SPACE MAPPING
8108
8109      ;*      TEST THAT 'ROM OUT14' COMES UP WHEN DSTM = 1, OR 2.
8110      ;*      THIS IS ANDED WITH DSTF = 7 AND NOT(MTP + MFP).
8111      ;*
8112 060400 012737 060410 001112  MOV      #12$,SLPERR  ;SET LOOP ON ERROR POINTER TO 12$
8113 060406 050514      BIS      R5,(R4)      ;ENABLE D-SPACE MAPPING IN KERNEL MODE
8114 060410 005717      12$:    TST      (PC)        ;USE D12.00, GOTO D12.10
8115      ;(INST. IS DAC * DM1)
8116 060412 005727 000000  TST      #0          ;USE D12.01, GOTO D12.10
8117      ;(INST IS DAC * DM2)
8118 060416 022717 000240  CMP      #240,(PC)   ;USE D12.80, GOTO D12.60
8119      ;(INST IS BIN * DM1)
8120 060422 000240      NOP                    ;THIS SHOULD BE COMPARED WITH IMMEDIATE
8121      ;DATA IN ABOVE COMPARE INSTRUCTION
8122 060424 040514      BIC      R5,(R4)      ;DISABLE KERNEL D-SPACE MAPPING
8123
8124      ;*      TEST THAT 'ROM OUT15' COMES UP WHEN DSTM = 3.
8125      ;*      THIS IS ANDED WITH DSTF = 7.
8126      ;*
8127 060426 012737 060436 001112  MOV      #13$,SLPERR  ;SET LOOP ON ERROR POINTER TO 13$
8128 060434 050514      BIS      R5,(R4)      ;ENABLE D-SPACE MAPPING IN KERNEL MODE
8129 060436 005037 100000      13$:    CLR      @#100000  ;CLEAR LOCATION 100000
8130      ;USE D30.00, GOTO D30.10
8131 060442 012737 000000 100000  MOV      #0,@#100000 ;LOAD ZERO TO LOCATION 100000
8132      ;USE D30.80, GOTO D30.10
8133 060450 112737 000000 100001  MOVB     #0,@#100001 ;LOAD ZERO INTO UPPER BYTE OF 100000
8134      ;USE D30.90, GOTO D30.10
8135 060456 040514      BIC      R5,(R4)      ;DISABLE KERNEL D-SPACE MAPPING
8136
8137      ;*      TEST THAT 'ROM OUT09' FORCES I-SPACE DURING INDEX WORD
8138      ;*      FETCHES.
8139      ;*
8140 060460 012737 060470 001112  MOV      #14$,SLPERR  ;SET LOOP ON ERROR POINTER TO 14$
8141 060466 050514      BIS      R5,(R4)      ;ENABLE D-SPACE MAPPING IN KERNEL MODE
8142 060470 012700 000000      14$:    MOV      #0,R0        ;BIN * DM0, USE D00.90, GOTO FET.10
8143 060474 010001      MOV      R0,R1        ;BIN * DM0 * SMO, USE EXC.80, GOTO FET.10
8144 060476 012767 000001 017274  MOV      #1,<77772-.>(PC) ;LOAD 1 INTO ADR 100000
8145      ;DSTM = 6, DSTF = 7
8146      ;USE D67.80, AND D67.00
8147 060504 112777 000000 017266  MOVB     #0,@100000  ;LOAD ZERO INTO ADR 000001
8148      ;DSTM = 7, DSTF = 7
8149      ;USE D67.90 AND D67.01
8150 060512 016700 017262      MOV      <77774-.>(PC),R0 ;READ FROM ADRS 100000
8151      ;SRCM = 6, SRCF = 7
8152      ;USE S67.00
8153 060516 040514      BIC      R5,(R4)      ;DISABLE KERNEL D-SPACE MAPPING
8154
8155      ;*      TEST 'ROM OUT11' ANDED WITH (PREV=I)
8156      ;*
8157 060520 012737 060530 001112  MOV      #15$,SLPERR  ;SET LOOP ON ERROR POINTER TO 15$
8158 060526 050514      BIS      R5,(R4)      ;ENABLE D-SPACE MAPPING IN KERNEL MODE
8159 060530 012737 000000 100000      15$:    MOV      #0,@#100000  ;CLEAR LOCATION 100000
8160 060536 005437 100000      NEG      @#100000     ;NEGATE ZERO, USE NEG.20, GOTO EXC.10
8161 060542 105437 100001      NEGB     @#100001     ;NEGATE UPPER BYTE OF 100000
8162      ;USE SHR.10, GOTO EXC.10
    
```


8163 060546 005537 100000
8164
8165 060552 040514
8166 060554 012737 060140 001112
8167
8168
8169

ADC @#100000 ;ADD CARRY BIT TO ADRS 100000
;USE EXC.00, GOTO EXC.10
19\$: BIC R5,(R4) ;DISABLE KERNEL D-SPACE MAPPING
MOV #20\$,SLPERR ;SET LOOP POINTER TO START OF TEST

:TEST 73 ENABLE KERNEL D-SPACE AND SEE THAT I-SPACE IS NOT FORCED
:THIS TEST SHOWS THAT I-SPACE IS NOT FORCED IF THE REGISTER
:FIELD IS NOT 7 BUT THE OTHER CONDITIONS ARE STILL MET.
:ALL ERRORS FOUND IN THIS TEST ARE REPORTED WHEN THE C.P.U.
:ABORTS THRU 'MMVEC' TO SUBROUTINE 'MODSPAC'. THIS SUBROUTINE
:WILL REPORT THAT D-SPACE WAS NOT ENABLED PROPERLY.

8179
8180 060562
8181 060562 000004
8182 060564 012737 061064 001316
8183
8184 060572 012737 077406 172310
8185 060600 012737 001000 172350
8186 060606 012737 077406 172320
8187 060614 012737 077406 172322
8188 060622 012737 077406 172330
8189 060630 012737 077400 172324
8190 060636 012737 077400 172326
8191 060644 012737 000000 172360
8192 060652 012737 001000 172370
8193 060660 012737 077406 172336
8194 060666 012737 177600 172376
8195 060674 012704 172516
8196 060700 012705 000004
8197 060704 012700 100000
8198 060710 012737 100000 100000
8199 060716 012737 100000 100002
8200 060724 105037 172310
8201 060730 012737 060736 001112

TST73:
SCOPE
MOV #TST74,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
20\$: MOV #77406,KIPDR4 ;MAKE KERNEL I PAGE 4 200 BLOCKS, R/W
MOV #1000,KIPAR4 ;MAP KERNEL I PAGE 4 TO 16K
MOV #77406,KDPDR0 ;MAKE KERNEL D PAGE 0 200 BLOCKS R/W
MOV #77406,KDPDR1 ;MAKE KERNEL D PAGE 1 200 BLOCKS R/W
MOV #77406,KDPDR4 ;MAKE KERNEL D PAGE 4 200 BLOCKS R/W
MOV #77400,KDPDR2 ;MAKE KERNEL D PAGE 2 NON-RESIDENT
MOV #77400,KDPDR3 ;MAKE KERNEL D PAGE 3 NON-RESIDENT
MOV #000,KDPAR0 ;MAP KERNEL D PAGE 0 TO PHYSICAL 0
MOV #1000,KDPAR4 ;MAP KERNEL D PAGE 4 TO 16K
MOV #77406,KDPDR7 ;MAP KERNEL D PAGE 7 TO 200 BLOCKS R/W
MOV #177600,KDPAR7 ;MAP KERNEL D PAGE 7 TO I/O PAGE
MOV #MRR3,R4 ;PUT ADDRS OF MRR3 IN R4
MOV #BIT2,R5 ;PUT KERNEL D-SPACE ENABLE BIT INTO R5
MOV #100000,R0 ;LOAD ADDRESS 100000 INTO R0
MOV #100000,@#100000 ;LOAD ADDRESS 100000 WITH 100000
MOV #100000,@#100002 ;LOAD ADDRESS 100002 WITH 100000
CLRB KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
MOV #11\$,SLPERR ;SET LOOP ON ERROR POINTER TO 11\$

:
:TEST THAT 'ROM OUT13' COMES UP WHEN SRCM = 1, 2, OR 3.
:IN THIS CASE SRCF = -7.

8202
8203
8204
8205
8206 060736 012700 100000
8207 060742 050514
8208 060744 000240
8209 060746 011001
8210 060750 012001
8211 060752 013001
8212 060754 040514
8213

11\$: MOV #100000,R0 ;LOAD ADDRESS 100000 INTO R0
BIS R5,(R4) ;ENABLE D-SPACE MAPPING IN KERNEL MODE
NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV (R0),R1 ;USE S13.00, SOURCE MODE IS 1
MOV (R0)+,R1 ;USE S13.01, SOURCE MODE IS 2
MOV @(R0)+,R1 ;USE S13.01, SOURCE MODE IS 3
BIC R5,(R4) ;DISABLE KERNEL D-SPACE MAPPING

:
:TEST THAT 'ROM OUT14' COMES UP WHEN DSTM = 1, OR 2.
:THIS IS ANDED WITH DSTF = -7 AND NOT(MTP + MFP).

8214
8215
8216
8217 060756 012737 060764 001112
8218 060764 012700 100000

12\$: MOV #12\$,SLPERR ;SET LOOP ON ERROR POINTER TO 12\$
MOV #100000,R0 ;LOAD ADDRESS 100000 INTO R0

```

8219 060770 050514 BIS R5,(R4) ;ENABLE D-SPACE MAPPING IN KERNEL MODE
8220 060772 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
8221 060774 005710 TST (R0) ;USE D12.00, GOTO D12.10
8222 (INST. IS DAC * DM1)
8223 060776 005720 TST (R0)+ ;USE D12.01, GOTO D12.10
8224 (INST IS DAC * DM2)
8225 061000 021010 CMP (R0),(R0) ;USE D12.80, GOTO D12.60
8226 (INST IS BIN * DM1)
8227 061002 122020 CMPB (R0)+,(R0)+ ;USE D12.90, GOTO D12.60
8228 (INST IS BIN * DM2)
8229 061004 040514 BIC R5,(R4) ;DISABLE KERNEL D-SPACE MAPPING

```

```

::
* TEST THAT 'ROM OUT15' COMES UP WHEN DSTM = 3.
* THIS IS ANDED WITH DSTF = -7.
::

```

```

8234 061006 012737 061014 001112 MOV #13$, $LPERR ;SET LOOP ON ERROR POINTER TO 13$
8235 061014 112737 000006 172310 13$: MOVB #006, KIPDR4 ;MAKE KERNEL I-SPACE PAGE 4 RESIDENT
8236 061022 012700 100000 MOV #100000, R0 ;LOAD ADDRESS 100000 INTO R0
8237 061026 105037 172310 CLRB KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
8238 061032 050514 BIS R5,(R4) ;ENABLE D-SPACE MAPPING IN KERNEL MODE
8239 061034 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
8240 061036 005030 CLR @ (R0)+ ;USE D30.00, GOTO D30.10
8241 061040 012730 100000 MOV #100000, @ (R0)+ ;USE D30.80, GOTO D30.10
8242 061044 012710 100001 MOV #100001, (R0) ;LOAD ODD ADDRESS INTO LOCATION OF R0
8243 061050 112730 000200 MOVB #200, @ (R0)+ ;USE D30.90, GOTO D30.10
8244 061054 040514 19$: BIC R5,(R4) ;DISABLE KERNEL D-SPACE MAPPING
8245 061056 012737 060572 001112 MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST

```

```

*****
*TEST 74 PROPER ENABLING OF SUPERVISOR D-SPACE
*
* THIS TEST SHOWS THE PROPER FUNCTIONING OF SUPERVISOR D-SPACE.
* BOTH 'SSRB I SPACEA L' AND 'SSRB I SPACEB L' ARE ASSERTED
* DURING THIS TEST FORCING I-SPACE.
*
* ALL ERRORS FOUND IN THIS TEST ARE REPORTED WHEN THE C.P.U.
* ABORTS THRU 'MPVEC' TO SUBROUTINE 'NODSPAC'. THIS SUBROUTINE
* WILL REPORT THAT D-SPACE WAS NOT ENABLED PROPERLY.
*****

```

```

8261 061064 TST74: SCOPE
8262 061064 000004 MOV #TST75, NXTTST ;SAVE STARTING ADDRESS OF NEXT
8263 061066 012737 061536 001316 20$: MOV #77406, R0 ;TEST FOR ESCAPE ON PARITY ERRORS
8264 ;MAKE THE NEXT FEW PAGES 4K, R/W, UP
8265 061074 012700 077406 MOV R0, SIPDR0 ;SUPERVISOR I-SPACE PAGE 0
8266 061100 010037 172200 MOV R0, SIPDR1 ;SUPERVISOR I-SPACE PAGE 1
8267 061104 010037 172202 MOV R0, SIPDR2 ;SUPERVISOR I-SPACE PAGE 2
8268 061110 010037 172204 MOV R0, SIPDR3 ;SUPERVISOR I-SPACE PAGE 3
8269 061114 010037 172206 MOV R0, SIPDR4 ;SUPERVISOR I-SPACE PAGE 4
8270 061120 010037 172210 MOV R0, SIPDR7 ;SUPERVISOR I-SPACE PAGE 7
8271 061124 010037 172216 MOV R0, SDPDR0 ;SUPERVISOR D-SPACE PAGE 0
8272 061130 010037 172220 MOV R0, SDPDR1 ;SUPERVISOR D-SPACE PAGE 1
8273 061134 010037 172222 MOV R0, SDPDR4 ;SUPERVISOR D-SPACE PAGE 4
8274 061140 010037 172230

```

```

8275 061144 010037 172236      MOV     R0,SDPDR7      ;SUPERVISOR D-SPACE PAGE 7
8276 061150 105000              CLRFB   R0             ;MAKE THE NEXT TWO PAGES NON-RESIDENT
8277 061152 010037 172224      MOV     R0,SDPDR2      ;SUPERVISOR D-SPACE PAGE 2
8278 061156 010037 172226      MOV     R0,SDPDR3      ;SUPERVISOR D-SPACE PAGE 3
8279 061162 012737 000000 172240  MOV     #000,SIPAR0     ;MAP SUPER I-SPACE PAGE 0 TO PHY 0
8280 061170 012737 000200 172242  MOV     #200,SIPAR1     ;MAP SUPER I-SPACE PAGE 1 TO 4K - 8K
8281 061176 012737 000400 172244  MOV     #400,SIPAR2     ;MAP SUPER I-SPACE PAGE 2 TO 8K - 12K
8282 061204 012737 000600 172246  MOV     #600,SIPAR3     ;MAP SUPER I-SPACE PAGE 3 TO 12K - 16K
8283 061212 012737 001000 172250  MOV     #1000,SIPAR4    ;MAP SUPER I-SPACE PAGE 4 TO 16K - 20K
8284 061220 012737 177600 172256  MOV     #177600,SIPAR7  ;MAP SUPER I-SPACE PAGE 7 TO I/O PAGE
8285 061226 012737 000000 172260  MOV     #000,SDPAR0     ;MAP SUPER D-SPACE PAGE 0 TO PHY 0
8286 061234 012737 000200 172262  MOV     #200,SDPAR1     ;MAP SUPER D-SPACE PAGE 1 TO 4K - 8K
8287 061242 012737 177600 172276  MOV     #177600,SDPAR7  ;MAP SUPER D-SPACE PAGE 7 TO I/O PAGE
8288 061250 012704 172516              MOV     #MPR3,R4       ;PUT ADDRESS OF MPR3 IN R4
8289 061254 012705 000002              MOV     #BIT1,R5       ;PUT ENABLE SUPERVISOR D-SPACE BIT IN R5
8290 061260 052737 040000 177776  BIS     #40000,PSW      ;GO INTO SUPERVISOR MODE HERE
8291 061266 105037 172210              CLRFB   SIPDR4        ;MAKE I-SPACE PAGE 4 NON-RESIDENT
8292 061272 012737 061300 001112  MOV     #10$, $LPERR   ;SET LOOP ON ERROR POINTER TO 10$
8293
8294      ;:
8295      ;: THIS SECTION CHECKS SIGNAL SPACEA FOR DISABLING D-SPACE
8296      ;:
8296 061300 050514 10$: BIS     R5,(R4)        ;ENABLE SUPER D-SPACE
8297 061302 000240      NOP
8298 061304 000400      BR     1$             ;THIS INST FETCH SHOULD USE ROM OUT09
8299 061306 040514 1$: BIC     R5,(R4)        ;DISABLE SUPER D-SPACE
8300 061310 012737 061316 001112  MOV     #11$, $LPERR   ;SET LOOP ON ERROR POINTER TO 11$
8301 061316 050514 11$: BIS     R5,(R4)        ;ENABLE SUPER D-SPACE
8302 061320 000240      NOP
8303 061322 017777 016452 016450  MOV     @100000,@100000 ;THIS INST USES ROM OUT09
8304 061330 040514      BIC     R5,(R4)        ;DISABLE SUPER D-SPACE
8305 061332 012737 061340 001112  MOV     #12$, $LPERR   ;SET LOOP ON ERROR POINTER TO 12$
8306 061340 050514 12$: BIS     R5,(R4)        ;ENABLE SUPER D-SPACE
8307 061342 000240      NOP
8308 061344 005037 100000      CLR     @#100000      ;THIS INST USES ROM OUT15 & DSTF7
8309 061350 040514      BIC     R5,(R4)        ;DISABLE SUPER D-SPACE
8310
8311      ;:
8312      ;: THIS SECTION CHECKS SIGNAL SPACEB FOR DISABLING D-SPACE
8313      ;:
8313 061352 012737 061360 001112  MOV     #13$, $LPERR   ;SET LOOP ON ERROR POINTER TO 13$
8314 061360 050514 13$: BIS     R5,(R4)        ;ENABLE SUPER D-SPACE
8315 061362 000240      NOP
8316 061364 013700 100000      MOV     @#100000,R0    ;THIS INST USES ROM OUT13 & SRCF7
8317 061370 040514      BIC     R5,(R4)        ;DISABLE SUPER D-SPACE
8318 061372 012737 061400 001112  MOV     #14$, $LPERR   ;SET LOOP ON ERROR POINTER TO 14$
8319 061400 050514 14$: BIS     R5,(R4)        ;ENABLE SUPER D-SPACE
8320 061402 000240      NOP
8321 061404 005727 000000      TST     #0             ;THIS INST USES ROM OUT14 & DSTF7
8322 061410 040514      BIC     R5,(R4)        ;DISABLE SUPER D-SPACE
8323
8324      ;:
8325      ;: THE FOLLOWING SECTION VERIFIES THAT I-SPACE IS NOT
8326      ;: FORCED DURING THE ADDRESSING CYCLES OF THE INSTRUCTIONS.
8327      ;:
8327 061412 012737 061420 001112  MOV     #16$, $LPERR   ;SET LOOP ON ERROR POINTER TO 16$
8328 061420 112737 000006 172210  MOVFB  #006,SIPDR4     ;MAKE I-SPACE PAGE 4 RESIDENT
8329 061426 012700 100000      MOV     #100000,R0     ;LOAD ADDRESS 100000 INTO R0
8330 061432 012737 100000 100000  MOV     #100000,@#100000 ;LOAD NO. 100000 INTO ADDRESS 100000
    
```

8331	061440	105037	172210	CLRB	SIPDR4	:MAKE I-SPACE PAGE 4 NON-RESIDENT
8332	061444	050514		BIS	R5,(R4)	:ENABLE SUPER D-SPACE
8333	061446	000240		NOP		:THIS IS A SYNC POINT FOR SCOPING
8334	061450	012730	100000	MOV	#100000,a(R0)+	:THIS INST USES ROM OUT15 & -DSTF7
8335	061454	040514		BIC	R5,(R4)	:DISABLE SUPER D-SPACE
8336	061456	012737	061464	MOV	#17\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 17\$
8337	061466	012700	100000	MOV	#100000,R0	:LOAD ADDRESS 100000 INTO R0
8338	061470	050514		BIS	R5,(R4)	:ENABLE SUPER D-SPACE
8339	061472	000240		NOP		:THIS IS A SYNC POINT FOR SCOPING
8340	061474	011001		MOV	(R0),R1	:THIS INST USES ROM OUT13 & -SRCF7
8341	061476	040514		BIC	R5,(R4)	:DISABLE SUPER D-SPACE
8342	061500	012737	061506	MOV	#18\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 18\$
8343	061506	012700	100000	MOV	#100000,R0	:LOAD ADDRESS 100000 INTO R0
8344	061512	050514		BIS	R5,(R4)	:ENABLE SUPER D-SPACE
8345	061514	000240		NOP		:THIS IS A SYNC POINT FOR SCOPING
8346	061516	005720		TST	(R0)+	:THIS INST USES ROM OUT14 & -DSTF7
8347	061520	040514		BIC	R5,(R4)	:DISABLE SUPER D-SPACE
8348	061522	042737	040000	BIC	#40000,PSW	:GO BACK TO KERNEL MODE
8349	061530	012737	061074	MOV	#20\$, \$LPERR	:SET LOOP POINTER TO START OF TEST

8350

8351

8352

8353

8354

8355

8356

8357

8358

8359

8360

8361

8362

8363

8364

8365

8366

8367

8368

8369

8370

8371

8372

8373

8374

8375

8376

8377

8378

8379

8380

8381

8382

8383

8384

8385

8386

*TEST 75 PROPER ENABLING OF USER D-SPACE

*

* THIS TEST SHOWS THE PROPER FUNCTIONING OF USER D-SPACE.
* BOTH 'SSRB I SPACEA L' AND 'SSRB I SPACEB L' ARE ASSERTED
* DURING THIS TEST FORCING I-SPACE.

*

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

*

TST75:

8364	061536			SCOPE		
8365	061536	000004		MOV	#TST76,NXTTST	:SAVE STARTING ADDRESS OF NEXT
8366	061540	012737	062210	MOV		:TEST FOR ESCAPE ON PARITY ERRORS
8367						:MAKE THE NEXT FEW PAGES 4K, R/W, UP
8368	061546	012700	077406	MOV	#77406,R0	:USER I-SPACE PAGE 0
8369	061552	010037	177600	MOV	R0,UIPDR0	:USER I-SPACE PAGE 1
8370	061556	010037	177602	MOV	R0,UIPDR1	:USER I-SPACE PAGE 2
8371	061562	010037	177604	MOV	R0,UIPDR2	:USER I-SPACE PAGE 3
8372	061566	010037	177606	MOV	R0,UIPDR3	:USER I-SPACE PAGE 4
8373	061572	010037	177610	MOV	R0,UIPDR4	:USER I-SPACE PAGE 7
8374	061576	010037	177616	MOV	R0,UIPDR7	:USER D-SPACE PAGE 0
8375	061602	010037	177620	MOV	R0,UDPDR0	:USER D-SPACE PAGE 1
8376	061606	010037	177622	MOV	R0,UDPDR1	:USER D-SPACE PAGE 4
8377	061612	010037	177630	MOV	R0,UDPDR4	:USER D-SPACE PAGE 7
8378	061616	010037	177636	MOV	R0,UDPDR7	:MAKE THE NEXT TWO PAGES NON-RESIDENT
8379	061622	105000		CLRB	R0	:USER D-SPACE PAGE 2
8380	061624	010037	177624	MOV	R0,UDPDR2	:USER D-SPACE PAGE 3
8381	061630	010037	177626	MOV	R0,UDPDR3	:MAP USER I-SPACE PAGE 0 TO PHY 0
8382	061634	012737	000000	MOV	#000,UIPAR0	:MAP USER I-SPACE PAGE 1 TO 4K - 8K
8383	061642	012737	000200	MOV	#200,UIPAR1	:MAP USER I-SPACE PAGE 2 TO 8K - 12K
8384	061650	012737	000400	MOV	#400,UIPAR2	:MAP USER I-SPACE PAGE 3 TO 12K - 16K
8385	061656	012737	000600	MOV	#600,UIPAR3	:MAP USER I-SPACE PAGE 4 TO 16K - 20K
8386	061664	012737	001000	MOV	#1000,UIPAR4	

8387	061672	012737	177600	177656	MOV	#177600,U1PAR7	:MAP USER I-SPACE PAGE 7 TO I/O PAGE
8388	061700	012737	000000	177660	MOV	#000,U1PAR0	:MAP USER D-SPACE PAGE 0 TO PHY 0
8389	061706	012737	000200	177662	MOV	#200,U1PAR1	:MAP USER D-SPACE PAGE 1 TO 4K - 8K
8390	061714	012737	177600	177676	MOV	#177600,U1PAR7	:MAP USER D-SPACE PAGE 7 TO I/O PAGE
8391	061722	012704	172516		MOV	#MMR3,R4	:PUT ADDRESS OF MMR3 IN R4
8392	061726	012705	000001		MOV	#BIT0,R5	:PUT ENABLE USER D-SPACE BIT IN R5
8393	061732	052737	140000	177776	BIS	#140000,PSW	:GO INTO USER MODE HERE
8394	061740	105037	177610		CLRB	U1PDR4	:MAKE I-SPACE PAGE 4 NON-RESIDENT
8395	061744	012737	061752	001112	MOV	#10\$,SLPERR	:SET LOOP ON ERROR POINTER TO 10\$
8396	061752	050514		10\$:	BIS	R5,(R4)	:ENABLE USER D-SPACE
8397	061754	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
8398	061756	000400			BR	1\$:THIS INST FETCH SHOULD USE ROM OUT09
8399	061760	040514		1\$:	BIC	R5,(R4)	:DISABLE USER D-SPACE
8400	061762	012737	061770	001112	MOV	#11\$,SLPERR	:SET LOOP ON ERROR POINTER TO 11\$
8401	061770	050514		11\$:	BIS	R5,(R4)	:ENABLE USER D-SPACE
8402	061772	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
8403	061774	017777	016000	015776	MOV	@100000,@100000	:THIS INST USES ROM OUT09
8404	062002	040514			BIC	R5,(R4)	:DISABLE USER D-SPACE
8405	062004	012737	062012	001112	MOV	#12\$,SLPERR	:SET LOOP ON ERROR POINTER TO 12\$
8406	062012	050514		12\$:	BIS	R5,(R4)	:ENABLE USER D-SPACE
8407	062014	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
8408	062016	005037	100000		CLR	@#100000	:THIS INST USES ROM OUT15 & DSTF7
8409	062022	040514			BIC	R5,(R4)	:DISABLE USER D-SPACE
8410					::		
8411					::		
8412					::		
8413	062024	012737	062032	001112	MOV	#13\$,SLPERR	:SET LOOP ON ERROR POINTER TO 13\$
8414	062032	050514		13\$:	BIS	R5,(R4)	:ENABLE USER D-SPACE
8415	062034	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
8416	062036	013700	100000		MOV	@#100000,R0	:THIS INST USES ROM OUT13 & SRCF7
8417	062042	040514			BIC	R5,(R4)	:DISABLE USER D-SPACE
8418	062044	012737	062052	001112	MOV	#14\$,SLPERR	:SET LOOP ON ERROR POINTER TO 14\$
8419	062052	050514		14\$:	BIS	R5,(R4)	:ENABLE USER D-SPACE
8420	062054	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
8421	062056	005727	000000		TST	#0	:THIS INST USES ROM OUT14 & DSTF7
8422	062062	040514			BIC	R5,(R4)	:DISABLE USER D-SPACE
8423					::		
8424					::		
8425					::		
8426					::		
8427	062064	012737	062072	001112	MOV	#16\$,SLPERR	:SET LOOP ON ERROR POINTER TO 16\$
8428	062072	112737	000006	177610	MOVB	#006,U1PDR4	:MAKE I-SPACE PAGE 4 RESIDENT
8429	062100	012700	100000		MOV	#100000,R0	:LOAD ADDRESS 100000 INTO R0
8430	062104	012737	100000	100000	MOV	#100000,@#100000	:LOAD NO. 100000 INTO ADDRESS 100000
8431	062112	105037	177610		CLRB	U1PDR4	:MAKE I-SPACE PAGE 4 NON-RESIDENT
8432	062116	050514			BIS	R5,(R4)	:ENABLE USER D-SPACE
8433	062120	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
8434	062122	012730	100000		MOV	#100000,@(R0)+	:THIS INST USES ROM OUT15 & -DSTF7
8435	062126	040514			BIC	R5,(R4)	:DISABLE USER D-SPACE
8436	062130	012737	062136	001112	MOV	#17\$,SLPERR	:SET LOOP ON ERROR POINTER TO 17\$
8437	062136	012700	100000	17\$:	MOV	#100000,R0	:LOAD ADDRESS 100000 INTO R0
8438	062142	050514			BIS	R5,(R4)	:ENABLE USER D-SPACE
8439	062144	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
8440	062146	011001			MOV	(R0),R1	:THIS INST USES ROM OUT13 & -SRCF7
8441	062150	040514			BIC	R5,(R4)	:DISABLE USER D-SPACE
8442	062152	012737	062160	001112	MOV	#18\$,SLPERR	:SET LOOP ON ERROR POINTER TO 18\$

```

8443 062160 012700 100000 18$: MOV #100000,R0 ;LOAD ADDRESS 100000 INTO R0
8444 062164 050514 BIS R5,(R4) ;ENABLE USER D-SPACE
8445 062166 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
8446 062170 005720 TST (R0)+ ;THIS INST USES ROM OUT14 & -DSTF7
8447 062172 040514 BIC R5,(R4) ;DISABLE USER D-SPACE
8448 062174 042737 140000 177776 BIC #140000,PSW ;GO BACK TO KERNEL MODF
8449 062202 012737 061546 001112 MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
8450
8451
8452
8453 :*****
8454 :*TEST 76 TRAPPING IN D-SPACE KERNEL MODE
8455 :*
8456 :* THIS TEST VERIFIES THAT THE ABORT VECTOR IS TAKEN FROM D-SPACE
8457 :* AND NOT I-SPACE. THE I-SPACE VECTOR POINTS TO 10$, AND
8458 :* THE D-SPACE VECTOR POINTS TO 15$. EACH PSW IN VIRTUAL 252 IS
8459 :* DIFFERENT SO THE PROGRAM CAN TELL WHICH AREA IT IS PICKED
8460 :* UP FROM.
8461 :*****
8461 062210 TST76:
8462 062210 000004 SCOPE
8463 062212 012737 062456 001316 MOV #TST77,NXTTST ;SAVE STARTING ADDRESS OF NEXT
8464 ;TEST FOR ESCAPE ON PARITY ERRORS
8465 062220 104416 TBITO ;MAKE SURE T-BIT IS OFF FOR THIS TEST
8466 062222 012737 062260 001112 20$: MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
8467 062230 012737 062332 000250 MOV #10$,M#250 ;SET M.M.VEC TO HOLD BAD VECTOR
8468 062236 012737 000000 000252 MOV #000,M#252 ;PROC. STAT. IN 252 HAS PRIORITY OF ZERO
8469 062244 012737 062340 000350 MOV #15$,M#350 ;D-SPACE M.M.VECTOR IS AT 35C
8470 062252 012737 000340 000352 MOV #340,M#352 ;PRIORITY FOR D-SPACE VECTOR IS 7
8471 062260 012706 001000 1$: MOV #1000,KSP ;SET UP KERNEL VECTOR
8472 062264 042737 177776 177572 BIC #177776,MMRO ;CLEAR ALL ERROR BITS IN MMRO
8473 ; NOW SET UP FOR AN ABORT IN KERNEL MODE D-SPACE ENABLED
8474 ;
8475 062272 012737 077402 172330 MOV #77402,KDPDR4 ;KERNEL D-SPACE PAGE 4 IS READ ONLY
8476 062300 012737 001000 172370 MOV #1000,KDPAR4 ;MAP D-SPACE PAGE 4 TO 16K
8477 062306 052737 000004 172516 BIS #BIT2,MMR3 ;ENABLE KERNEL D-SPACE MAPPING
8478 062314 012737 000001 172360 MOV #1,KDPAR0 ;MAP KERNEL D PAGE 0 TO 000100
8479 062322 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
8480 062324 012737 177777 100000 MOV #-1,M#100000 ;TRY TO WRITE TO PAGE 4
8481 062332 013700 177776 10$: MOV PSW,R0 ;SAVE PROC. STAT. FOR COMPARE
8482 062336 000402 BR 16$ ;BRANCH TO D-SPACE READ CODE
8483 062340 013700 177776 15$: MOV PSW,R0 ;SAVE PROC. STAT FOR COMPARE
8484 062344 005037 172360 16$: CLR KDPAR0 ;RE-MAP KERNEL D PAGE 0 TO PHYSICAL 0
8485 062350 012706 001100 MOV #KERSTK,KSP ;RE-SET STACK POINTER AFTER D-SPACE
8486 ;ABORT.
8487 062354 013737 177572 001250 MOV MMRO,PMRO ;SAVE MMRO FOR COMPARE
8488 062362 013737 177574 001252 MOV MMR1,PMR1 ;SAVE MMR1 FOR COMPARE
8489 062370 013737 177576 001254 MOV MMR2,PMR2 ;SAVE MMR2 FOR COMPARE
8490 062376 122700 000340 CMPB #340,R0 ;DID YOU PICK UP THE CORRECT PSW
8491 062402 001401 BEQ 2$ ;BRANCH IF PSW IS 340
8492 062404 104130 ERROR 130 ;WRONG PSW PICKED UP IN ABORT SEQUENCE.
8493 062406 022737 020031 001250 2$: CMP #020031,PMRO ;EXPECTING READ ONLY ABORT
8494 ;KERNEL MODE, D-SPACE, PAGE 4
8495 062414 001401 BEQ 3$ ;BRANCH IF CONDITION IS CORRECT
8496 062416 104131 ERROR 131 ;WRONG M.M. ABORT CONDITION
8497 062420 042737 177776 177572 3$: BIC #177776,MMRO ;CLEAR MMRO FOR EXIT OF TEST
8498 062426 042737 000004 172516 BIC #BIT2,MMR3 ;TURN OFF KERNEL D-SPACE ENABLE
    
```

```

8499 062434 012737 032226 000250      MOV      #MMTRAP,MMVEC      ;RESTORE NORMAL M.M. TRAP VECTOR
8500 062442 012737 000340 000252      MOV      #340,MMVEC+2      ;PRIORITY EQUALS 7
8501 062450 012737 062222 001112      MOV      #20$,SLPERR      ;SET LOOP POINTER TO START OF TEST
8502
8503
8504      .SBTTL ***** ENTRY POINT 7 --- STARTING ADDRESS 230 *****
8505      .SBTTL ***** A & W BIT LOGIC TEST AND DUAL MAPPING TESTS *****
8506
8507      *
8508      * THIS GROUP OF TESTS CHECKS OUT THE A-BIT AND W-BIT LOGIC ON
8509      * 'SAPD' AND THEN USES THAT LOGIC TO VERIFY THAT THERE IS NO
8510      * DUAL MAPPING OF PAR/PDR PAIRS BETWEEN GROUPS OR INSIDE A GROUP.
8511      * ALL A-BITS AND W-BITS ARE SET TO ENSURE THAT THERE ISN'T A BAD
8512      * CHIP IN THE PDR'S.
8513      *
8514
8515      :*****
8516      :*TEST 77      TEST A-BIT AND W-BIT LOGIC
8517      :*
8518      :* THIS TEST CHECKS ALL THE LOGIC FOR THE A-BIT AND W-BIT ON
8519      :* 'SAPD'. THE BITS ARE SET ONE AT A TIME THEN A REFERENCE
8520      :* TO THAT PAGE CAUSING AN ABORT IS MADE TO SEE IF THE BIT REMAINS
8521      :* SET. LAST EITHER THE PAR OR PDR IS WRITTEN TO SEE THAT THE
8522      :* BITS ARE CLEARED DURING A PAR OR PDR LOAD.
8523      :*
8524      :*****
8525      :TST77:
8526      :SCOPE
8527      :MOV      #TST100,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
8528      :TBITR      ;TEST FOR ESCAPE ON PARITY ERRORS
8529      :104420      ;RESTORE THE T-BIT TO ITS CONDITION
8530      :BEFORE THE LAST TEST.
8531      :ENTPT7:
8532      :MOV      #20$,SLPADR      ;SET LOOP ADDRESS POINTER TO 20$
8533      :MOV      #20$,SLPERR      ;SET LOOP ON ERROR POINTER TO 20$
8534      :MOV      #77,$TSTNM      ;LOAD TEST NUMBER INTO MEMORY
8535      :MOV      $TSTNM,DISPLAY      ;DISPLAY TEST NUMBER FOR THIS TEST
8536      :MOV      #77406,KIPDR0      ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
8537      :MOV      #77406,KIPDR1      ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
8538      :MOV      #77406,KIPDR2      ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
8539      :MOV      #77406,KIPDR3      ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
8540      :MOV      #77406,KIPDR7      ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
8541      :MOV      #000,KIPAR0      ;MAP KERNEL I PAGE 0 TO 0 - 4K
8542      :MOV      #200,KIPAR1      ;MAP KERNEL I PAGE 1 TO 4K - 8K
8543      :MOV      #400,KIPAR2      ;MAP KERNEL I PAGE 2 TO 8K - 12K
8544      :MOV      #600,KIPAR3      ;MAP KERNEL I PAGE 3 TO 12K - 16K
8545      :MOV      #177600,KIPAR7      ;MAP KERNEL I PAGE 7 TO THE I/O PAGE
8546      :MOV      #BIT0,MMR0      ;ENABLE 18-BIT RELOCATION IF NOT ON
8547      :MOV      #BIT4,MMR3      ;ENABLE 22-BIT RELOCATION IF NOT ON
8548      :EQUIV      BIT6,WBIT      ;LABEL 'WBIT' IS EQUIVALENT TO BIT 6
8549      :EQUIV      BIT7,ABIT      ;LABEL 'ABIT' IS EQUIVALENT TO BIT 7
8550      :20$:      MOV      #00006,KIPDR4      ;PAGE 4 HAS A PAGE LENGTH OF 1
8551      :MOV      #1000,KIPAR4      ;MAP PAGE 4 TO 16K
8552      :MOV      #10$,MMVEC      ;SET M.M. TRAP VECTOR TO 10$
8553      :MOV      #11$,SLPERR      ;SET LOOP ON ERROR POINTER TO 11$
8554      :11$:      NOP      ;THIS IS A SYNC POINT FOR SCOPING
    
```

8555	062662	012737	012345	100000	18:	MOV	#12345, @#100000	:WRITE INTO PAGE 4
8556	062670	013737	172310	001172		MOV	KIPDR4, \$TMP0	:READ PDR 4 TO CHECK W-BIT
8557	062676	032737	000100	001172		BIT	#WBIT, \$TMP0	:SEE IF W-BIT IS SET
8558	062704	001001				BNE	2\$:BRANCH IF W-BIT IS SET
8559	062706	104102				ERROR	102	:W-BIT NOT SET
8560	062710	012737	062716	001112	2\$:	MOV	#12\$, \$SLPERR	:SET LOOP ON ERROR POINTER TO 12\$
8561	062716	000240			12\$:	NOP		:THIS IS A SYNC POINT FOR SCOPING
8562	062720	012737	012345	100100		MOV	#12345, @#100100	:WRITE INTO PAGE 4, BLOCK 2
8563	062726	013737	172310	001172		MOV	KIPDR4, \$TMP0	:READ PDR 4 TO SEE IF W-BIT REMAINED SET
8564	062734	032737	000100	001172		BIT	#WBIT, \$TMP0	:SEE IF W-BIT REMAINED SET
8565	062742	001001				BNE	3\$:BRANCH IF W-BIT IS SET
8566	062744	104103				ERROR	103	:W-BIT DID NOT REMAIN SET
8567	062746	012737	062754	001112	3\$:	MOV	#13\$, \$SLPERR	:SET LOOP ON ERROR POINTER TO 13\$
8568	062754	000240			13\$:	NOP		:THIS IS A SYNC POINT FOR SCOPING
8569	062756	112737	000004	172310		MOVB	#004, KIPDR4	:CHANGE PAGE 4'S ACCESS CONTROL FIELD
8570	062764	013737	172310	001172		MOV	KIPDR4, \$TMP0	:READ PDR 4 TO CHECK W-BIT CLEARING
8571	062772	022737	000004	001172		CMPI	#00004, \$TMP0	:SEE IF W-BIT GOT CLEARED ON PDR WRITE
8572	063000	001401				BEQ	4\$:BRANCH IF W-BIT GOT CLEARED
8573	063002	104104				ERROR	104	:W-BIT DIDN'T GET CLEARED ON PDR WRITE
8574	063004	012737	063012	001112	4\$:	MOV	#14\$, \$SLPERR	:SET LOOP ON ERROR POINTER TO 14\$
8575	063012	052737	001000	177572	14\$:	BIS	#BIT9, \$MPRO	:ENABLE M.M. TRAPS
8576	063020	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
8577	063022	013700	100000			MOV	@#100000, R0	:READ FROM PAGE 4, BLOCK 1
8578	063026	013737	172310	001172		MOV	KIPDR4, \$TMP0	:READ PDR 4 TO CHECK A-BIT
8579	063034	032737	000200	001172		BIT	#ABIT, \$TMP0	:SEE IF A-BIT WAS SET DURING READ
8580	063042	001001				BNE	5\$:BRANCH IF A-BIT WAS SET
8581	063044	104105				ERROR	105	:A-BIT DIDN'T GET SET
8582	063046	012737	063054	001112	5\$:	MOV	#15\$, \$SLPERR	:SET LOOP ON ERROR POINTER TO 15\$
8583	063054	052737	001000	177572	15\$:	BIS	#BIT9, \$MPRO	:ENABLE M.M. TRAPS
8584	063062	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
8585	063064	013700	100100			MOV	@#100100, R0	:READ FROM PAGE 4, BLOCK 2
8586	063070	013737	172310	001172		MOV	KIPDR4, \$TMP0	:READ PDR 4 TO SEE IF A-BIT REMAINED SET
8587	063076	032737	000200	001172		BIT	#ABIT, \$TMP0	:SEE IF A-BIT IS STILL SET
8588	063104	001001				BNE	6\$:BRANCH IF A-BIT IS STILL SET
8589	063106	104106				ERROR	106	:A-BIT DIDN'T REMAIN SET
8590	063110	012737	063116	001112	6\$:	MOV	#16\$, \$SLPERR	:SET LOOP ON ERROR POINTER TO 16\$
8591	063116	000240			16\$:	NOP		:THIS IS A SYNC POINT FOR SCOPING
8592	063120	012737	001000	172350		MOV	#1000, KIPAR4	:WRITE PAR 4 TO SEE IF A-BIT CLEARS
8593	063126	013737	172310	001172		MOV	KIPDR4, \$TMP0	:READ PDR 4 TO CHECK A-BIT
8594	063134	022737	000004	001172		CMPI	#00004, \$TMP0	:SEE IF A-BIT IS CLEAR
8595	063142	001401				BEQ	7\$:BRANCH IF A-BIT IS CLEAR
8596	063144	104107				ERROR	107	:A-BIT DIDN'T CLEAR ON PAR WRITE
8597	063146	012737	063154	001112	7\$:	MOV	#17\$, \$SLPERR	:SET LOOP ON ERROR POINTER TO 17\$
8598	063154	000240			17\$:	NOP		:THIS IS A SYNC POINT FOR SCOPING
8599	063156	012737	123456	170200		MOV	#123456, \$MAPLO	:LOAD A MAP REGISTER TO CHECK 'INT REG'
8600	063164	013737	172316	001172		MOV	KIPDR7, \$TMP0	:READ PDR 7 TO CHECK W-BIT
8601	063172	032737	000100	001172		BIT	#WBIT, \$TMP0	:SEE IF W-BIT SET ON MAP REGISTER WRITE
8602	063200	001001				BNE	8\$:BRANCH IF W-BIT WAS SET
8603	063202	104110				ERROR	110	:W-BIT DID NOT GET SET
8604	063204	012737	063212	001112	8\$:	MOV	#18\$, \$SLPERR	:SET LOOP ON ERROR POINTER TO 18\$
8605	063212	000240			18\$:	NOP		:THIS IS A SYNC POINT FOR SCOPING
8606	063214	012737	077406	172310		MOV	#77406, KIPDR4	:LOAD AN INTERNAL REGISTER, SHOULDN'T
8607								:AFFECT PAGE 7'S W-BIT
8608	063222	013737	172316	001172		MOV	KIPDR7, \$TMP0	:READ PDR 7 TO CHECK W-BIT AGAIN
8609	063230	032737	000100	001172		BIT	#WBIT, \$TMP0	:SEE IF W-BIT IS STILL SET
8610	063236	001001				BNE	9\$:BRANCH IF W-BIT IS STILL SET

8611	063240	104111				ERROR	111		:W-BIT NOT SET AFTER WRITING PDR 4
8612	063242	012737	062630	001112	9\$:	MOV	#20\$,SLPERR		:SET LOOP POINTER TO START OF TEST
8613	063250	012737	032226	000250		MOV	MMTRAP,MMVEC		:RESTORE NORMAL M.M. TRAP ROUTINE
8614	063256	000404				BR	TST100		:BRANCH TO NEXT TEST
8615									
8616	063260	042737	177776	177572	10\$:	BIC	#177776,MMRO		:CLEAR MMRO FOR NEXT READ OR WRITE
8617	063266	000002				RTI			:RETURN TO TEST AND CHECK A OR W BIT

```

: *
: * THESE NEXT SIX (6) TESTS USE A.C.F. = 5 (TRAP ON WRITE) TO
: * TEST ALL A & W BITS. THE TESTS ALSO CHECK FOR DUAL MAPPING
: * AMONG GROUPS OR INSIDE A GROUP OF PAR/PDR'S. THIS IS DONE BY
: * WRITING ONLY ONE PAGE IN A MODE OF OPERATION THEN CHECKING ALL
: * PDR'S TO SEE THAT ONLY THE ONE UNDER TEST HAS BOTH ITS A & W
: * BITS SET. THIS TEST IS RUN IN ALL MODES, WITH D-SPACE DISABLED
: * AND THEN AGAIN WITH D-SPACE ENABLED, SO THAT ALL THE REGISTER
: * PAIRS ARE TESTED.

```

```

: *
: * *****
: * TEST 100 DUAL MAPPING KERNEL MODE I-SPACE
: *

```

```

: * THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405
: * (4K PAGE, TRAP ON WRITE). THEN THE VIRTUAL ADDRESS IS SET TO
: * 010200 AND THAT WORD IS WRITTEN INTO ITSELF. THIS WRITE WILL
: * SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP
: * SINCE BIT09 OF MMRO IS CLEAR) AND SET BOTH THE A & W BITS IN
: * THE KERNEL I-SPACE P.D.R. UNDER TEST. NOW ALL OF THE
: * P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE
: * ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
: * WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
: * ADDRESS OF 'MAPLOO' (17770200) WHICH SHOULD ALWAYS EXIST.

```

```

: *
: * *****
: * TST100:

```

8645	063270					SCOPE			
8646	063270	000004				MOV	#TST101,NXTTST		:SAVE STARTING ADDRESS OF NEXT
8647	063272	012737	064514	001316					:TEST FOR ESCAPE ON PARITY ERRORS
8648									
8649									
8650									
8651									
8652									
8653	063300	012737	000000	172340		MOV	#0,KIPAR0		:
8654	063306	012737	000200	172342		MOV	#200,KIPAR1		:
8655	063314	012737	000400	172344		MOV	#400,KIPAR2		:
8656	063322	012737	000600	172346		MOV	#600,KIPAR3		:
8657	063330	012737	001000	172350		MOV	#1000,KIPAR4		:
8658	063336	012737	001000	172352		MOV	#1000,KIPAR5		:
8659	063344	012737	001000	172354		MOV	#1000,KIPAR6		:
8660	063352	012737	177600	172356		MOV	#177600,KIPAR7		:
8661	063360	012737	000000	172360		MOV	#0,KDPAR0		:
8662	063366	012737	000200	172362		MOV	#200,KDPAR1		:
8663	063374	012737	000400	172364		MOV	#400,KDPAR2		:
8664	063402	012737	000600	172366		MOV	#600,KDPAR3		:
8665	063410	012737	001000	172370		MOV	#1000,KDPAR4		:
8666	063416	012737	001000	172372		MOV	#1000,KDPAR5		:

8667	063424	012737	001000	172374		MOV	#1000,KDPA6	:
8668	063432	012737	177600	172376		MOV	#177600,KDPA7	:
8669	063440	012737	000000	172240		MOV	#0,SIPAR0	:
8670	063446	012737	000200	172242		MOV	#200,SIPAR1	:
8671	063454	012737	000400	172244		MOV	#400,SIPAR2	:
8672	063462	012737	000600	172246		MOV	#600,SIPAR3	:
8673	063470	012737	001000	172250		MOV	#1000,SIPAR4	:
8674	063476	012737	001000	172252		MOV	#1000,SIPAR5	:
8675	063504	012737	001000	172254		MOV	#1000,SIPAR6	:
8676	063512	012737	177600	172256		MOV	#177600,SIPAR7	:
8677	063520	012737	000000	172260		MOV	#0,SDPAR0	:
8678	063526	012737	000200	172262		MOV	#200,SDPAR1	:
8679	063534	012737	000400	172264		MOV	#400,SDPAR2	:
8680	063542	012737	000600	172266		MOV	#600,SDPAR3	:
8681	063550	012737	001000	172270		MOV	#1000,SDPAR4	:
8682	063556	012737	001000	172272		MOV	#1000,SDPAR5	:
8683	063564	012737	001000	172274		MOV	#1000,SDPAR6	:
8684	063572	012737	177600	172276		MOV	#177600,SDPAR7	:
8685	063600	012737	000000	177640		MOV	#0,UIPAR0	:
8686	063606	012737	000200	177642		MOV	#200,UIPAR1	:
8687	063614	012737	000400	177644		MOV	#400,UIPAR2	:
8688	063622	012737	000600	177646		MOV	#600,UIPAR3	:
8689	063630	012737	001000	177650		MOV	#1000,UIPAR4	:
8690	063636	012737	001000	177652		MOV	#1000,UIPAR5	:
8691	063644	012737	001000	177654		MOV	#1000,UIPAR6	:
8692	063652	012737	177600	177656		MOV	#177600,UIPAR7	:
8693	063660	012737	000000	177660		MOV	#0,UDPAR0	:
8694	063666	012737	000200	177662		MOV	#200,UDPAR1	:
8695	063674	012737	000400	177664		MOV	#400,UDPAR2	:
8696	063702	012737	000600	177666		MOV	#600,UDPAR3	:
8697	063710	012737	001000	177670		MOV	#1000,UDPAR4	:
8698	063716	012737	001000	177672		MOV	#1000,UDPAR5	:
8699	063724	012737	001000	177674		MOV	#1000,UDPAR6	:
8700	063732	012737	177600	177676		MOV	#177600,UDPAR7	:
8701	063740	012737	063756	001110		MOV	#20\$,SLPADR	:SET LOOP ON TEST POINTER TO 20\$
8702	063746	012737	063756	001112		MOV	#20\$,SLPERR	:SET LOOP ON ERROR POINTER TO 20\$
8703	063754	104416				TBITO		:MAKE SURE THAT T-BIT IS OFF FOR
8704								:THE SIX DUAL MAPPING TESTS
8705	063756	012737	064126	001112	20\$:	MOV	#21\$,SLPERR	:SET LOOP ON ERROR POINTER TO 21\$
8706	063764	012737	000000	177776		MOV	#00000,PSW	:GO TO KERNEL MODE
8707	063772	012703	172300			MOV	#KIPDR0,R3	:LOAD FIRST ADDRESS OF KERNEL PDR'S
8708								:THAT WILL BE TESTED IN THIS TEST
8709	063776	012704	000010			MOV	#10,R4	:TEST THE NEXT EIGHT PDR'S
8710	064002	012705	010200			MOV	#10200,R5	:LOAD STARTING VIRTUAL ADDRESS INTO R5
8711	064006	012700	077405		19\$:	MOV	#77405,R0	:ALL PAGES WILL BE TRAP ON WRITE
8712	064012	005037	001172			CLR	\$TMP0	:CLEAR CORRECT PDR SET INDICATOR
8713	064016	012702	000010			MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
8714	064022	012701	177620			MOV	#UDPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
8715	064026	010021			1\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
8716	064030	077202				SOB	R2,1\$:BRANCH BACK TO 1\$ IF R2 IS NOT ZERO
8717	064032	012702	000010			MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
8718	064036	012701	177600			MOV	#UIPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
8719	064042	010021			2\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
8720	064044	077202				SOB	R2,2\$:BRANCH BACK TO 2\$ IF R2 IS NOT ZERO
8721	064046	012702	000010			MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
8722	064052	012701	172220			MOV	#SDPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1

8723	064056	010021		3\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
8724	064060	077202			SOB	R2,3\$:BRANCH BACK TO 3\$ IF R2 IS NOT ZERO
8725	064062	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
8726	064066	012701	172200		MOV	#SIPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
8727	064072	010021		4\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
8728	064074	077202			SOB	R2,4\$:BRANCH BACK TO 4\$ IF R2 IS NOT ZERO
8729	064076	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
8730	064102	012701	172320		MOV	#KDPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
8731	064105	010021		5\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
8732	064110	077202			SOB	R2,5\$:BRANCH BACK TO 5\$ IF R2 IS NOT ZERO
8733	064112	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
8734	064116	012701	172300		MOV	#KIPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
8735	064122	010021		6\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
8736	064124	077202			SOB	R2,6\$:BRANCH BACK TO 6\$ IF R2 IS NOT ZERO
8737	064126	012700	077405	21\$:	MOV	#77405,R0	:MUST RE-INIT THESE PDRS IF ERROR
8738	064132	010037	172300		MOV	R0,KIPDR0	:LOAD KERNEL PDR0
8739	064136	010037	172302		MOV	R0,KIPDR1	:LOAD KERNEL PDR1
8740	064142	010037	172316		MOV	R0,KIPDR7	:LOAD KERNEL PDR7
8741	064146	010037	172300		MOV	R0,KIPDR0	:LOAD PDR0 OF PRESENT SPACE
8742	064152	010037	172316		MOV	R0,KIPDR7	:LOAD PDR7 OF PRESENT SPACE
8743	064156	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
8744	064160	011515			MOV	(R5),(R5)	:WRITE INTO PAGE UNDER TEST
8745	064162	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
8746	064166	012701	172300		MOV	#KIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
8747	064172	011100		7\$:	MOV	(R1),R0	:READ PDR INTO R0
8748	064174	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
8749	064200	001023			BNE	9\$:BRANCH IF THIS IS NOT THE ONE
8750	064202	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8751	064204	001417			BEQ	8\$:BRANCH IF ADDRESS IS CORRECT
8752	064206	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR
8753	064210	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
8754	064214	010037	172300		MOV	R0,KIPDR0	:RELOAD KERNEL PDR0
8755	064220	010037	172302		MOV	R0,KIPDR1	:RELOAD KERNEL PDR1
8756	064224	010037	172316		MOV	R0,KIPDR7	:RELOAD KERNEL PDR7
8757	064230	010037	172300		MOV	R0,KIPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
8758	064234	010037	172316		MOV	R0,KIPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
8759	064240	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
8760							:WERE TESTING PAGE SEVEN
8761	064242	000402			BR	9\$:GO UPDATE R1 FOR NEXT READ
8762	064244	005237	001172	8\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
8763	064250	062701	000002	9\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
8764	064254	077232			SOB	R2,7\$:BRANCH TO 7\$ IF ALL PDR'S NOT READ
8765	064256	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
8766	064262	012701	172200		MOV	#SIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
8767	064266	011100		10\$:	MOV	(R1),R0	:READ PDR INTO R0
8768	064270	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
8769	064274	001023			BNE	12\$:BRANCH IF THIS IS NOT THE ONE
8770	064276	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8771	064300	001417			BEQ	11\$:BRANCH IF ADDRESS IS CORRECT
8772	064302	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR
8773	064304	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
8774	064310	010037	172300		MOV	R0,KIPDR0	:RELOAD KERNEL PDR0
8775	064314	010037	172302		MOV	R0,KIPDR1	:RELOAD KERNEL PDR1
8776	064320	010037	172316		MOV	R0,KIPDR7	:RELOAD KERNEL PDR7
8777	064324	010037	172300		MOV	R0,KIPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
8778	064330	010037	172316		MOV	R0,KIPDR7	:RE-LOAD I/O PAGE PDR IF ERROR

8779	064334	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
8780							:WERE TESTING PAGE SEVEN
8781	064336	000402			BR	12\$:GO UPDATE R1 FOR NEXT READ
8782	064340	005237	001172	11\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
8783	064344	062701	000002	12\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
8784	064350	077232			SOB	R2,10\$:BRANCH TO 10\$ IF ALL PDR'S NOT READ
8785	064352	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
8786	064356	012701	177600		MOV	#UIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
8787	064362	011100		13\$:	MOV	(R1),R0	:READ PDR INTO R0
8788	064364	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH ABW BITS ON
8789	064370	001023			BNE	15\$:BRANCH IF THIS IS NOT THE ONE
8790	064372	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8791	064374	001417			BEQ	14\$:BRANCH IF ADDRESS IS CORRECT
8792	064376	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR
8793	064400	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
8794	064404	010037	172300		MOV	R0,KIPDR0	:RELOAD KERNEL PDR0
8795	064410	010037	172302		MOV	R0,KIPDR1	:RELOAD KERNEL PDR1
8796	064414	010037	172316		MOV	R0,KIPDR7	:RELOAD KERNEL PDR7
8797	064420	010037	172300		MOV	R0,KIPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
8798	064424	010037	172316		MOV	R0,KIPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
8799	064430	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
8800							:WERE TESTING PAGE SEVEN
8801	064432	000402			BR	15\$:GO UPDATE R1 FOR NEXT READ
8802	064434	005237	001172	14\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
8803	064440	062701	000002	15\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
8804	064444	077232			SOB	R2,13\$:BRANCH TO 13\$ IF ALL PDR'S NOT READ
8805	064446	005737	001172		TST	\$TMP0	:SEE IF THERE WAS A CORRECT PDR
8806	064452	001002			BNE	16\$:BRANCH IF THERE WAS
8807	064454	011300			MOV	(R3),R0	:SAVE CONTENTS OF PDR UNDER TEST
8808	064456	104113			ERROR	113	:NO PDR ADDRESSES MATCHED
8809	064460	062703	000002	16\$:	ADD	#2,R3	:POINT TO NEXT PDR UNDER TEST
8810	064464	062705	020000		ADD	#20000,R5	:CHANGE PAGE NUMBER IN VIRT. ADDR.
8811	064470	005304			DEC	R4	:DECREMENT COUNTER
8812	064472	001402			BEQ	17\$:BRANCH IF COUNTER IS ZERO
8813	064474	000137	064006		JMP	19\$:JUMP TO LOAD PDR'S AGAIN
8814	064500			17\$:			
8815	064500	012737	000340	177776	MOV	#340,PSW	:RETURN TO KERNEL MODE, PRIORITY 7
8816	064506	012737	063756	001112	MOV	#20\$,SLPERR	:SET LOOP POINTER TO START OF TEST

```

8817
8818
8819
8820 *****
      *TEST 101      DUAL MAPPING SUPERVISOR MODE I-SPACE
8821
8822
8823      THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405
8824      (4K PAGE, TRAP ON WRITE). THEN THE VIRTUAL ADDRESS IS SET TO
8825      010200 AND THAT WORD IS WRITTEN INTO ITSELF. THIS WRITE WILL
8826      SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP
8827      SINCE BIT09 OF MMRO IS CLEAR) AND SET BOTH THE A & W BITS IN
8828      THE SUPERVISOR I-SPACE P.D.R. UNDER TEST. NOW ALL OF THE
8829      P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE
8830      ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
8831      WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
8832      ADDRESS OF 'MAPLOO' (17770200) WHICH SHOULD ALWAYS EXIST.
8833 *****
8834 TST101:
    
```

064514

Address	Offset	Label	Operation	Comments
8891	065002	010037	172216	MOV R0,SIPDR7 ;RE-LOAD I/O PAGE PDR IF ERROR
8892	065006	011515		MOV (R5),(R5) ;TRY WRITE AGAIN, IN CASE YOU
8893				WERE TESTING PAGE SEVEN
8894	065010	000402		BR 9\$;UPDATE R1 FOR NEXT READ
8895	065012	005237	001172	8\$: INC \$TMP0 ;SET FLAG SINCE ADDRESSES MATCHED
8896	065016	062701	000002	9\$: ADD #2,R1 ;POINT TO NEXT PDR TO BE READ
8897	065022	077232		SOB R2,7\$;BRANCH TO 7\$ IF ALL PDR'S NOT READ
8898	065024	012702	000020	MOV #20,R2 ;SET COUNTER TO READ NEXT 20 REGISTERS
8899	065030	012701	172200	MOV #SIPDR0,R1 ;LOAD ADDRESS OF BEGINNING PDR
8900	065034	011100		MOV (R1),R0 ;READ PDR INTO R0
8901	065036	022700	077705	10\$: CMP #77705,R0 ;SEE IF THIS WAS THE PDR WITH ABW BITS ON
8902	065042	001023		BNE 12\$;BRANCH IF THIS IS NOT THE ONE
8903	065044	020103		CMP R1,R3 ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8904	065046	001417		BEQ 11\$;BRANCH IF ADDRESS IS CORRECT
8905	065050	104112		ERROR 112 ;A & W BITS GOT SET IN WRONG PDR
8906	065052	012700	077405	MOV #77405,R0 ;RE-SET PAGES MODIFIED BY ERROR
8907	065056	010037	172300	MOV R0,KIPDR0 ;RELOAD KERNEL PDR0
8908	065062	010037	172302	MOV R0,KIPDR1 ;RELOAD KERNEL PDR1
8909	065066	010037	172316	MOV R0,KIPDR7 ;RELOAD KERNEL PDR7
8910	065072	010037	172200	MOV R0,SIPDR0 ;RELOAD PAGE 0 OF PRESENT SPACE
8911	065076	010037	172216	MOV R0,SIPDR7 ;RE-LOAD I/O PAGE PDR IF ERROR
8912	065102	011515		MOV (R5),(R5) ;TRY WRITE AGAIN, IN CASE YOU
8913				WERE TESTING PAGE SEVEN
8914	065104	000402		BR 12\$;GO UPDATE R1 FOR NEXT READ
8915	065106	005237	001172	11\$: INC \$TMP0 ;SET FLAG SINCE ADDRESSES MATCHED
8916	065112	062701	000002	12\$: ADD #2,R1 ;POINT TO NEXT PDR TO BE READ
8917	065116	077232		SOB R2,10\$;BRANCH TO 10\$ IF ALL PDR'S NOT READ
8918	065120	012702	000020	MOV #20,R2 ;SET COUNTER TO READ NEXT 20 REGISTERS
8919	065124	012701	177600	MOV #KIPDR0,R1 ;LOAD ADDRESS OF BEGINNING PDR
8920	065130	011100		MOV (R1),R0 ;READ PDR INTO R0
8921	065132	022700	077705	13\$: CMP #77705,R0 ;SEE IF THIS WAS THE PDR WITH ABW BITS ON
8922	065136	001023		BNE 15\$;BRANCH IF THIS IS NOT THE ONE
8923	065140	020103		CMP R1,R3 ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8924	065142	001417		BEQ 14\$;BRANCH IF ADDRESS IS CORRECT
8925	065144	104112		ERROR 112 ;A & W BITS GOT SET IN WRONG PDR
8926	065146	012700	077405	MOV #77405,R0 ;RE-SET PAGES MODIFIED BY ERROR
8927	065152	010037	172300	MOV R0,KIPDR0 ;RELOAD KERNEL PDR0
8928	065156	010037	172302	MOV R0,KIPDR1 ;RELOAD KERNEL PDR1
8929	065162	010037	172316	MOV R0,KIPDR7 ;RELOAD KERNEL PDR7
8930	065166	010037	172200	MOV R0,SIPDR0 ;RELOAD PAGE 0 OF PRESENT SPACE
8931	065172	010037	172216	MOV R0,SIPDR7 ;RE-LOAD I/O PAGE PDR IF ERROR
8932	065176	011515		MOV (R5),(R5) ;TRY WRITE AGAIN, IN CASE YOU
8933				WERE TESTING PAGE SEVEN
8934	065200	000402		BR 15\$;GO UPDATE R1 FOR NEXT READ
8935	065202	005237	001172	14\$: INC \$TMP0 ;SET FLAG SINCE ADDRESSES MATCHED
8936	065206	062701	000002	15\$: ADD #2,R1 ;POINT TO NEXT PDR TO BE READ
8937	065212	077232		SOB R2,13\$;BRANCH TO 13\$ IF ALL PDR'S NOT READ
8938	065214	005737	001172	TST \$TMP0 ;SEE IF THERE WAS A CORRECT PDR
8939	065220	001002		BNE 16\$;BRANCH IF THERE WAS
8940	065222	011300		MOV (R3),R0 ;SAVE CONTENTS OF PDR UNDER TEST
8941	065224	104113		ERROR 113 ;NO PDR ADDRESSES MATCHED
8942	065226	062703	000002	16\$: ADD #2,R3 ;POINT TO NEXT PDR UNDER TEST
8943	065232	062705	020000	ADD #20000,R5 ;CHANGE PAGE NUMBER IN VIRT. ADDR.
8944	065236	005304		DEC R4 ;DECREMENT COUNTER
8945	065240	001402		BEQ 17\$;BRANCH IF COUNTER IS ZERO
8946	065242	000137	064554	JMP 19\$;JUMP TO LOAD PDR'S AGAIN

```

8947 065246
8948 065246 012737 000340 177776
8949 065254 012737 064524 001112
8950
8951
8952
8953
8954
8955
8956
8957
8958
8959
8960
8961
8962
8963
8964
8965
8966
8967 065262
8968 065262 000004
8969 065264 012737 066030 001316
8970
8971 065272 012737 065442 001112
8972 065300 012737 140000 177776
8973 065306 012703 177600
8974
8975 065312 012704 000010
8976 065316 012705 010200
8977 065322 012700 077405
8978 065326 005037 001172
8979 065332 012702 000010
8980 065336 012701 172220
8981 065342 010021
8982 065344 077202
8983 065346 012702 000010
8984 065352 012701 172200
8985 065356 010021
8986 065360 077202
8987 065362 012702 000010
8988 065366 012701 172320
8989 065372 010021
8990 065374 077202
8991 065376 012702 000010
8992 065402 012701 172300
8993 065406 010021
8994 065410 077202
8995 065412 012702 000010
8996 065416 012701 177620
8997 065422 010021
8998 065424 077202
8999 065426 012702 000010
9000 065432 012701 177600
9001 065436 010021
9002 065440 077202

17$:
MOV #340,PSW ;RETURN TO KERNEL MODE, PRIORITY 7
MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST

*****
*TEST 102 DUAL MAPPING USER MODE I-SPACE
*****
THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405
(4K PAGE, TRAP ON WRITE). THEN THE VIRTUAL ADDRESS IS SET TO
010200 AND THAT WORD IS WRITTEN INTO ITSELF. THIS WRITE WILL
SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP
SINCE BIT09 OF MMRO IS CLEAR) AND SET BOTH THE A & W BITS IN
THE USER I-SPACE P.D.R. UNDER TEST. NOW ALL OF THE
P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE
ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
ADDRESS OF 'MAPLOO' (17770200) WHICH SHOULD ALWAYS EXIST.
*****
TST102:
SCOPE
MOV #TST103,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
;SET LOOP ON ERROR POINTER TO 21$
20$: MOV #21$,SLPERR
MOV #140000,PSW ;GO TO USER MODE
MOV #UIPDR0,R3 ;LOAD FIRST ADDRESS OF USER PDR'S
;THAT WILL BE TESTED IN THIS TEST
;TEST THE NEXT EIGHT PDR'S
MOV #10,R4 ;LOAD STARTING VIRTUAL ADDRESS INTO R5
MOV #10200,R5 ;ALL PAGES WILL BE TRAP ON WRITE
19$: MOV #77405,R0 ;CLEAR CORRECT PDR SET INDICATOR
CLR $TMP0 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #SDPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
;LOAD R0 INTO PDR ADDRESSED BY R1
1$: MOV R0,(R1)+
SOB R2,1$ ;BRANCH BACK TO 1$ IF R2 IS NOT ZERO
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #SIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
;LOAD R0 INTO PDR ADDRESSED BY R1
2$: MOV R0,(R1)+
SOB R2,2$ ;BRANCH BACK TO 2$ IF R2 IS NOT ZERO
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #KDPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
;LOAD R0 INTO PDR ADDRESSED BY R1
3$: MOV R0,(R1)+
SOB R2,3$ ;BRANCH BACK TO 3$ IF R2 IS NOT ZERO
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #KIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
;LOAD R0 INTO PDR ADDRESSED BY R1
4$: MOV R0,(R1)+
SOB R2,4$ ;BRANCH BACK TO 4$ IF R2 IS NOT ZERO
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #UDPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
;LOAD R0 INTO PDR ADDRESSED BY R1
5$: MOV R0,(R1)+
SOB R2,5$ ;BRANCH BACK TO 5$ IF R2 IS NOT ZERO
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #UIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
;LOAD R0 INTO PDR ADDRESSED BY R1
6$: MOV R0,(R1)+
SOB R2,6$ ;BRANCH BACK TO 6$ IF R2 IS NOT ZERO
    
```

```

9003 065442 012700 077405 21$: MOV #77405,R0 ;MUST RE-INIT THESE PDRS IF ERROR
9004 065446 010037 172300 MOV R0,KIPDR0 ;LOAD KERNEL PDR0
9005 065452 010037 172302 MOV R0,KIPDR1 ;LOAD KERNEL PDR1
9006 065456 010037 172316 MOV R0,KIPDR7 ;LOAD KERNEL PDR7
9007 065462 010037 177600 MOV R0,UIPDR0 ;LOAD PDR0 OF PRESENT SPACE
9008 065466 010037 177616 MOV R0,UIPDR7 ;LOAD PDR7 OF PRESENT SPACE
9009 065472 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
9010 065474 011515 MOV (R5),(R5) ;WRITE INTO PAGE UNDER TEST
9011 065476 012702 000020 MOV #20,R2 ;SET COUNTER TO READ NEXT 20 REGISTERS
9012 065502 012701 172300 MOV #KIPDR0,R1 ;LOAD ADDRESS OF BEGINNING PDR
9013 065506 011100 7$: MOV (R1),R0 ;READ PDR INTO R0
9014 065510 022700 077705 CMP #77705,R0 ;SEE IF THIS WAS THE PDR WITH ABW BITS ON
9015 065514 001023 BNE 9$ ;BRANCH IF THIS IS NOT THE ONE
9016 065516 020103 CMP R1,R3 ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9017 065520 001417 BEQ 8$ ;BRANCH IF ADDRESS IS CORRECT
9018 065522 104112 ERROR 112 ;A & W BITS GOT SET IN WRONG PDR
9019 065524 012700 077405 MOV #77405,R0 ;RE-SET PAGES MODIFIED BY ERROR
9020 065530 010037 172300 MOV R0,KIPDR0 ;RELOAD KERNEL PDR0
9021 065534 010037 172302 MOV R0,KIPDR1 ;RELOAD KERNEL PDR1
9022 065540 010037 172316 MOV R0,KIPDR7 ;RELOAD KERNEL PDR7
9023 065544 010037 177600 MOV R0,UIPDR0 ;RELOAD PAGE 0 OF PRESENT SPACE
9024 065550 010037 177616 MOV R0,UIPDR7 ;RE-LOAD I/O PAGE PDR IF ERROR
9025 065554 011515 MOV (R5),(R5) ;TRY WRITE AGAIN, IN CASE YOU
9026 ;WERE TESTING PAGE SEVEN
9027 065556 000402 BR 9$ ;GO UPDATE R1 FOR NEXT READ
9028 065560 005237 001172 8$: INC $TMP0 ;SET FLAG SINCE ADDRESSES MATCHED
9029 065564 062701 000002 9$: ADD #2,R1 ;POINT TO NEXT PDR TO BE READ
9030 065570 077232 SOB R2,7$ ;BRANCH TO 7$ IF ALL PDR'S NOT READ
9031 065572 012702 000020 MOV #20,R2 ;SET COUNTER TO READ NEXT 20 REGISTERS
9032 065576 012701 172200 MOV #SIPDR0,R1 ;LOAD ADDRESS OF BEGINNING PDR
9033 065602 011100 10$: MOV (R1),R0 ;READ PDR INTO R0
9034 065604 022700 077705 CMP #77705,R0 ;SEE IF THIS WAS THE PDR WITH ABW BITS ON
9035 065610 001023 BNE 12$ ;BRANCH IF THIS IS NOT THE ONE
9036 065612 020103 CMP R1,R3 ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9037 065614 001417 BEQ 11$ ;BRANCH IF ADDRESS IS CORRECT
9038 065616 104112 ERROR 112 ;A & W BITS GOT SET IN WRONG PDR
9039 065620 012700 077405 MOV #77405,R0 ;RE-SET PAGES MODIFIED BY ERROR
9040 065624 010037 172300 MOV R0,KIPDR0 ;RELOAD KERNEL PDR0
9041 065630 010037 172302 MOV R0,KIPDR1 ;RELOAD KERNEL PDR1
9042 065634 010037 172316 MOV R0,KIPDR7 ;RELOAD KERNEL PDR7
9043 065640 010037 177600 MOV R0,UIPDR0 ;RELOAD PAGE 0 OF PRESENT SPACE
9044 065644 010037 177616 MOV R0,UIPDR7 ;RE-LOAD I/O PAGE PDR IF ERROR
9045 065650 011515 MOV (R5),(R5) ;TRY WRITE AGAIN, IN CASE YOU
9046 ;WERE TESTING PAGE SEVEN
9047 065652 000402 BR 12$ ;GO UPDATE R1 FOR NEXT READ
9048 065654 005237 001172 11$: INC $TMP0 ;SET FLAG SINCE ADDRESSES MATCHED
9049 065660 062701 000002 12$: ADD #2,R1 ;POINT TO NEXT PDR TO BE READ
9050 065664 077232 SOB R2,10$ ;BRANCH TO 10$ IF ALL PDR'S NOT READ
9051 065666 012702 000020 MOV #20,R2 ;SET COUNTER TO READ NEXT 20 REGISTERS
9052 065672 012701 177600 MOV #UIPDR0,R1 ;LOAD ADDRESS OF BEGINNING PDR
9053 065676 011100 13$: MOV (R1),R0 ;READ PDR INTO R0
9054 065700 022700 077705 CMP #77705,R0 ;SEE IF THIS WAS THE PDR WITH ABW BITS ON
9055 065704 001023 BNE 15$ ;BRANCH IF THIS IS NOT THE ONE
9056 065706 020103 CMP R1,R3 ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9057 065710 001417 BEQ 14$ ;BRANCH IF ADDRESS IS CORRECT
9058 065712 104112 ERROR 112 ;A & W BITS GOT SET IN WRONG PDR
  
```


9059	065714	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
9060	065720	010037	172300		MOV	R0,KIPDR0	:RELOAD KERNEL PDR0
9061	065724	010037	172302		MOV	R0,KIPDR1	:RELOAD KERNEL PDR1
9062	065730	010037	172316		MOV	R0,KIPDR7	:RELOAD KERNEL PDR7
9063	065734	010037	177600		MOV	R0,UIPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
9064	065740	010037	177616		MOV	R0,UIPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
9065	065744	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
9066							:WERE TESTING PAGE SEVEN
9067	065746	000402			BR	15\$:GO UPDATE R1 FOR NEXT READ
9068	065750	005237	001172	14\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
9069	065754	062701	000002	15\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
9070	065760	077232			SOB	R2,13\$:BRANCH TO 13\$ IF ALL PDR'S NOT READ
9071	065762	005737	001172		TST	\$TMP0	:SEE IF THERE WAS A CORRECT PDR
9072	065766	001002			BNE	16\$:BRANCH IF THERE WAS
9073	065770	011300			MOV	(R3),R0	:SAVE CONTENTS OF PDR UNDER TEST
9074	065772	104113			ERROR	113	:NO PDR ADDRESSES MATCHED
9075	065774	062703	000002	16\$:	ADD	#2,R3	:POINT TO NEXT PDR UNDER TEST
9076	066000	062705	020000		ADD	#20000,R5	:CHANGE PAGE NUMBER IN VIRT. ADDR.
9077	066004	005304			DEC	R4	:DECREMENT COUNTER
9078	066006	001402			BEQ	17\$:BRANCH IF COUNTER IS ZERO
9079	066010	000137	065322		JMP	19\$:JUMP TO LOAD PDR'S AGAIN
9080	066014			17\$:			
9081	066014	012737	000340	177776	MOV	#340,PSW	:RETURN TO KERNEL MODE, PRIORITY 7
9082	066022	012737	065272	001112	MOV	#20\$,SLPERR	:SET LOOP POINTER TO START OF TEST

 *TEST 103 DUAL MAPPING KERNEL MODE D-SPACE

THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405 (4K PAGE, TRAP ON WRITE). THEN THE VIRTUAL ADDRESS IS SET TO 010200 AND THAT WORD IS WRITTEN INTO ITSELF. THIS WRITE WILL SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP SINCE BIT09 OF MPRO IS CLEAR) AND SET BOTH THE A & W BITS IN THE KERNEL D-SPACE P.D.R. UNDER TEST. NOW ALL OF THE P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS. WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE ADDRESS OF 'MAPLOO' (17770200) WHICH SHOULD ALWAYS EXIST.

 TST103:

9100	066030				SCOPE		
9101	066030	000004			MOV	#TST104,NXTTST	:SAVE STARTING ADDRESS OF NEXT
9102	066032	012737	066612	001316			:TEST FOR ESCAPE ON PARITY ERRORS
9103							:SET LOOP ON ERROR POINTER TO 21\$
9104	066040	012737	066216	001112	20\$:	MOV	#21\$,SLPERR
9105	066046	012737	000000	177776		MOV	#00000,PSW
9106	066054	052737	000007	172516		BIS	#7,MPR3
9107	066062	012703	172320			MOV	#KDPDR0,R3
9108							:ENABLE ALL D-SPACE MAPPING
9109	066066	012704	000010				:LOAD FIRST ADDRESS OF KERNEL PDR'S
9110	066072	012705	010200				:THAT WILL BE TESTED IN THIS TEST
9111	066076	012700	077405				:TEST THE NEXT EIGHT PDR'S
9112	066102	005037	001172		19\$:	MOV	#10,R4
9113	066106	012702	000010			MOV	#10200,R5
9114	066112	012701	177600			MOV	#77405,R0
						CLR	\$TMP0
						MOV	#10,R2
						MOV	#UIPDR0,R1

:ALL PAGES WILL BE TRAP ON WRITE
 :CLEAR CORRECT PDR SET INDICATOR
 :SET COUNT TO LOAD 8 ADDRESSES
 :PUT ADDRESS OF FIRST PDR IN R1

9115	066116	010021		1\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9116	066120	077202			SOB	R2,1\$:BRANCH BACK TO 1\$ IF R2 IS NOT ZERO
9117	066122	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
9118	066126	012701	177620		MOV	#UDPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
9119	066132	010021		2\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9120	066134	077202			SOB	R2,2\$:BRANCH BACK TO 2\$ IF R2 IS NOT ZERO
9121	066136	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
9122	066142	012701	172200		MOV	#SIPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
9123	066146	010021		3\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9124	066150	077202			SOB	R2,3\$:BRANCH BACK TO 3\$ IF R2 IS NOT ZERO
9125	066152	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
9126	066156	012701	172220		MOV	#SDPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
9127	066162	010021		4\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9128	066164	077202			SOB	R2,4\$:BRANCH BACK TO 4\$ IF R2 IS NOT ZERO
9129	066166	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
9130	066172	012701	172300		MOV	#KIPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
9131	066176	010021		5\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9132	066200	077202			SOB	R2,5\$:BRANCH BACK TO 5\$ IF R2 IS NOT ZERO
9133	066202	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
9134	066206	012701	172320		MOV	#KDPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
9135	066212	010021		6\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9136	066214	077202			SOB	R2,6\$:BRANCH BACK TO 6\$ IF R2 IS NOT ZERO
9137	066216	012700	077405	21\$:	MOV	#77405,R0	:MUST RE-INIT THESE PDRS IF ERROR
9138	066222	010037	172320		MOV	RC,KDPDR0	:LOAD KERNEL PDR0
9139	066226	010037	172322		MOV	R0,KDPDR1	:LOAD KERNEL PDR1
9140	066232	010037	172336		MOV	R0,KDPDR7	:LOAD KERNEL PDR7
9141	066236	010037	172320		MOV	R0,KDPDR0	:LOAD PDR0 OF PRESENT SPACE
9142	066242	010037	172336		MOV	R0,KDPDR7	:LOAD PDR7 OF PRESENT SPACE
9143	066246	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
9144	066250	011515			MOV	(R5),(R5)	:WRITE INTO PAGE UNDER TEST
9145	066252	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
9146	066256	012701	172300		MOV	#KIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
9147	066262	011100		7\$:	MOV	(R1),R0	:READ PDR INTO R0
9148	066264	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH ABW BITS ON
9149	066270	001023			BNE	9\$:BRANCH IF THIS IS NOT THE ONE
9150	066272	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9151	066274	001417			BEQ	8\$:BRANCH IF ADDRESS IS CORRECT
9152	066276	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR
9153	066300	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
9154	066304	010037	172320		MOV	R0,KDPDR0	:RELOAD KERNEL PDR0
9155	066310	010037	172322		MOV	R0,KDPDR1	:RELOAD KERNEL PDR1
9156	066314	010037	172336		MOV	R0,KDPDR7	:RELOAD KERNEL PDR7
9157	066320	010037	172320		MOV	R0,KDPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
9158	066324	010037	172336		MOV	R0,KDPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
9159	066330	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
9160							:WERE TESTING PAGE SEVEN
9161	066332	000402			BR	9\$:GO UPDATE R1 FOR NEXT READ
9162	066334	005237	001172	8\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
9163	066340	062701	000002	9\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
9164	066344	077232			SOB	R2,7\$:BRANCH TO 7\$ IF ALL PDR'S NOT READ
9165	066346	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
9166	066352	012701	172200		MOV	#SIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
9167	066356	011100		10\$:	MOV	(R1),R0	:READ PDR INTO R0
9168	066360	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH ABW BITS ON
9169	066364	001023			BNE	12\$:BRANCH IF THIS IS NOT THE ONE
9170	066366	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST


```

9219
9220
9221
9222
9223
9224
9225
9226
9227
9228
9229
9230
9231
9232
9233
9234
9235 066612
9236 066612 000004
9237 066614 012737 067374 001316
9238
9239 066622 012737 067000 001112
9240 066630 012737 040000 177776
9241 066636 052737 000007 172516
9242 066644 012703 172220
9243
9244 066650 012704 000010
9245 066654 012705 010200
9246 066660 012700 077405
9247 066664 005037 001172
9248 066670 012702 000010
9249 066674 012701 172300
9250 066700 010021
9251 066702 077202
9252 066704 012702 000010
9253 066710 012701 172320
9254 066714 010021
9255 066716 077202
9256 066720 012702 000010
9257 066724 012701 177600
9258 066730 010021
9259 066732 077202
9260 066734 012702 000010
9261 066740 012701 177620
9262 066744 010021
9263 066746 077202
9264 066750 012702 000010
9265 066754 012701 172200
9266 066760 010021
9267 066762 077202
9268 066764 012702 000010
9269 066770 012701 172220
9270 066774 010021
9271 066776 077202
9272 067000 012700 077405
9273 067004 010037 172320
9274 067010 010037 172322

```

```

*****
*TEST 104      DUAL MAPPING SUPERVISOR MODE D-SPACE
*****
THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405
(4K PAGE, TRAP ON WRITE). THEN THE VIRTUAL ADDRESS IS SET TO
010200 AND THAT WORD IS WRITTEN INTO ITSELF. THIS WRITE WILL
SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP
SINCE BIT09 OF MMRO IS CLEAR) AND SET BOTH THE A & W BITS IN
THE SUPERVISOR D-SPACE P.D.R. UNDER TEST. NOW ALL OF THE
P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE
ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
ADDRESS OF 'MAPLOO' (17770200) WHICH SHOULD ALWAYS EXIST.
*****

```

```

*****
TST104:
SCOPE
MOV #TST105,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
;SET LOOP ON ERROR POINTER TO 21$
MOV #21$,SLPERR
MOV #40000,PSW ;GO TO SUPERVISOR MODE
BIS #7,MMR3 ;ENABLE ALL D-SPACE MAPPING
MOV #SDPDR0,R3 ;LOAD FIRST ADDRESS OF SUPERVISOR PDR'S
;THAT WILL BE TESTED IN THIS TEST
MOV #10,R4 ;TEST THE NEXT EIGHT PDR'S
MOV #10200,R5 ;LOAD STARTING VIRTUAL ADDRESS INTO R5
MOV #77405,R0 ;ALL PAGES WILL BE TRAP ON WRITE
19$: CLR $TMP0 ;CLEAR CORRECT PDR SET INDICATOR
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #KIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
13$: MOV R0,(R1)+ ;LOAD R0 INTO PDR ADDRESSED BY R1
SOB R2,1$ ;BRANCH BACK TO 1$ IF R2 IS NOT ZERO
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #KDPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
2$: MOV R0,(R1)+ ;LOAD R0 INTO PDR ADDRESSED BY R1
SOB R2,2$ ;BRANCH BACK TO 2$ IF R2 IS NOT ZERO
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #UIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
3$: MOV R0,(R1)+ ;LOAD R0 INTO PDR ADDRESSED BY R1
SOB R2,3$ ;BRANCH BACK TO 3$ IF R2 IS NOT ZERO
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #LDPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
4$: MOV R0,(R1)+ ;LOAD R0 INTO PDR ADDRESSED BY R1
SOB R2,4$ ;BRANCH BACK TO 4$ IF R2 IS NOT ZERO
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #SIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
5$: MOV R0,(R1)+ ;LOAD R0 INTO PDR ADDRESSED BY R1
SOB R2,5$ ;BRANCH BACK TO 5$ IF R2 IS NOT ZERO
MOV #,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #SDPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
6$: MOV R0,(R1)+ ;LOAD R0 INTO PDR ADDRESSED BY R1
SOB R2,6$ ;BRANCH BACK TO 6$ IF R2 IS NOT ZERO
21$: MOV #77405,R0 ;MUST RE-INIT THESE PDRS IF ERROR
MOV R0,KDPDR0 ;LOAD KERNEL PDR0
MOV R0,KDPDR1 ;LOAD KERNEL PDR1

```

```

9275 067014 010037 172336      MOV      R0,KDPDR7      ;LOAD KERNEL PDR7
9276 067020 010037 172220      MOV      R0,SDPDR0      ;LOAD PDR0 OF PRESENT SPACE
9277 067024 010037 172236      MOV      R0,SDPDR7      ;LOAD PDR7 OF PRESENT SPACE
9278 067030 000240                NOP                    ;THIS IS A SYNC POINT FOR SCOPING
9279 067032 011515                MOV      (R5),(R5)      ;WRITE INTO PAGE UNDER TEST
9280 067034 012702 000020      MOV      #20,R2         ;SET COUNTER TO READ NEXT 20 REGISTERS
9281 067040 012701 172300      MOV      #KIPDR0,R1     ;LOAD ADDRESS OF BEGINNING PDR
9282 067044 011100                MOV      (R1),R0        ;READ PDR INTO R0
9283 067046 022700 077705      CMP      #77705,R0      ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
9284 067052 001023                BNE      9$             ;BRANCH IF THIS IS NOT THE ONE
9285 067054 020103                CMP      R1,R3          ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9286 067056 001417                BEQ      8$             ;BRANCH IF ADDRESS IS CORRECT
9287 067060 104112                ERROR    112           ;A & W BITS GOT SET IN WRONG PDR
9288 067062 012700 077405      MOV      #77405,R0      ;RE-SET PAGES MODIFIED BY ERROR
9289 067066 010037 172320      MOV      R0,KDPDR0      ;RELOAD KERNEL PDR0
9290 067072 010037 172322      MOV      R0,KDPDR1      ;RELOAD KERNEL PDR1
9291 067076 010037 172336      MOV      R0,KDPDR7      ;RELOAD KERNEL PDR7
9292 067102 010037 172220      MOV      R0,SDPDR0      ;RELOAD PAGE 0 OF PRESENT SPACE
9293 067106 010037 172236      MOV      R0,SDPDR7      ;RE-LOAD I/O PAGE PDR IF ERROR
9294 067112 011515                MOV      (R5),(R5)      ;TRY WRITE AGAIN, IN CASE YOU
9295                                ;WERE TESTING PAGE SEVEN
9296 067114 000402                BR       9$             ;GO UPDATE R1 FOR NEXT READ
9297 067116 005237 001172      8$: INC      $TMP0        ;SET FLAG SINCE ADDRESSES MATCHED
9298 067122 062701 000002      9$: ADD      #2,R1        ;POINT TO NEXT PDR TO BE READ
9299 067126 077232                SOB      R2,7$         ;BRANCH TO 7$ IF ALL PDR'S NOT READ
9300 067130 012702 000020      MOV      #20,R2         ;SET COUNTER TO READ NEXT 20 REGISTERS
9301 067134 012701 172200      MOV      #SIPDR0,R1     ;LOAD ADDRESS OF BEGINNING PDR
9302 067140 011100                MOV      (R1),R0        ;READ PDR INTO R0
9303 067142 022700 077705      CMP      #77705,R0      ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
9304 067146 001023                BNE      12$           ;BRANCH IF THIS IS NOT THE ONE
9305 067150 020103                CMP      R1,R3          ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9306 067152 001417                BEQ      11$           ;BRANCH IF ADDRESS IS CORRECT
9307 067154 104112                ERROR    112           ;A & W BITS GOT SET IN WRONG PDR
9308 067156 012700 077405      MOV      #77405,R0      ;RE-SET PAGES MODIFIED BY ERROR
9309 067162 010037 172320      MOV      R0,KDPDR0      ;RELOAD KERNEL PDR0
9310 067166 010037 172322      MOV      R0,KDPDR1      ;RELOAD KERNEL PDR1
9311 067172 010037 172336      MOV      R0,KDPDR7      ;RELOAD KERNEL PDR7
9312 067176 010037 172220      MOV      R0,SDPDR0      ;RELOAD PAGE 0 OF PRESENT SPACE
9313 067202 010037 172236      MOV      R0,SDPDR7      ;RE-LOAD I/O PAGE PDR IF ERROR
9314 067206 011515                MOV      (R5),(R5)      ;TRY WRITE AGAIN, IN CASE YOU
9315                                ;WERE TESTING PAGE SEVEN
9316 067210 000402                BR       12$           ;GO UPDATE R1 FOR NEXT READ
9317 067212 005237 001172      11$: INC     $TMP0        ;SET FLAG SINCE ADDRESSES MATCHED
9318 067216 062701 000002      12$: ADD     #2,R1        ;POINT TO NEXT PDR TO BE READ
9319 067222 077232                SOB      R2,10$        ;BRANCH TO 10$ IF ALL PDR'S NOT READ
9320 067224 012702 000020      MOV      #20,R2         ;SET COUNTER TO READ NEXT 20 REGISTERS
9321 067230 012701 177600      MOV      #UIPDR0,R1     ;LOAD ADDRESS OF BEGINNING PDR
9322 067234 011100                MOV      (R1),R0        ;READ PDR INTO R0
9323 067236 022700 077705      CMP      #77705,R0      ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
9324 067242 001023                BNE      15$           ;BRANCH IF THIS IS NOT THE ONE
9325 067244 020103                CMP      R1,R3          ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9326 067246 001417                BEQ      14$           ;BRANCH IF ADDRESS IS CORRECT
9327 067250 104112                ERROR    112           ;A & W BITS GOT SET IN WRONG PDR
9328 067252 012700 077405      MOV      #77405,R0      ;RE-SET PAGES MODIFIED BY ERROR
9329 067256 010037 172320      MOV      R0,KDPDR0      ;RELOAD KERNEL PDR0
9330 067262 010037 172322      MOV      R0,KDPDR1      ;RELOAD KERNEL PDR1

```

```

9331 067266 010037 172336      MOV      R0,KDPDR7      ;RELOAD KERNEL PDR7
9332 067272 010037 172220      MOV      R0,SDPDR0      ;RELOAD PAGE 0 OF PRESENT SPACE
9333 067276 010037 172236      MOV      R0,SDPDR7      ;RE-LOAD I/O PAGE PDR IF ERROR
9334 067302 011515                MOV      (R5),(R5)      ;TRY WRITE AGAIN, IN CASE YOU
9335                                ;WERE TESTING PAGE SEVEN
9336 067304 000402                BR       15$            ;GO UPDATE R1 FOR NEXT READ
9337 067306 005237 001172      14$: INC      $TMP0      ;SET FLAG SINCE ADDRESSES MATCHED
9338 067312 062701 000002      15$: ADD      #2,R1      ;POINT TO NEXT PDR TO BE READ
9339 067316 077232                SOB      R2,13$        ;BRANCH TO 13$ IF ALL PDR'S NOT READ
9340 067320 005737 001172      TST      $TMP0        ;SEE IF THERE WAS A CORRECT PDR
9341 067324 001002                BNE     16$            ;BRANCH IF THERE WAS
9342 067326 011300                MOV      (R3),R0       ;SAVE CONTENTS OF PDR UNDER TEST
9343 067330 104113                ERROR   113           ;NO PDR ADDRESSES MATCHED
9344 067332 062703 000002      16$: ADD      #2,R3      ;POINT TO NEXT PDR UNDER TEST
9345 067336 062705 020000      ADD      #20000,R5     ;CHANGE PAGE NUMBER IN VIRT. ADDR.
9346 067342 005304                DEC      R4            ;DECREMENT COUNTER
9347 067344 001402                BEQ     17$            ;BRANCH IF COUNTER IS ZERO
9348 067346 000137 066660      JMP      19$           ;JUMP TO LOAD PDR'S AGAIN
9349 067352                                ;
9350 067352 042737 000007 172516      BIC      #7,MMR3       ;DISABLE ALL D-SPACE MAPPING
9351 067360 012737 000340 177776      MOV      #340,PSW     ;RETURN TO KERNEL MODE, PRIORITY 7
9352 067366 012737 066622 001112      MOV      #20$,$LPERR  ;SET LOOP POINTER TO START OF TEST
9353
9354
9355 :*****
9356 :*TEST 105      DUAL MAPPING USER MODE D-SPACE
9357 :
9358 :       THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405
9359 :       (4K PAGE, TRAP ON WRITE). THEN THE VIRTUAL ADDRESS IS SET TO
9360 :       010200 AND THAT WORD IS WRITTEN INTO ITSELF. THIS WRITE WILL
9361 :       SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP
9362 :       SINCE BIT09 OF MPRO IS CLEAR) AND SET BOTH THE A & W BITS IN
9363 :       THE USER D-SPACE P.D.R. UNDER TEST. NOW ALL OF THE
9364 :       P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE
9365 :       ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
9366 :       WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
9367 :       ADDRESS OF 'MAPLOO' (17770200) WHICH SHOULD ALWAYS EXIST.
9368 :
9369 :*****
9370 :TST105:
9371 :       SCOPE
9372 :       MOV      #TST106,NXTTST ;SAVE STARTING ADDRESS OF NEXT
9373 :                                ;TEST FOR ESCAPE ON PARITY ERRORS
9374 :       20$: MOV      #21$,$LPERR ;SET LOOP ON ERROR POINTER TO 21$
9375 :       MOV      #140000,PSW    ;GO TO USER MODE
9376 :       BIS      #7,MMR3       ;ENABLE ALL D-SPACE MAPPING
9377 :       MOV      #UDPDR0,R3     ;LOAD FIRST ADDRESS OF USER PDR'S
9378 :                                ;THAT WILL BE TESTED IN THIS TEST
9379 :       MOV      #10,R4        ;TEST THE NEXT EIGHT PDR'S
9380 :       MOV      #10200,R5     ;LOAD STARTING VIRTUAL ADDRESS INTO R5
9381 :       19$: MOV      #77405,R0 ;ALL PAGES WILL BE TRAP ON WRITE
9382 :       CLR      $TMP0        ;CLEAR CORRECT PDR SET INDICATOR
9383 :       MOV      #10,R2        ;SET COUNT TO LOAD 8 ADDRESSES
9384 :       MOV      #SIPDR0,R1    ;PUT ADDRESS OF FIRST PDR IN R1
9385 :       1$: MOV      R0,(R1)+   ;LOAD R0 INTO PDR ADDRESSED BY R1
9386 :       SOB     R2,1$         ;BRANCH BACK TO 1$ IF R2 IS NOT ZERO
    
```

9387	067466	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
9388	067472	012701	172220		MOV	#SDPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
9389	067476	010021		2\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9390	067500	077202			SOB	R2,2\$:BRANCH BACK TO 2\$ IF R2 IS NOT ZERO
9391	067502	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
9392	067506	012701	172300		MOV	#KIPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
9393	067512	010021		3\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9394	067514	077202			SOB	R2,3\$:BRANCH BACK TO 3\$ IF R2 IS NOT ZERO
9395	067516	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
9396	067522	012701	172320		MOV	#KDPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
9397	067526	010021		4\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9398	067530	077202			SOB	R2,4\$:BRANCH BACK TO 4\$ IF R2 IS NOT ZERO
9399	067532	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
9400	067536	012701	177600		MOV	#UIPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
9401	067542	010021		5\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9402	067544	077202			SOB	R2,5\$:BRANCH BACK TO 5\$ IF R2 IS NOT ZERO
9403	067546	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
9404	067552	012701	177620		MOV	#UDPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
9405	067556	010021		6\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9406	067560	077202			SOB	R2,6\$:BRANCH BACK TO 6\$ IF R2 IS NOT ZERO
9407	067562	012700	077405	21\$:	MOV	#77405,R0	:MUST RE-INIT THESE PDRS IF ERROR
9408	067566	010037	172320		MOV	R0,KDPDR0	:LOAD KERNEL PDR0
9409	067572	010037	172322		MOV	R0,KDPDR1	:LOAD KERNEL PDR1
9410	067576	010037	172336		MOV	R0,KDPDR7	:LOAD KERNEL PDR7
9411	067602	010037	177620		MOV	R0,UDPDR0	:LOAD PDR0 OF PRESENT SPACE
9412	067606	010037	177636		MOV	R0,UDPDR7	:LOAD PDR7 OF PRESENT SPACE
9413	067612	000240			NOP		:THIS IS A SYNC POINT FOR SCGPING
9414	067614	011515			MOV	(R5),(R5)	:WRITE INTO PAGE UNDER TEST
9415	067616	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
9416	067622	012701	172300		MOV	#KIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
9417	067626	011100		7\$:	MOV	(R1),R0	:READ PDR INTO R0
9418	067630	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
9419	067634	001023			BNE	9\$:BRANCH IF THIS IS NOT THE ONE
9420	067636	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9421	067640	001417			BEQ	8\$:BRANCH IF ADDRESS IS CORRECT
9422	067642	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR
9423	067644	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
9424	067650	010037	172320		MOV	R0,KDPDR0	:RELOAD KERNEL PDR0
9425	067654	010037	172322		MOV	R0,KDPDR1	:RELOAD KERNEL PDR1
9426	067660	010037	172336		MOV	R0,KDPDR7	:RELOAD KERNEL PDR7
9427	067664	010037	177620		MOV	R0,UDPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
9428	067670	010037	177636		MOV	R0,UDPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
9429	067674	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
9430							:WERE TESTING PAGE SEVEN
9431	067676	000402			BR	9\$:GO UPDATE R1 FOR NEXT READ
9432	067700	005237	001172	8\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
9433	067704	062701	000002	9\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
9434	067710	077232			SOB	R2,7\$:BRANCH TO 7\$ IF ALL PDR'S NOT READ
9435	067712	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
9436	067716	012701	172200		MOV	#SIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
9437	067722	011100		10\$:	MOV	(R1),R0	:READ PDR INTO R0
9438	067724	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
9439	067730	001023			BNE	12\$:BRANCH IF THIS IS NOT THE ONE
9440	067732	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9441	067734	001417			BEQ	11\$:BRANCH IF ADDRESS IS CORRECT
9442	067736	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR

9443	067740	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
9444	067744	010037	172320		MOV	R0,KDPDR0	:RELOAD KERNEL PDR0
9445	067750	010037	172322		MOV	R0,KDPDR1	:RELOAD KERNEL PDR1
9446	067754	010037	172336		MOV	R0,KDPDR7	:RELOAD KERNEL PDR7
9447	067760	010037	177620		MOV	R0,UDPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
9448	067764	010037	177636		MOV	R0,UDPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
9449	067770	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
9450							:WERE TESTING PAGE SEVEN
9451	067772	000402			BR	12\$:GO UPDATE R1 FOR NEXT READ
9452	067774	005237	001172	11\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
9453	070000	062701	000002	12\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
9454	070004	077232			SOB	R2,10\$:BRANCH TO 10\$ IF ALL PDR'S NOT READ
9455	070006	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
9456	070012	012701	177600		MOV	#UIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
9457	070016	011100		13\$:	MOV	(R1),R0	:READ PDR INTO R0
9458	070020	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH ABW BITS ON
9459	070024	001023			BNE	15\$:BRANCH IF THIS IS NOT THE ONE
9460	070026	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9461	070030	001417			BEQ	14\$:BRANCH IF ADDRESS IS CORRECT
9462	070032	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR
9463	070034	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
9464	070040	010037	172320		MOV	R0,KDPDR0	:RELOAD KERNEL PDR0
9465	070044	010037	172322		MOV	R0,KDPDR1	:RELOAD KERNEL PDR1
9466	070050	010037	172336		MOV	R0,KDPDR7	:RELOAD KERNEL PDR7
9467	070054	010037	177620		MOV	R0,UDPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
9468	070060	010037	177636		MOV	R0,UDPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
9469	070064	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
9470							:WERE TESTING PAGE SEVEN
9471	070066	000402			BR	15\$:GO UPDATE R1 FOR NEXT READ
9472	070070	005237	001172	14\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
9473	070074	062701	000002	15\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
9474	070100	077232			SOB	R2,13\$:BRANCH TO 13\$ IF ALL PDR'S NOT READ
9475	070102	005737	001172		TST	\$TMP0	:SEE IF THERE WAS A CORRECT PDR
9476	070106	001002			BNE	16\$:BRANCH IF THERE WAS
9477	070110	011300			MOV	(R3),R0	:SAVE CONTENTS OF PDR UNDER TEST
9478	070112	104113			ERROR	113	:NO PDR ADDRESSES MATCHED
9479	070114	062703	000002	16\$:	ADD	#2,R3	:POINT TO NEXT PDR UNDER TEST
9480	070120	062705	020000		ADD	#20000,R5	:CHANGE PAGE NUMBER IN VIRT. ADDR.
9481	070124	005304			DEC	R4	:DECREMENT COUNTER
9482	070126	001402			BEQ	17\$:BRANCH IF COUNTER IS ZERO
9483	070130	000137	067442		JMP	19\$:JUMP TO LOAD PDR'S AGAIN
9484	070134			17\$:			
9485	070134	042737	000007	172516	BIC	#7,MPR3	:DISABLE ALL D-SPACE MAPPING
9486	070142	012737	000340	177776	MOV	#340,PSW	:RETURN TO KERNEL MODE, PRIORITY 7
9487	070150	012737	067404	001112	MOV	#20\$,SLPERR	:SET LOOP POINTER TO START OF TEST
9488							
9489							
9490					.SBTTL	***** ENTRY POINT 8 --- STARTING ADDRESS 234 *****	
9491					.SBTTL	***** MOVE FROM AND MOVE TO PREVIOUS MODE INSTRUCTION TEST *****	
9492					*		
9493					*		
9494					*		
9495					*		
9496					*		
9497					*		
9498					*		

***** MOVE FROM AND MOVE TO PREVIOUS MODE INSTRUCTION TEST *****

SEQ 0183

9499
 9500
 9501
 9502
 9503
 9504
 9505
 9506
 9507
 9508
 9509
 9510
 9511
 9512 070156
 9513 070156 000004
 9514 070160 012737 070760 001316
 9515
 9516 070166 104420
 9517
 9518 070170
 9519 070170 012737 070372 001110
 9520 070176 012737 070372 001112
 9521 070204 012737 000106 001102
 9522 070212 013737 001102 177570
 9523 070220 012737 077406 172300
 9524 070226 012737 077406 172302
 9525 070234 012737 077406 172304
 9526 070242 012737 077406 172306
 9527 070250 012737 077406 172316
 9528 070256 012737 000000 172340
 9529 070264 012737 000200 172342
 9530 070272 012737 000400 172344
 9531 070300 012737 000600 172346
 9532 070306 012737 177600 172356
 9533 070314 012737 000001 177572
 9534 070322 012737 000020 172516
 9535 070330 042737 000007 172516
 9536 070336 012700 077406
 9537
 9538 070342 012702 000010
 9539 070346 012701 172300
 9540 070352 010021
 9541 070354 077202
 9542 070356 012737 070372 001110
 9543 070364 012737 070372 001112
 9544 070372 012700 077400
 9545
 9546 070376 010037 172330
 9547 070402 010037 172230
 9548 070406 010037 177630
 9549 070412 010037 177610
 9550 070416 012737 077406 172310
 9551 070424 012737 077406 172210
 9552 070432 012737 001000 172350
 9553 070440 012737 001000 172250
 9554 070446 012700 036514

```

*****
*TEST 106      MOVE FROM PREVIOUS (SUPERVISOR) I-SPACE
*
*      THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE
*      PREVIOUS MODE IS CLOKED CORRECTLY BY 'ROM OUT05'.
*      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
*      WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
*      THEIR ADDRESSES).
*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
*      WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
*****
    
```

```

TST106:
SCOPE
MOV      #TST107,NXTTST  :SAVE STARTING ADDRESS OF NEXT
:TEST FOR ESCAPE ON PARITY ERRORS
TBITR    :RESTORE T-BIT TO ITS STATUS BEFORE
:THE SIX DUAL MAPPING TESTS

ENTPT8:
MOV      #20$,SLPADR     :SET LOOP ADDRESS POINTER TO 20$
MOV      #20$,SLPERR     :SET LOOP ON ERROR POINTER TO 20$
MOV      #106,$TSTNM     :LOAD TEST NUMBER INTO MEMORY
MOV      $TSTNM,DISPLAY  :DISPLAY TEST NUMBER FOR THIS TEST
MOV      #77406,KIPDR0   :MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
MOV      #77406,KIPDR1   :MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
MOV      #77406,KIPDR2   :MAKE KERNEL I PAGE 2 200 BLCCKS, R/W
MOV      #77406,KIPDR3   :MAKE KERNEL I PAGE 3 200 BLOC:KS, R/W
MOV      #77406,KIPDR7   :MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
MOV      #000,KIPAR0     :MAP KERNEL I PAGE 0 TO 0 - 4K
MOV      #200,KIPAR1     :MAP KERNEL I PAGE 1 TO 4K - 8K
MOV      #400,KIPAR2     :MAP KERNEL I PAGE 2 TO 8K - 12K
MOV      #600,KIPAR3     :MAP KERNEL I PAGE 3 TO 12K - 16K
MOV      #177600,KIPAR7  :MAP KERNEL I PAGE 7 TO THE I/O PAGE
MOV      #BIT0,MPR0      :ENABLE 18-BIT RELOCATION IF NOT ON
MOV      #BIT4,MPR3      :ENABLE 22-BIT RELOCATION IF NOT ON
BIC      #7,MPR3         :MAKE SURE THAT ALL D-SPACE IS DISABLED
MOV      #77406,R0       :MAKE ALL KERNEL I-SPACE PAGES RESIDENT
:READ/WRITE, LENGTH 200 BLOCKS
MOV      #10,R2          :SET COUNT TO LOAD 8 ADDRESSES
MOV      #KIPDR0,R1      :PUT ADDRESS OF FIRST PDR IN R1
19$:    MOV      R0,(R1)+  :LOAD R0 INTO PDR ADDRESSED BY R1
:BRANCH BACK TO 19$ IF R2 IS NOT ZERO
SOB      R2,19$
MOV      #20$,SLPADR     :SET LOOP POINTER TO 20$
MOV      #20$,SLPERR     :SET LOOP ON ERROR TO 20$
20$:    MOV      #77400,R0  :MAKE PAGE 4 IN ALL BUT SUPERVISOR I
:AND KERNEL I NON-RESIDENT
MOV      R0,KDPDR4       :KERNEL D-SPACE PAGE 4
MOV      R0,SDPDR4       :SUPERVISOR D-SPACE PAGE 4
MOV      R0,UDPDR4       :USER D-SPACE PAGE 4
MOV      R0,UIPDR4       :USER I-SPACE PAGE 4
MOV      #77406,KIPDR4   :KERNEL I-SPACE PAGE 4 READ/WRITE
MOV      #77406,SIPDR4   :SUPER I-SPACE PAGE 4 READ/WRITE
MOV      #1000,KIPAR4    :MAP KERNEL I PAGE 4 TO 16K
MOV      #1000,SIPAR4    :MAP SUPER I PAGE 4 TO 16K
MOV      #36514,R0       :LOAD DATA PATTERN INTO R0
    
```

```

9555 070452 010037 100000      MOV      R0,#100000      ;LOAD DATA PATTERN INTO PHY 100000
9556 070456 012737 070732 000250  MOV      #10$,MVEEC      ;SET M.M. VECTOR TO 10$
9557 070464 105037 172310      CLR      KIPDR4          ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
9558 070470 012737 070476 001112  MOV      #11$,SLPERR      ;SET LOOP ON ERROR POINTER TO 11$
9559 070476 012737 010340 177776 11$:  MOV      #010340,PSW      ;MAKE PREVIOUS MODE SUPERVISOR
9560 070504 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
9561 070506 006506      MFPI     SSP             ;PUT SUPERVISOR STACK POINTER ON KERNEL
9562                                ;STACK
9563 070510 022706 001100      CMP      #KERSTK,KSP      ;WAS SOMETHING PUSHED ON STACK AT 1$
9564 070514 001407      BEQ     3$              ;BRANCH IF NOTHING WAS PUSHED
9565 070516 012601      MOV      (KSP)+,R1        ;POP KERNEL STACK INTO R1
9566 070520 012702 000700      MOV      #SUPSTK,R2       ;EXPECTING TO GET 700 AS SSP
9567 070524 020201      CMP      R2,R1            ;DID YOU GET THE RIGHT POINTER?
9568 070526 001403      BEQ     2$              ;BRANCH IF YOU DID
9569 070530 104114      ERROR   114             ;WRONG THING WAS PUSHED ON STACK
9570 070532 000401      BR      2$              ;BRANCH TO NEXT TRY
9571 070534 104115      3$:  ERROR   115             ;NOTHING PUSHED ON STACK
9572 070536      2$:  ;THE FOLLOWING WILL TEST DSTM=1 MFPI. BELOW ARE THE
          ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
          ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
          ;INTO THE F/F'S ON SSRB.
          ; * D12.00 (001)
          ; * D12.10 (175)
          ; * MFP.00 (066)
          ; * MFP.10 (250)
          ; * SVC.80 (222)
          ; * SVC.90 (300)
          ; * FET.00 (217)
9583 070536 012737 070544 001112  MOV      #12$,SLPERR      ;SET LOOP ON ERROR POINTER TO 12$
9584 070544 012737 010340 177776 12$:  MOV      #010340,PSW      ;MAKE PREVIOUS MODE SUPERVISOR
9585 070552 012702 100000      MOV      #100000,R2       ;LOAD VIRTUAL ADDRESS INTO R2
9586 070556 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
9587 070560 006512      MFPI     (R2)           ;READ FROM PHYSICAL 100000
9588 070562 012601      MOV      (KSP)+,R1        ;POP KERNEL STACK INTO R1
9589 070564 020001      CMP      R0,R1            ;WAS DATA FETCHED SAME AS STORED
9590 070566 001401      BEQ     4$              ;BRANCH IF CORRECT DATA WAS FETCHED
9591 070570 104116      ERROR   116             ;WRONG DATA WAS FETCHED
9592 070572      4$:  ;THE FOLLOWING WILL TEST DSTM=2 MFPI. BELOW ARE THE
          ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
          ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
          ;INTO THE F/F'S ON SSRB.
          ; * D12.01 (002)
          ; * D12.10 (175)
          ; * MFP.00 (066)
          ; * MFP.10 (250)
          ; * SVC.80 (222)
          ; * SVC.90 (300)
          ; * FET.00 (217)
9603 070572 012737 070600 001112  MOV      #14$,SLPERR      ;SET LOOP ON ERROR POINTER TO 14$
9604 070600 012737 010340 177776 14$:  MOV      #010340,PSW      ;MAKE PREVIOUS MODE SUPERVISOR
9605 070606 012702 100000      MOV      #100000,R2       ;LOAD VIRTUAL ADDRESS INTO R2
9606 070612 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
9607 070614 006522      MFPI     (R2)+          ;READ FROM PHYSICAL 100000
9608 070616 012601      MOV      (KSP)+,R1        ;POP KERNEL STACK INTO R1
9609 070620 020001      CMP      R0,R1            ;WAS DATA FETCHED SAME AS STORED
9610 070622 001401      BEQ     5$              ;BRANCH IF CORRECT DATA WAS FETCHED

```

```

9611 070624 104116
9612 070626
9613
9614
9615
9616
9617
9618
9619
9620
9621
9622
9623
9624
9625
9626 070626 012737 070634 001112
9627 070634 012737 010340 177776 15$:
9628 070642 000240
9629 070644 006537 100000
9630 070650 012601
9631 070652 020001
9632 070654 001401
9633 070656 104116
9634 070660
9635
9636
9637
9638
9639
9640
9641
9642
9643
9644
9645
9646 070660 012737 070666 001112
9647 070666 012737 010340 177776 16$:
9648 070674 012702 100002
9649 070700 000240
9650 070702 006542
9651 070704 012601
9652 070706 020001
9653 070710 001401
9654 070712 104116
9655 070714 012737 032226 000250 7$:
9656 070722 012737 070372 001112
9657 070730 000413
9658
9659
9660 070732 013737 177572 001250 10$:
9661 070740 013737 177574 001252
9662 070746 013737 177576 001254
9663 070754 104117
9664 070756 000002
9665
9666

ERROR 116 ;WRONG DATA WAS FETCHED
5$: ;THE FOLLOWING WILL TEST DSTM=3 MFPI. BELOW ARE THE
;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
;INTO THE F/F'S ON SSRB.
; D30.00 (003)
; D30.10 (221)
; D10.20 (233)
; * D10.50 (311)
; * D10.60 (177)
; * MFP.00 (066)
; MFP.10 (250)
; SVC.80 (222)
; SVC.90 (300)
; FET.00 (217)
MOV #15$,SLPERR ;SET LOOP ON ERROR POINTER TO 15$
MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
NOP ;THIS IS A SYNC POINT FOR SCOPING
MFPI @#100000 ;READ FROM PHYSICAL 100000
MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
BEQ 6$ ;BRANCH IF CORRECT DATA WAS FETCHED
ERROR 116 ;WRONG DATA WAS FETCHED
6$: ;THE FOLLOWING WILL TEST DSTM=4 MFPI. BELOW ARE THE
;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
;INTO THE F/F'S ON SSRB.
; D45.00 (004)
; * D10.30 (122)
; * D10.60 (177)
; * MFP.00 (066)
; MFP.10 (250)
; SVC.80 (222)
; SVC.90 (300)
; FET.00 (217)
MOV #16$,SLPERR ;SET LOOP ON ERROR POINTER TO 16$
MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
MOV #100002,R2 ;LOAD VIRTUAL ADDRESS INTO R2
NOP ;THIS IS A SYNC POINT FOR SCOPING
MFPI -(R2) ;READ FROM PHYSICAL 100000
MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
BEQ 7$ ;BRANCH IF CORRECT DATA WAS FETCHED
ERROR 116 ;WRONG DATA WAS FETCHED
MOV #MMTRAP,MMVEC ;SET M.M.VECTOR TO NORMAL ROUTINE
MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
BR TST107 ;BRANCH TO NEXT TEST

MOV MMRO,MMRO ;SAVE MMRO FOR ERROR TYPEOUT
MOV MMR1,MMR1 ;SAVE MMR1 FOR ERROR TYPEOUT
MOV MMR2,MMR2 ;SAVE MMR2 FOR ERROR TYPEOUT
ERROR 117 ;TRIED TO READ NON-RESIDENT PAGE
RTI
    
```

9667
9668
9669
9670
9671
9672
9673
9674
9675
9676
9677
9678
9679
9680
9681
9682
9683
9684
9685
9686
9687
9688
9689
9690
9691
9692
9693
9694
9695
9696
9697
9698
9699
9700
9701
9702
9703
9704
9705
9706
9707
9708
9709
9710
9711
9712
9713
9714
9715
9716
9717
9718
9719
9720
9721
9722

070760
070760 000004
070762 012737 071440 001316
070770 012700 077400
070774 010037 172330
071000 010037 172230
071004 010037 177630
071010 010037 177610
071014 012737 077406 172310
071022 012737 077406 172210
071030 012737 001000 172350
071036 012737 001000 172250
071044 012737 071412 000250
071052 012737 010340 177776
071060 012746 007777
071064 006606
071066 006506
071070 012601
071072 022701 007777
071076 001401
071100 104120
071102 012737 010340 177776
071110 012746 000700
071114 006606
071116
071116 012737 071134 001112
071124 012702 100000
071130 012700 125252
071134 010046
071136 105037 172310
071142 000240
071144 006612
071146 112737 000006 172310
071154 011201
071156 020001

```
*****
*TEST 107 MOVE TO PREVIOUS (SUPERVISOR) I-SPACE
*
* THIS TEST USES THE 'MTP1' INSTRUCTION TO ENSURE THAT THE
* PREVIOUS MODE IS CLOKED CORRECTLY BY 'ROM OUT05'.
* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
* WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
* THEIR ADDRESSES).
* IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
* WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
*****
TST107:
SCOPE
MOV #TST110,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
20$: MOV #77400,R0 ;MAKE PAGE 4 IN ALL BUT SUPERVISOR I
;AND KERNEL I NON-RESIDENT
MOV R0,KDPDR4 ;KERNEL D-SPACE PAGE 4
MOV R0,SDPDR4 ;SUPERVISOR D-SPACE PAGE 4
MOV R0,UDPDR4 ;USER D-SPACE PAGE 4
MOV R0,U1PDR4 ;USER I-SPACE PAGE 4
MOV #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
MOV #77406,SIPDR4 ;SUPER I-SPACE PAGE 4 READ/WRITE
MOV #1000,KIPAR4 ;MAP KERNEL I PAGE 4 TO 16K
MOV #1000,SIPAR4 ;MAP SUPER I PAGE 4 TO 16K
MOV #10$,MVEC ;SET M.M. VECTOR TO 10$
1$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
MOV #7777,-(KSP) ;PUSH DATA ON KERNEL STACK
MTP1 SSP ;LOAD SUPERVISOR STACK POINTER
MFPI SSP ;READ SUPERVISOR STACK POINTER
MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
CMP #7777,R1 ;WAS SUPERVISOR STACK POINTER CHANGED
BEQ 2$ ;BRANCH IF IT WAS
ERROR 120 ;SUPER STACK POINTER NOT CHANGED
2$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
MOV #SUPSTK,-(KSP) ;GET READY TO RESTORE SUPERVISOR S. POINT
MTP1 SSP ;RESTORE SUPERVISOR STACK POINTER
3$: ;THIS WILL TEST DSTH = 1 MTP1. BELOW ARE THE
;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
;INTO THE FLIP/FLOPS ON SSRB.
; * D12.80 (111)
; * D12.60 (155)
; D12.20 (312)
; FET.10 (260)
MOV #13$,SLPERR ;SET LOOP ON ERROR POINTER TO 13$
MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
MOV #125252,R0 ;LOAD TEST DATA INTO R0
13$: MOV R0,-(KSP) ;PUSH TEST DATA ON KERNEL STACK
CLRB KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
NOP ;THIS IS A SYNC POINT FOR SCOPING
MTP1 (R2) ;LOAD TEST DATA INTO PHYSICAL 100000
MOVB #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
MOV (R2),R1 ;READ FROM ADDRESS 100000
CMP R0,R1 ;SEE IF DATA WAS STORED AT CORRECT PLACE
```

```

9723 071160 001401      BEQ      4$      ;BRANCH IF STORE WAS CORRECT
9724 071162 104121      ERROR    121     ;INCORRECT STORE
9725 071164      4$:  ;THIS WILL TEST DSTM = 3 MTP1. BELOW ARE THE
9726      ;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
9727      ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
9728      ;INTO THE FLIP/FLOPS ON SSRB.
9729      :      D30.80   (113)
9730      :      D30.10   (221)
9731      :      D10.20   (233)
9732      :      * D10.50   (311)
9733      :      * D10.40   (157)
9734      :      FET.01   (331)
9735 071164 012737 071204 001112      MOV      #14$, $LPERR ;SET LOOP ON ERROR POINTER TO 14$
9736 071172 012737 010340 177776      MOV      #010340, PSW ;MAKE PREVIOUS MODE SUPERVISOR
9737 071200 012700 052525      MOV      #52525, R0   ;LOAD TEST DATA INTO R0
9738 071204 010046      14$:  MOV      R0, -(KSP)  ;PUSH TEST DATA ON KERNEL STACK
9739 071206 105037 172310      CLRB     KIPDR4      ;MAKE KERNEL I PAGE 4 NON-RESIDENT
9740 071212 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
9741 071214 006637 100000      MTP1     @#100000     ;LOAD TEST DATA INTO PHYSICAL 100000
9742 071220 112737 000006 172310      MOV     #006, KIPDR4  ;MAKE KERNEL PAGE 4 RESIDENT
9743 071226 013701 100000      MOV     @#100000, R1  ;READ FROM ADDRESS 100000
9744 071232 020001      CMP     R0, R1        ;SEE IF DATA WAS STORED CORRECTLY
9745 071234 001401      BEQ     5$           ;BRANCH IF STORE WAS CORRECT
9746 071236 104121      ERROR    121     ;INCORRECT STORE
9747 071240      5$:  ;THIS WILL TEST DSTM = 4 MTP1. BELOW ARE THE
9748      ;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
9749      ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
9750      ;INTO THE FLIP/FLOPS ON SSRB.
9751      :      D45.80   (115)
9752      :      * D40.20   (121)
9753      :      * D10.40   (157)
9754      :      FET.01   (331)
9755 071240 012737 071260 001112      MOV      #15$, $LPERR ;SET LOOP ON ERROR POINTER TO 15$
9756 071246 012737 010340 177776      MOV      #010340, PSW ;MAKE PREVIOUS MODE SUPERVISOR
9757 071254 012700 125252      MOV      #125252, R0  ;LOAD TEST DATA INTO R0
9758 071260 010046      15$:  MOV      R0, -(KSP)  ;PUSH TEST DATA ON KERNEL STACK
9759 071262 012702 100002      MOV      #100002, R2  ;LOAD VIRTUAL ADDRESS INTO R2
9760 071266 105037 172310      CLRB     KIPDR4      ;MAKE KERNEL I PAGE 4 NON-RESIDENT
9761 071272 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
9762 071274 006642      MTP1     -(R2)        ;LOAD TEST DATA INTO PHYSICAL 100000
9763 071276 112737 000006 172310      MOV     #006, KIPDR4  ;MAKE KERNEL PAGE 4 RESIDENT
9764 071304 013701 100000      MOV     @#100000, R1  ;READ FROM ADDRESS 100000
9765 071310 020001      CMP     R0, R1        ;SEE IF DATA WAS STORED CORRECTLY
9766 071312 001401      BEQ     6$           ;BRANCH IF STORE WAS CORRECT
9767 071314 104121      ERROR    121     ;INCORRECT STORE
9768 071316      6$:  ;THIS WILL TEST DSTM = 6 MTP1. BELOW ARE THE
9769      ;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
9770      ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
9771      ;INTO THE FLIP/FLOPS ON SSRB.
9772      :      D67.80   (117)
9773      :      D67.00   (006)
9774      :      D67.10   (251)
9775      :      * D10.30   (122)
9776      :      * D10.40   (157)
9777      :      FET.01   (331)
9778 071316 012737 071340 001112      MOV      #16$, $LPERR ;SET LOOP ON ERROR POINTER TO 16$
    
```

```

9779 071324 012737 010340 177776      MOV      #010340,PSW      ;MAKE PREVIOUS MODE SUPERVISOR
9780 071332 012700 052525      MOV      #52525,R0       ;LOAD TEST DATA INTO R0
9781 071336 005002      CLR      R2              ;MAKE REGISTER 2 ZERO
9782 071340 010046      16$:    MOV      R0,-(KSP)    ;PUSH TEST DATA ON KERNEL STACK
9783 071342 105037 172310      CLR8    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
9784 071346 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
9785 071350 006662 100000      MTP1    100000(R2)      ;LOAD TEST DATA INTO PHYSICAL 100000
9786 071354 112737 000006 172310      MOV8    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
9787 071362 013701 100000      MOV      @#100000,R1     ;READ FROM ADDRESS 100000
9788 071366 020001      CMP      R0,R1          ;SEE IF DATA WAS STORED CORRECTLY
9789 071370 001401      BEQ     7$              ;BRANCH IF STORE WAS CORRECT
9790 071372 104121      ERROR   121             ;INCORRECT STORE
9791 071374 012737 070770 001112 7$:    MOV      #20$,SLPERR     ;SET LOOP POINTER TO START OF TEST
9792 071402 012737 032226 000250      MOV      #MMTRAP,MMVEC   ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
9793 071410 000413      BR      TST110          ;:BRANCH TO NEXT TEST
9794
9795
9796 071412 013737 177572 001250 10$:    MOV      MMR0,MMR0       ;SAVE MMR0 FOR ERROR TYPEOUT
9797 071420 013737 177574 001252      MOV      MMR1,MMR1       ;SAVE MMR1 FOR ERROR TYPEOUT
9798 071426 013737 177576 001254      MOV      MMR2,MMR2       ;SAVE MMR2 FOR ERROR TYPEOUT
9799 071434 104117      ERROR   117             ;TRIED TO LOAD A N.R. PAGE 4
9800 071436 000002      RTI                      ;RETURN TO TEST

```

```

9801
9802
9803 *****
9804 *TEST 110      MFPI (SUPERVISOR) WITH SUPERVISOR D-SPACE ENABLED
9805 *
9806 *      THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE
9807 *      PREVIOUS MODE IS CLOKED CORRECTLY BY 'ROM OUT05' AND THAT
9808 *      D-SPACE IS NOT ENABLED. THIS IS DONE BY 'ROM OUT08 H' AND
9809 *      'SSRB IR15 L' NOT ASSERTED GENERATING 'SSRB I SPACEB L'.
9810 *      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
9811 *      WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
9812 *      THEIR ADDRESSES).
9813 *      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
9814 *      WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
9815 *
9816 *****

```

```

9817 071440      TST110:
9818 071440 000004      SCOPE
9819 071442 012737 072040 001316      MOV      #TST111,NXTTST ;SAVE STARTING ADDRESS OF NEXT
9820      MOV      #77406,R0    ;TEST FOR ESCAPE ON PARITY ERRORS
9821 071450 012700 077406      MOV      #77406,R0      ;MAKE ALL KERNEL I-SPACE PAGES RESIDENT
9822      MOV      #10,R2      ;READ/WRITE, LENGTH 200 BLOCKS
9823 071454 012702 000010      MOV      #KIPDR0,R1     ;SET COUNT TO LOAD 8 ADDRESSES
9824 071460 012701 172300      MOV      R0,(R1)+       ;PUT ADDRESS OF FIRST PDR IN R1
9825 071464 010021      19$:    MOV      R2,19$         ;LOAD R0 INTO PDR ADDRESSED BY R1
9826 071466 077202      SOB     R2,19$          ;BRANCH BACK TO 19$ IF R2 IS NOT ZERO
9827 071470 012737 071504 001110      MOV      #20$,SLPADR    ;SET LOOP POINTER TO 20$
9828 071476 012737 071504 001112      MOV      #20$,SLPERR    ;SET LOOP ON ERROR TO 20$
9829 071504 012700 077400      20$:    MOV      #77400,R0      ;MAKE PAGE 4 IN ALL BUT SUPERVISOR I
9830      MOV      R0,KDPDR4   ;AND KERNEL I NON-RESIDENT
9831 071510 010037 172330      MOV      R0,SDPDR4      ;KERNEL D-SPACE PAGE 4
9832 071514 010037 172230      MOV      R0,UDPDR4      ;SUPERVISOR D-SPACE PAGE 4
9833 071520 010037 177630      MOV      R0,UIPDR4      ;USER D-SPACE PAGE 4
9834 071524 010037 177610      MOV      R0,UIPDR4      ;USER I-SPACE PAGE 4

```

```
9835 071530 012737 077406 172310 MOV #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
9836 071536 012737 077406 172210 MOV #77406,SIPDR4 ;SUPER I-SPACE PAGE 4 READ/WRITE
9837 071544 012737 001000 172350 MOV #1000,KIPAR4 ;MAP KERNEL I PAGE 4 TO 16K
9838 071552 012737 001000 172250 MOV #1000,SIPAR4 ;MAP SUPER I PAGE 4 TO 16K
9839 071560 012700 036514 MOV #36514,R0 ;LOAD DATA PATTERN INTO R0
9840 071564 010037 100000 MOV R0,#100000 ;LOAD DATA PATTERN INTO PHY 100000
9841 071570 012737 072012 000250 MOV #10$,MMVEC ;SET M.M. VECTOR TO 10$
9842 071576 052737 000002 172516 BIS #BIT1,MPR3 ;ENABLE SUPERVISOR D-SPACE
9843 071604 105037 172310 CLR# KIPDR4 ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
9844 ;THE FOLLOWING WILL TEST DSTM=1 MFPI. BELOW ARE THE
9845 ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
9846 ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
9847 ;INTO THE F/F'S ON SSRB.
9848 ; * D12.00 (001)
9849 ; * D12.10 (175)
9850 ; * MFP.00 (066)
9851 ; MFP.10 (250)
9852 ; SVC.80 (222)
9853 ; SVC.90 (300)
9854 ; FET.00 (217)
9855 071610 012737 071616 001112 MOV #12$,SLPERR ;SET LOOP ON ERROR POINTER TO 12$
9856 071616 012737 010340 177776 12$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
9857 071624 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
9858 071630 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
9859 071632 006512 MFPI (R2) ;READ FROM PHYSICAL 100000
9860 071634 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
9861 071636 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
9862 071640 001401 BEQ 4$ ;BRANCH IF CORRECT DATA WAS FETCHED
9863 071642 104116 ERROR 116 ;WRONG DATA WAS FETCHED
9864 071644 4$: ;THE FOLLOWING WILL TEST DSTM=2 MFPI. BELOW ARE THE
9865 ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
9866 ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
9867 ;INTO THE F/F'S ON SSRB.
9868 ; * D12.01 (002)
9869 ; * D12.10 (175)
9870 ; * MFP.00 (066)
9871 ; MFP.10 (250)
9872 ; SVC.80 (222)
9873 ; SVC.90 (300)
9874 ; FET.00 (217)
9875 071644 012737 071652 001112 MOV #14$,SLPERR ;SET LOOP ON ERROR POINTER TO 14$
9876 071652 012737 010340 177776 14$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
9877 071660 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
9878 071664 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
9879 071666 006522 MFPI (R2)+ ;READ FROM PHYSICAL 100000
9880 071670 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
9881 071672 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
9882 071674 001401 BEQ 5$ ;BRANCH IF CORRECT DATA WAS FETCHED
9883 071676 104116 ERROR 116 ;WRONG DATA WAS FETCHED
9884 071700 5$: ;THE FOLLOWING WILL TEST DSTM=3 MFPI. BELOW ARE THE
9885 ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
9886 ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
9887 ;INTO THE F/F'S ON SSRB.
9888 ; * D30.00 (003)
9889 ; * D30.10 (221)
9890 ; * D10.20 (233)
```

```

9891      : * D10.50      (311)
9892      : * D10.60      (177)
9893      : * MFP.00      (066)
9894      : * MFP.10      (250)
9895      : * SVC.80      (222)
9896      : * SVC.90      (300)
9897      : * FET.00      (217)
9898 071700 012737 071706 001112      MOV      #158,SLPERR      ;SET LOOP ON ERROR POINTER TO 158
9899 071706 012737 010340 177776 158:  MOV      #010340,PSW      ;MAKE PREVIOUS MODE SUPERVISOR
9900 071714 000240      NOP      ;THIS IS A SYNC POINT FOR SCOPING
9901 071716 006537 100000      MFPI     @#100000      ;READ FROM PHYSICAL 100000
9902 071722 012601      MOV      (KSP)+,R1      ;POP KERNEL STACK INTO R1
9903 071724 020001      CMP      R0,R1          ;WAS DATA FETCHED SAME AS STORED
9904 071726 001401      BEQ      68             ;BRANCH IF CORRECT DATA WAS FETCHED
9905 071730 104116      ERROR    116          ;WRONG DATA WAS FETCHED
9906 071732      68:      ;THE FOLLOWING WILL TEST DSTN=4 MFPI. BELOW ARE THE
9907      ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
9908      ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
9909      ;INTO THE F/F'S ON SSRB.
9910      : * D45.00      (004)
9911      : * D10.30      (122)
9912      : * D10.60      (177)
9913      : * MFP.00      (066)
9914      : * MFP.10      (250)
9915      : * SVC.80      (222)
9916      : * SVC.90      (300)
9917      : * FET.00      (217)
9918 071732 012737 071740 001112      MOV      #168,SLPERR      ;SET LOOP ON ERROR POINTER TO 168
9919 071740 012737 010340 177776 168:  MOV      #010340,PSW      ;MAKE PREVIOUS MODE SUPERVISOR
9920 071746 012702 100002      MOV      #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
9921 071752 000240      NOP      ;THIS IS A SYNC POINT FOR SCOPING
9922 071754 006542      MFPI     -(R2)          ;READ FROM PHYSICAL 100000
9923 071756 012601      MOV      (KSP)+,R1      ;POP KERNEL STACK INTO R1
9924 071760 020001      CMP      R0,R1          ;WAS DATA FETCHED SAME AS STORED
9925 071762 001401      BEQ      78             ;BRANCH IF CORRECT DATA WAS FETCHED
9926 071764 104116      ERROR    116          ;WRONG DATA WAS FETCHED
9927 071766 012737 032226 000250 78:  MOV      #MMTRAP,MMVEC    ;SET M.M.VECTOR TO NORMAL ROUTINE
9928 071774 012737 071504 001112      MOV      #208,SLPERR      ;SET LOOP POINTER TO START OF TEST
9929 072002 042737 000002 172516      BIC      #BIT1,MMR3      ;DISABLE SUPERVISOR D-SPACE
9930 072010 000413      BR       TST111        ;BRANCH TO NEXT TEST
9931
9932
9933 072012 013737 177572 001250 108:  MOV      MMR0,PMR0      ;SAVE MMR0 FOR ERROR TYPEOUT
9934 072020 013737 177574 001252      MOV      MMR1,PMR1      ;SAVE MMR1 FOR ERROR TYPEOUT
9935 072026 013737 177576 001254      MOV      MMR2,PMR2      ;SAVE MMR2 FOR ERROR TYPEOUT
9936 072034 104117      ERROR    117          ;TRIED TO READ NON-RESIDENT PAGE
9937 072036 000002      RTI      ;RETURN TO TEST
9938
9939
9940
9941
9942
9943
9944
9945
9946
    
```

```

*****
*TEST 111      MTP1 (SUPERVISOR) WITH SUPERVISOR D-SPACE ENABLED
*
*      THIS TEST USES THE 'MTP1' INSTRUCTION TO ENSURE(TIQT(TIE
*      PREVIOUS MODE IS CLOKED CORRECTLY BY 'ROM OUT05', AND THAT
    
```



```

9947      : * D-SPACE IS NOT ENABLED. THIS IS DONE BY 'ROM OUT08 H' AND
9948      : * 'SSRB IR15 L' NOT ASSERTED GENERATING 'SSRB I SPACEB L'.
9949      : * THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.
9950      : * WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
9951      : * THEIR ADDRESSES).
9952      : * IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
9953      : * WILL OCCUR AND T'OP TO 10$, WHERE THE ERRORS ARE REPORTED.
9954      : *
9955      : *****
9956 072040  : TST111:
9957 072040 000004      SCOPE
9958 072042 012737 072470 001316      MOV      #TST112,NXTTST      :SAVE STARTING ADDRESS OF NEXT
9959      :TEST FOR ESCAPE ON PARITY ERRORS
9960 072050 012700 077400      20$: MOV      #77400,R0      :MAKE PAGE 4 IN ALL BUT SUPERVISOR I
9961      :AND KERNEL I NON-RESIDENT
9962 072054 010037 172330      MOV      R0,KDPDR4      :KERNEL D-SPACE PAGE 4
9963 072060 010037 172230      MOV      R0,SDPDR4      :SUPERVISOR D-SPACE PAGE 4
9964 072064 010037 177630      MOV      R0,UDPDR4      :USER D-SPACE PAGE 4
9965 072070 010037 177610      MOV      R0,UIPDR4      :USER I-SPACE PAGE 4
9966 072074 012737 077406 172310      MOV      #77406,KIPDR4      :KERNEL I-SPACE PAGE 4 READ/WRITE
9967 072102 012737 077406 172210      MOV      #77406,SIPDR4      :SUPER I-SPACE PAGE 4 READ/WRITE
9968 072110 012737 001000 172350      MOV      #1000,KIPAR4      :MAP KERNEL I PAGE 4 TO 16K
9969 072116 012737 001000 172250      MOV      #1000,SIPAR4      :MAP SUPER I PAGE 4 TO 16K
9970 072124 012737 072442 000250      MOV      #10$,MVEEC      :SET M.M. VECTOR TO 10$
9971 072132 052737 000002 172516      BIS      #BIT1,MVR3      :ENABLE SUPERVISOR D-SPACE
9972      :THIS WILL TEST DSTM = 1 MTP1. BELOW ARE THE
9973      :ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
9974      :THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
9975      :INTO THE FLIP/FLOPS ON SSRB.
9976      : * D12.80 (111)
9977      : * D12.60 (155)
9978      : D12.20 (312)
9979      : FET.10 (260)
9980 072140 012737 072156 001112      MOV      #13$,SLPERR      :SET LOOP ON ERROR POINTER TO 13$
9981 072146 012702 100000      MOV      #100000,R2      :LOAD VIRTUAL ADDRESS INTO R2
9982 072152 012700 125252      MOV      #125252,R0      :LOAD TEST DATA INTO R0
9983 072156 010046      13$: MOV      R0,-(KSP)      :PUSH TEST DATA ON KERNEL STACK
9984 072160 105037 172310      CLRB     KIPDR4      :MAKE KERNEL I PAGE 4 NON-RESIDENT
9985 072164 000240      NOP      :THIS IS A SYNC POINT FOR SCOPING
9986 072166 006612      MTP1     (R2)      :LOAD TEST DATA INTO PHYSICAL 100000
9987 072170 112737 000006 172310      MOV      #006,KIPDR4      :MAKE KERNEL PAGE 4 RESIDENT
9988 072176 011201      MOV      (R2),R1      :READ FROM ADDRESS 100000
9989 072200 020001      CMP      R0,R1      :SEE IF DATA WAS STORED AT CORRECT PLACE
9990 072202 001401      BEQ      4$      :BRANCH IF STORE WAS CORRECT
9991 072204 104121      ERROR   121      :INCORRECT STORE
9992 072206      4$: :THIS WILL TEST DSTM = 3 MTP1. BELOW ARE THE
9993      :ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
9994      :THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
9995      :INTO THE FLIP/FLOPS ON SSRB.
9996      : D30.80 (113)
9997      : D30.10 (221)
9998      : D10.20 (233)
9999      : * D10.50 (311)
10000      : * D10.40 (157)
10001      : FET.01 (331)
10002 072206 012737 072226 001112      MOV      #14$,SLPERR      :SET LOOP ON ERROR POINTER TO 14$
    
```

K 15

CEKBEE0 11/70 MEM MGMT MACY11 30A(1052) 02-APR-80 09:15 PAGE 183
 CEKBEE.P11 02-APR-80 08:48 T111 MTP1 (SUPERVISOR) WITH SUPERVISOR D-SPACE ENABLED SEQ 0192

10003	072214	012737	010340	177776		MOV	#010340,PSW	:MAKE PREVIOUS MODE SUPERVISOR
10004	072222	012700	052525			MOV	#52525,R0	:LOAD TEST DATA INTO R0
10005	072226	010046			14\$:	MOV	R0,-(KSP)	:PUSH TEST DATA ON KERNEL STACK
10006	072230	105037	172310			CLRB	KIPDR4	:MAKE KERNEL I PAGE 4 NON-RESIDENT
10007	072234	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
10008	072236	006637	100000			MTP1	@#100000	:LOAD TEST DATA INTO PHYSICAL 100000
10009	072242	112737	000006	172310		MOVB	#006,KIPDR4	:MAKE KERNEL PAGE 4 RESIDENT
10010	072250	013701	100000			MOV	@#100000,R1	:READ FROM ADDRESS 100000
10011	072254	020001				CMP	R0,R1	:SEE IF DATA WAS STORED CORRECTLY
10012	072256	001401				BEQ	5\$:BRANCH IF STORE WAS CORRECT
10013	072260	104121				ERROR	121	:INCORRECT STORE
10014	072262				5\$:			:THIS WILL TEST DSTM = 4 MTP1. BELOW ARE THE
10015								:ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
10016								:THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
10017								:INTO THE FLIP/FLOPS ON SSRB.
10018							D45.80	(115)
10019							* D40.20	(121)
10020							* D10.40	(157)
10021							FET.01	(331)
10022	072262	012737	072302	001112		MOV	#15\$,SLPERR	:SET LOOP ON ERROR POINTER TO 15\$
10023	072270	012737	010340	177776		MOV	#010340,PSW	:MAKE PREVIOUS MODE SUPERVISOR
10024	072276	012700	125252			MOV	#125252,R0	:LOAD TEST DATA INTO R0
10025	072302	010046			15\$:	MOV	R0,-(KSP)	:PUSH TEST DATA ON KERNEL STACK
10026	072304	012702	100002			MOV	#100002,R2	:LOAD VIRTUAL ADDRESS INTO R2
10027	072310	105037	172310			CLRB	KIPDR4	:MAKE KERNEL I PAGE 4 NON-RESIDENT
10028	072314	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
10029	072316	006642				MTP1	-(R2)	:LOAD TEST DATA INTO PHYSICAL 100000
10030	072320	112737	000006	172310		MOVB	#006,KIPDR4	:MAKE KERNEL PAGE 4 RESIDENT
10031	072326	013701	100000			MOV	@#100000,R1	:READ FROM ADDRESS 100000
10032	072332	020001				CMP	R0,R1	:SEE IF DATA WAS STORED CORRECTLY
10033	072334	001401				BEQ	6\$:BRANCH IF STORE WAS CORRECT
10034	072336	104121				ERROR	121	:INCORRECT STORE
10035	072340				6\$:			:THIS WILL TEST DSTM = 6 MTP1. BELOW ARE THE
10036								:ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
10037								:THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
10038								:INTO THE FLIP/FLOPS ON SSRB.
10039							D67.80	(117)
10040							D67.00	(006)
10041							D67.10	(251)
10042							* D10.30	(122)
10043							* D10.40	(157)
10044							FET.01	(331)
10045	072340	012737	072362	001112		MOV	#16\$,SLPERR	:SET LOOP ON ERROR POINTER TO 16\$
10046	072346	012737	010340	177776		MOV	#010340,PSW	:MAKE PREVIOUS MODE SUPERVISOR
10047	072354	012700	052525			MOV	#52525,R0	:LOAD TEST DATA INTO R0
10048	072360	005002				CLR	R2	:MAKE REGISTER 2 ZERO
10049	072362	010046			16\$:	MOV	R0,-(KSP)	:PUSH TEST DATA ON KERNEL STACK
10050	072364	105037	172310			CLRB	KIPDR4	:MAKE KERNEL I PAGE 4 NON-RESIDENT
10051	072370	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
10052	072372	006662	100000			MTP1	100000(R2)	:LOAD TEST DATA INTO PHYSICAL 100000
10053	072376	112737	000006	172310		MOVB	#006,KIPDR4	:MAKE KERNEL PAGE 4 RESIDENT
10054	072404	013701	100000			MOV	@#100000,R1	:READ FROM ADDRESS 100000
10055	072410	020001				CMP	R0,R1	:SEE IF DATA WAS STORED CORRECTLY
10056	072412	001401				BEQ	7\$:BRANCH IF STORE WAS CORRECT
10057	072414	104121				ERROR	121	:INCORRECT STORE
10058	072416	012737	072050	001112	7\$:	MOV	#20\$,SLPERR	:SET LOOP POINTER TO START OF TEST

```

10060 072424 012737 032226 000250      MOV      #MMTRAP,MMVEC      ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
10061 072432 042737 000002 172516      BIC      #BIT1,MMR3        ;DISABLE SUPERVISOR D-SPACE
10062 072440 000413          BR       TST112           ;:BRANCH TO NEXT TEST
10063
10064 072442 013737 177572 001250 10$:      MOV      MMR0,PMR0        ;SAVE MMR0 FOR ERROR TYPEOUT
10065 072450 013737 177574 001252      MOV      MMR1,PMR1        ;SAVE MMR1 FOR ERROR TYPEOUT
10066 072456 013737 177576 001254      MOV      MMR2,PMR2        ;SAVE MMR2 FOR ERROR TYPEOUT
10067 072464 104117          ERROR   117              ;TRIED TO LOAD A N.R. PAGE 4
10068 072466 000002          RTI                    ;RETURN TO TEST
10069
10070
10071
10072
10073
10074
10075
10076
10077
10078
10079
10080
10081
10082
10083
10084
10085 072470
10086 072470 000004
10087 072472 012737 073062 001316      MOV      #TST113,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
10088
10089 072500 012700 077400          20$:      MOV      #77400,R0        ;TEST FOR ESCAPE ON PARITY ERRORS
10090
10091 072504 010037 172330          MOV      R0,KDPDR4        ;MAKE PAGE 4 NON-RESIDENT IN ALL MODES
10092 072510 010037 177630          MOV      R0,UDPDR4        ;EXCEPT KERNEL I-SPACE & USER I-SPACE
10093 072514 010037 172230          MOV      R0,SDPDR4        ;KERNEL D-SPACE PAGE 4
10094 072520 010037 172210          MOV      R0,SIPDR4        ;USER D-SPACE PAGE 4
10095 072524 012737 077406 172310      MOV      #77406,KIPDR4    ;SUPERVISOR D-SPACE PAGE 4
10096 072532 012737 077406 177610      MOV      #77406,UIPDR4    ;SUPERVISOR I-SPACE PAGE 4
10097 072540 012737 001000 172350      MOV      #1000,KIPAR4     ;KERNEL I-SPACE PAGE 4 READ/WRITE
10098 072546 012737 001000 177650      MOV      #1000,UIPAR4     ;USER I-SPACE PAGE 4 READ/WRITE
10099 072554 012700 036514          MOV      #36514,R0        ;MAP KERNEL I PAGE 4 TO 16K
10100 072560 010037 100000          MOV      R0,#100000       ;MAP USER I PAGE 4 TO 16K
10101 072564 012702 100000          MOV      #100000,R2       ;LOAD DATA PATTERN INTO R0
10102 072570 012737 073034 000250      MOV      #10$,MMVEC       ;LOAD DATA PATTERN INTO PHY 100000
10103 072576 105037 172310          CLR      KIPDR4           ;LOAD VIRTUAL ADDRESS INTO R2
10104 072602 012737 030340 177776      MOV      #030340,PSW      ;SET M.M. VECTOR TO 10$
10105 072610 006506          1$:      MFPI    USP              ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
10106 072612 022706 001100          CMP      #KERSTK,KSP      ;MAKE PREVIOUS MODE USER
10107 072616 001407          BEQ     3$               ;PUT USER STACK POINTER ON KERNEL STACK
10108 072620 012601          MOV      (KSP)+,R1        ;WAS SOMETHING PUSHED ON STACK AT 1$
10109 072622 012702 000600          MOV      #USESTK,R2       ;BRANCH IF NOTHING WAS PUSHED
10110 072626 020201          CMP      R2,R1            ;POP KERNEL STACK INTO R1
10111 072630 001403          BEQ     2$               ;EXPECTING 600 AS USP
10112 072632 104114          ERROR   114              ;DID YOU GET THE RIGHT POINTER?
10113 072634 000401          BR      2$               ;BRANCH IF YOU DID
10114 072636 104115          3$:      ERROR   115              ;WRONG THING WAS PUSHED ON STACK
;BRANCH TO NEXT TRY
;NOTHING WAS PUSHED ON THE STACK

```

*TEST 112 MOVE FROM PREVIOUS (USER) I-SPACE

* THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS USER THE
 * "USER SPACE (1) L" FLIP-FLOP IS SET AND THE FETCH IS FROM
 * USER MODE.
 * THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
 * WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
 * THEIR ADDRESSES).
 * IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
 * WILL OCCUR AND TRAP TO 10\$, WHERE THE ERRORS ARE REPORTED.

TST112:

SCOPE

```

MOV      #TST113,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
;MAKE PAGE 4 NON-RESIDENT IN ALL MODES
;EXCEPT KERNEL I-SPACE & USER I-SPACE
;KERNEL D-SPACE PAGE 4
;USER D-SPACE PAGE 4
;SUPERVISOR D-SPACE PAGE 4
;SUPERVISOR I-SPACE PAGE 4
;KERNEL I-SPACE PAGE 4 READ/WRITE
;USER I-SPACE PAGE 4 READ/WRITE
;MAP KERNEL I PAGE 4 TO 16K
;MAP USER I PAGE 4 TO 16K
;LOAD DATA PATTERN INTO R0
;LOAD DATA PATTERN INTO PHY 100000
;LOAD VIRTUAL ADDRESS INTO R2
;SET M.M. VECTOR TO 10$
;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
;MAKE PREVIOUS MODE USER
;PUT USER STACK POINTER ON KERNEL STACK
;WAS SOMETHING PUSHED ON STACK AT 1$
;BRANCH IF NOTHING WAS PUSHED
;POP KERNEL STACK INTO R1
;EXPECTING 600 AS USP
;DID YOU GET THE RIGHT POINTER?
;BRANCH IF YOU DID
;WRONG THING WAS PUSHED ON STACK
;BRANCH TO NEXT TRY
;NOTHING WAS PUSHED ON THE STACK

```

```

10115 072640      2$:      ;THE FOLLOWING WILL TEST DSTM=1 MFPI. BELOW ARE THE
10116              ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10117              ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
10118              ;INTO THE F/F'S ON SSRB.
10119              ; * D12.00 (001)
10120              ; * D12.10 (175)
10121              ; * MFP.00 (066)
10122              ; * MFP.10 (250)
10123              ; * SVC.80 (222)
10124              ; * SVC.90 (300)
10125              ; * FET.00 (217)
10126 072640 012737 072646 001112      MOV #12$,SLPERR      ;SET LOOP ON ERROR POINTER TO 12$
10127 072646 012737 030340 177776 12$:  MOV #030340,PSW     ;MAKE PREVIOUS MODE USER
10128 072654 012702 100000              MOV #100000,R2     ;PUT VIRTUAL ADDRESS IN R2
10129 072660 000240              NOP                ;THIS IS A SYNC POINT FOR SCOPING
10130 072662 006512      MFPI (R2)          ;READ FROM PHYSICAL 100000
10131 072664 012601      MOV (KSP)+,R1     ;POP KERNEL STACK INTO R1
10132 072666 020001      CMP R0,R1        ;WAS DATA FETCHED SAME AS STORED
10133 072670 001401      BEQ 4$          ;BRANCH IF CORRECT DATA WAS FETCHED
10134 072672 104116      ERROR 116       ;WRONG DATA WAS FETCHED
10135 072674      4$:      ;THE FOLLOWING WILL TEST DSTM=2 MFPI. BELOW ARE THE
10136              ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10137              ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
10138              ;INTO THE F/F'S ON SSRB.
10139              ; * D12.01 (002)
10140              ; * D12.10 (175)
10141              ; * MFP.00 (066)
10142              ; * MFP.10 (250)
10143              ; * SVC.80 (222)
10144              ; * SVC.90 (300)
10145              ; * FET.00 (217)
10146 072674 012737 072702 001112      MOV #14$,SLPERR     ;SET LOOP ON ERROR POINTER TO 14$
10147 072702 012737 030340 177776 14$:  MOV #030340,PSW     ;MAKE PREVIOUS MODE USER
10148 072710 012702 100000              MOV #100000,R2     ;PUT VIRTUAL ADDRESS IN R2
10149 072714 000240              NOP                ;THIS IS A SYNC POINT FOR SCOPING
10150 072716 006522      MFPI (R2)+        ;READ FROM PHYSICAL 100000
10151 072720 012601      MOV (KSP)+,R1     ;POP KERNEL STACK INTO R1
10152 072722 020001      CMP R0,R1        ;WAS DATA FETCHED SAME AS STORED
10153 072724 001401      BEQ 5$          ;BRANCH IF CORRECT DATA WAS FETCHED
10154 072726 104116      ERROR 116       ;WRONG DATA WAS FETCHED
10155 072730      5$:      ;THE FOLLOWING WILL TEST DSTM=3 MFPI. BELOW ARE THE
10156              ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10157              ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
10158              ;INTO THE F/F'S ON SSRB.
10159              ; * D30.00 (003)
10160              ; * D30.10 (221)
10161              ; * D10.20 (233)
10162              ; * D10.50 (311)
10163              ; * D10.60 (177)
10164              ; * MFP.00 (066)
10165              ; * MFP.10 (250)
10166              ; * SVC.80 (222)
10167              ; * SVC.90 (300)
10168              ; * FET.00 (217)
10169 072730 012737 072736 001112      MOV #15$,SLPERR     ;SET LOOP ON ERROR POINTER TO 15$
10170 072736 012737 030340 177776 15$:  MOV #030340,PSW     ;MAKE PREVIOUS MODE USER
    
```

```

10171 072744 000240      NOP
10172 072746 006537 100000 MFPJ @#100000 ;THIS IS A SYNC POINT FOR SCOPING
10173 072752 012601      MOV (KSP)+,R1 ;READ FROM PHYSICAL 100000
10174 072754 020001      CMP R0,R1 ;POP KERNEL STACK INTO R1
10175 072756 001401      BEQ 6$ ;WAS DATA FETCHED SAME AS STORED
10176 072760 104116      ERROR 116 ;BRANCH IF CORRECT DATA WAS FETCHED
10177 072762      ;WRONG DATA WAS FETCHED
10178      6$: ;THE FOLLOWING WILL TEST DSTM=4 MFPJ. BELOW ARE THE
10179      ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10180      ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
10181      ;INTO THE F/F'S ON SSRB.
10182      : D45.00 (004)
10183      : * D10.30 (122)
10184      : * D10.60 (177)
10185      : * MFP.00 (066)
10186      : MFP.10 (250)
10187      : SVC.80 (222)
10188      : SVC.90 (300)
10189      : FET.00 (217)
10189 072762 012737 072770 001112 MOV #16$,SLPERR ;SET LOOP ON ERROR POINTER TO 16$
10190 072770 012737 030340 177776 16$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
10191 072776 012702 100002 MOV #100002,R2 ;LOAD VIRTUAL ADDRESS INTO R2
10192 073002 000240      NOP ;THIS IS A SYNC POINT FOR SCOPING
10193 073004 006542      MFPJ -(R2) ;READ FROM PHYSICAL 100000
10194 073006 012601      MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
10195 073010 020001      CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
10196 073012 001401      BEQ 7$ ;BRANCH IF CORRECT DATA WAS FETCHED
10197 073014 104116      ERROR 116 ;WRONG DATA WAS FETCHED
10198 073016 012737 032226 000250 7$: MOV #MMTRAP,MMVEC ;SET M.M.VECTOR TO NORMAL ROUTINE
10199 073020 012737 072500 001112 MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
10200 073032 000413      BR TST113 ;BRANCH TO NEXT TEST
10201
10202
10203 073034 013737 177572 001250 10$: MOV MMRO,PMRO ;SAVE MMRO FOR ERROR TYPEOUT
10204 073042 013737 177574 001252 MOV MMR1,PMR1 ;SAVE MMR1 FOR ERROR TYPEOUT
10205 073050 013737 177576 001254 MOV MMR2,PMR2 ;SAVE MMR2 FOR ERROR TYPEOUT
10206 073056 104117      ERROR 117 ;TRIED TO READ NON-RESIDENT PAGE
10207 073060 000002      RTI ;RETURN TO TEST
10208
10209
10210      :*****
10211      :*TEST 113 MOVE FROM PREVIOUS (KERNEL) I-SPACE TO SUPERVISOR MODE
10212      :*
10213      :* THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE
10214      :* 'KERNEL SPACE (1) L' FLIP-FLOP IS SET AND THE FETCH IS FROM
10215      :* KERNEL MODE.
10216      :* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
10217      :* WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
10218      :* THEIR ADDRESSES).
10219      :* IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
10220      :* WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
10221      :*****
10222      :TST113:
10222 073062      SCOPE
10223 073062 000004      MOV #TST114,NXTTST ;SAVE STARTING ADDRESS OF NEXT
10224 073064 012737 073526 001316 ;TEST FOR ESCAPE ON PARITY ERRORS
10225      MOV #77406,R0 ;MAKE ALL SUPER I-SPACE PAGES RESIDENT
10226 073072 012700 077406
    
```

```

10227      :READ/WRITE, LENGTH 200 BLOCKS
10228 073076 012702 000010      MOV      #10,R2      :SET COUNT TO LOAD 8 ADDRESSES
10229 073102 012701 172200      MOV      #SIPDR0,R1  :PUT ADDRESS OF FIRST PDR IN R1
10230 073106 010021      19$: MOV      R0,(R1)+   :LOAD R0 INTO PDR ADDRESSED BY R1
10231 073110 077202      SOB      R2,19$     :BRANCH BACK TO 19$ IF R2 IS NOT ZERO
10232 073112 012737 073126 001110      MOV      #20$,SLPADR :SET LOOP POINTER TO 20$
10233 073120 012737 073126 001112      MOV      #20$,SLPERR :SET LOOP ON ERROR TO 20$
10234 073126 012737 040340 177776      20$: MOV      #040340,PSW :GO TO SUPERVISOR MODE FOR THIS TEST
10235 073134 012700 077400      MOV      #77400,R0   :MAKE PAGE 4 NON-RESIDENT IN ALL MODES
10236      :EXCEPT SUPER I-SPACE & KERNEL I-SPACE
10237 073140 010037 172330      MOV      R0,KDPDR4  :KERNEL D-SPACE PAGE 4
10238 073144 010037 172230      MOV      R0,SDPDR4  :SUPERVISOR D-SPACE PAGE 4
10239 073150 010037 177630      MOV      R0,UDPDR4  :USER D-SPACE PAGE 4
10240 073154 010037 177610      MOV      R0,UJPDRA  :USER I-SPACE PAGE 4
10241 073160 012737 077406 172310      MOV      #77406,KIPDR4 :KERNEL I-SPACE PAGE 4 READ/WRITE
10242 073166 012737 077406 172210      MOV      #77406,SIPDR4 :SUPER I-SPACE PAGE 4 READ/WRITE
10243 073174 012737 001000 172350      MOV      #1000,KIPAR4 :MAP KERNEL I PAGE 4 TO 16K
10244 073202 012737 001000 172250      MOV      #1000,SIPAR4 :MAP SUPER I PAGE 4 TO 16K
10245 073210 012700 036514      MOV      #36514,R0   :LOAD DATA PATTERN INTO R0
10246 073214 010037 100000      MOV      R0,#100000 :LOAD DATA PATTERN INTO PHY 100000
10247 073220 012702 100000      MOV      #100000,R2  :LOAD VIRTUAL ADDRESS INTO R2
10248 073224 012737 073500 000250      MOV      #10$,MVEEC  :SET M.M. VECTOR TO 10$
10249 073232 105037 172210      CLRB    SIPDR4     :MAKE SUPER I-SPACE PAGE 4 NON-RESIDENT
10250 073236 012737 040340 177776      MOV      #040340,PSW :MAKE PREVIOUS MODE KERNEL PRESENT SUPER
10251 073244 000240      NOP      :THIS IS A SYNC POINT FOR SCOPING
10252 073246 006506      1$: MFPI    KSP      :PUT KERNEL STACK POINTER ON SUPER STACK
10253 073250 022706 000700      CMP      #SUPSTK,SSP :WAS SOMETHING PUSHED ON STACK AT 1$
10254 073254 001407      BEQ     3$        :BRANCH IF NOTHING WAS PUSHED
10255 073256 012601      MOV      (SSP)+,R1   :POP SUPERVISOR STACK INTO R1
10256 073260 012702 001100      MOV      #KERSTK,R2 :EXPECTING 1100 AS KSP
10257 073264 020201      CMP      R2,R1     :DID YOU GET THE RIGHT POINTER?
10258 073266 001403      BEQ     2$        :BRANCH IF YOU DID
10259 073270 104114      ERROR   114       :WRONG THING WAS PUSHED ON STACK
10260 073272 000401      BR      2$        :BRANCH TO NEXT TRY
10261 073274 104115      3$: ERROR   115       :NOTHING PUSHED ON STACK
10262 073276      2$: :THE FOLLOWING WILL TEST DSTM=1 MFPI. BELOW ARE THE
10263      :ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10264      :THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
10265      :INTO THE F/F'S ON SSRB.
10266      : * D12.00 (001)
10267      : * D12.10 (175)
10268      : * MFP.00 (066)
10269      : * MFP.10 (250)
10270      : * SVC.80 (222)
10271      : * SVC.90 (300)
10272      : * FET.00 (217)
10273 073276 012737 073304 001112      12$: MOV      #12$,SLPERR :SET LOOP ON ERROR POINTER TO 12$
10274 073304 012737 040340 177776      MOV      #040340,PSW :MAKE PREVIOUS MODE KERNEL PRESENT SUPER
10275 073312 012702 100000      MOV      #100000,R2 :LOAD VIRTUAL ADDRESS INTO R2
10276 073316 000240      NOP      :THIS IS A SYNC POINT FOR SCOPING
10277 073320 006512      MFPI    (R2)      :READ FROM PHYSICAL 100000
10278 073322 012601      MOV      (SSP)+,R1   :POP SUPER STACK INTO R1
10279 073324 020001      CMP      R0,R1     :WAS DATA FETCHED SAME AS STORED
10280 073326 001401      BEQ     4$        :BRANCH IF CORRECT DATA WAS FETCHED
10281 073330 104116      ERROR   116       :WRONG DATA WAS FETCHED
10282 073332      4$: :THE FOLLOWING WILL TEST DSTM=2 MFPI. BELOW ARE THE

```

```

10283 :ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10284 :THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
10285 :INTO THE F/F'S ON SSRB.
10286 : * D12.01 (002)
10287 : * D12.10 (175)
10288 : * MFP.00 (066)
10289 : MFP.10 (250)
10290 : SVC.80 (222)
10291 : SVC.90 (300)
10292 : FET.00 (217)
10293 073332 012737 073340 001112 14$: MOV #14$, $LPERR ;SET LOOP ON ERROR POINTER TO 14$
10294 073340 012737 040340 177776 MOV #040340, PSW ;MAKE PREVIOUS MODE KERNEL PRESENT SUPER
10295 073346 012702 100000 MOV #100000, R2 ;LOAD VIRTUAL ADDRESS INTO R2
10296 073352 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
10297 073354 006522 MFPI (R2)+ ;READ FROM PHYSICAL 100000
10298 073356 012601 MOV (SSP)+, R1 ;POP SUPER STACK INTO R1
10299 073360 020001 CMP R0, R1 ;WAS DATA FETCHED SAME AS STORED
10300 073362 001401 BEQ $S ;BRANCH IF CORRECT DATA WAS FETCHED
10301 073364 104116 ERROR 116 ;WRONG DATA WAS FETCHED
10302 073366 5$: ;THE FOLLOWING WILL TEST DSTN=3 MFPI. BELOW ARE THE
10303 :ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10304 :THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
10305 :INTO THE F/F'S ON SSRB.
10306 : D30.00 (003)
10307 : D30.10 (221)
10308 : D10.20 (233)
10309 : * D10.50 (311)
10310 : * D10.60 (177)
10311 : * MFP.00 (066)
10312 : MFP.10 (250)
10313 : SVC.80 (222)
10314 : SVC.90 (300)
10315 : FET.00 (217)
10316 073366 012737 073374 001112 15$: MOV #15$, $LPERR ;SET LOOP ON ERROR POINTER TO 15$
10317 073374 012737 040340 177776 MOV #040340, PSW ;MAKE PREVIOUS MODE KERNEL PRESENT SUPER
10318 073402 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
10319 073404 006537 100000 MFPI @#100000 ;READ FROM PHYSICAL 100000
10320 073410 012601 MOV (SSP)+, R1 ;POP SUPERVISOR STACK INTO R1
10321 073412 020001 CMP R0, R1 ;WAS DATA FETCHED SAME AS STORED
10322 073414 001401 BEQ $S ;BRANCH IF CORRECT DATA WAS FETCHED
10323 073416 104116 ERROR 116 ;WRONG DATA WAS FETCHED
10324 073420 6$: ;THE FOLLOWING WILL TEST DSTN=4 MFPI. BELOW ARE THE
10325 :ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10326 :THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
10327 :INTO THE F/F'S ON SSRB.
10328 : D45.00 (004)
10329 : * D10.30 (122)
10330 : * D10.60 (177)
10331 : * MFP.00 (066)
10332 : MFP.10 (250)
10333 : SVC.80 (222)
10334 : SVC.90 (300)
10335 : FET.00 (217)
10336 073420 012737 073426 001112 16$: MOV #16$, $LPERR ;SET LOOP ON ERROR POINTER TO 16$
10337 073426 012737 040340 177776 MOV #040340, PSW ;MAKE PREVIOUS MODE KERNEL PRESENT SUPER
10338 073434 012702 100002 MOV #100002, R2 ;LOAD VIRTUAL ADDRESS INTO R2

```

```

10339 073440 000240      NOP                ;THIS IS A SYNC POINT FOR SCOPING
10340 073442 006542      MFPI              ;READ FROM PHYSICAL 10000
10341 073444 012601      MOV              (SSP)+,R1 ;POP SUPERVISOR STACK INTO R1
10342 073446 020001      CMP              R0,R1    ;WAS DATA FETCHED SAME AS STORED
10343 073450 001401      BEQ              7$      ;BRANCH IF CORRECT DATA WAS FETCHED
10344 073452 104116      ERROR           116     ;WRONG DATA WAS FETCHED
10345 073454 012737 032226 000250 7$: MOV      #MMTRAP,MMVEC ;SET M.M.VECTOR TO NORMAL ROUTINE
10346 073462 012737 000340 177776 MOV      #00340,PSW    ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
10347 073470 012737 073126 001112 MOV      #20$,SLPERR  ;SET LOOP POINTER TO START OF TEST
10348 073476 000413      BR              TST114 ;:BRANCH TO NEXT TEST
10349
10350

```

```

10351 073500 013737 177572 001250 10$: MOV      #MMRO,MMRO    ;SAVE MMRO FOR ERROR TYPEOUT
10352 073506 013737 177574 001252 MOV      #MMR1,MMR1    ;SAVE MMR1 FOR ERROR TYPEOUT
10353 073514 013737 177576 001254 MOV      #MMR2,MMR2    ;SAVE MMR2 FOR ERROR TYPEOUT
10354 073522 104117      ERROR           117     ;TRIED TO READ NON-RESIDENT PAGE
10355 073524 000002      RTI              ;RETURN TO TEST
10356
10357
10358
10359

```

```

*****
*TEST 114      MFPD (SUPERVISOR) WITH SUPERVISOR D-SPACE ENABLED
*

```

```

* THIS TEST CHECKS THAT 'SSRB IR15 L' CAN INHIBIT THE ASSERTING
* OF 'SSRB I SPACEB L' SO THAT THE REFERENCE IS TO D-SPACE IF
* THE INSTRUCTION IS MFPD (OR MTPD). [THESE INSTRUCTIONS HAVE
* BIT 15 SET IN THE I.R.]
* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
* WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
* THEIR ADDRESSES).
* IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
* WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
*

```

```

*****
TST114:

```

```

10373 073526      SCOPE
10374 073526 000004      MOV      #TST115,NXTTST ;SAVE STARTING ADDRESS OF NEXT
10375 073530 012737 074126 001316      MOV      #77406,R0      ;TEST FOR ESCAPE ON PARITY ERRORS
10376                                     ;MAKE ALL KERNEL I-SPACE PAGES RESIDENT
10377 073536 012700 077406      MOV      #10,R2         ;READ/WRITE, LENGTH 200 BLOCKS
10378                                     ;SET COUNT TO LOAD 8 ADDRESSES
10379 073542 012702 000010      MOV      #KIPDR0,R1    ;PUT ADDRESS OF FIRST PDR IN R1
10380 073546 012701 172300      MOV      R0,(R1)+      ;LOAD R0 INTO PDR ADDRESSED BY R1
10381 073552 010021 19$: MOV      R2,19$        ;BRANCH BACK TO 19$ IF R2 IS NOT ZERO
10382 073554 077202      SOB      #20$,SLPADR   ;SET LOOP POINTER TO 20$
10383 073556 012737 073572 001110 MOV      #20$,SLPERR    ;SET LOOP ON ERROR TO 20$
10384 073564 012737 073572 001112 MOV      #77400,R0      ;MAKE PAGE 4 IN ALL BUT SUPERVISOR D
10385 073572 012700 077400 20$: MOV      R0,KDPDR4     ;AND KERNEL I NON-RESIDENT
10386                                     ;KERNEL D-SPACE PAGE 4
10387 073576 010037 172330      MOV      R0,SIPDR4     ;SUPERVISOR I-SPACE PAGE 4
10388 073602 010037 172210      MOV      R0,UDPDR4     ;USER D-SPACE PAGE 4
10389 073606 010037 177630      MOV      R0,UIPDR4     ;USER I-SPACE PAGE 4
10390 073612 010037 177610      MOV      #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
10391 073616 012737 077406 172310 MOV      #77406,SDPDR4  ;SUPER D-SPACE PAGE 4 READ/WRITE
10392 073624 012737 077406 172230 MOV      #1000,KIPAR4   ;MAP KERNEL I PAGE 4 TO 16K
10393 073632 012737 001000 172350 MOV      #1000,SDPAR4   ;MAP SUPER D PAGE 4 TO 16K
10394 073640 012737 001000 172270

```



```

10395 073646 012700 036514      MOV      #36514,R0      ;LOAD DATA PATTERN INTO R0
10396 073652 010037 100000      MOV      R0,#100000    ;LOAD DATA PATTERN INTO PHY 100000
10397 073656 012737 074100 000250  MOV      #10$,MVEC     ;SET M.M. VECTOR TO 10$
10398 073664 052737 000002 172516  BIS      #BIT1,MMR3    ;ENABLE SUPERVISOR D-SPACE
10399 073672 105037 172310      CLR      KIPDR4       ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
10400 073676      2$:      ;THE FOLLOWING WILL TEST DSTM=1 MFPD. BELOW ARE THE
;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
;INTO THE F/F'S ON SSRB.
;      * D12.00      (001)
;      * D12.10      (175)
;      * MFP.00      (066)
;      * MFP.10      (250)
;      * SVC.80      (222)
;      * SVC.90      (300)
;      * FET.00      (217)
10411 073676 012737 073704 001112  MOV      #12$,SLPERR   ;SET LOOP ON ERROR POINTER TO 12$
10412 073704 012737 010340 177776 12$:      MOV      #010340,PSW   ;MAKE PREVIOUS MODE SUPERVISOR
10413 073712 012702 100000      MOV      #100000,R2   ;LOAD VIRTUAL ADDRESS INTO R2
10414 073716 000240      NOP                    ;THIS IS A SYNC POINT FOR SCOPING
10415 073720 106512      MFPD      (R2)        ;READ FROM PHYSICAL 100000
10416 073722 012601      MOV      (KSP)+,R1    ;POP KERNEL STACK INTO R1
10417 073724 020001      CMP      R0,R1        ;WAS DATA FETCHED SAME AS STORED
10418 073726 001401      BEQ      4$           ;BRANCH IF CORRECT DATA WAS FETCHED
10419 073730 104116      ERROR    116         ;WRONG DATA WAS FETCHED
10420 073732      4$:      ;THE FOLLOWING WILL TEST DSTM=2 MFPD. BELOW ARE THE
;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
;INTO THE F/F'S ON SSRB.
;      * D12.01      (002)
;      * D12.10      (175)
;      * MFP.00      (066)
;      * MFP.10      (250)
;      * SVC.80      (222)
;      * SVC.90      (300)
;      * FET.00      (217)
10431 073732 012737 073740 001112  MOV      #14$,SLPERR   ;SET LOOP ON ERROR POINTER TO 14$
10432 073740 012737 010340 177776 14$:      MOV      #010340,PSW   ;MAKE PREVIOUS MODE SUPERVISOR
10433 073746 012702 100000      MOV      #100000,R2   ;LOAD VIRTUAL ADDRESS INTO R2
10434 073752 000240      NOP                    ;THIS IS A SYNC POINT FOR SCOPING
10435 073754 106522      MFPD      (R2)+       ;READ FROM PHYSICAL 100000
10436 073756 012601      MOV      (KSP)+,R1    ;POP KERNEL STACK INTO R1
10437 073760 020001      CMP      R0,R1        ;WAS DATA FETCHED SAME AS STORED
10438 073762 001401      BEQ      5$           ;BRANCH IF CORRECT DATA WAS FETCHED
10439 073764 104116      ERROR    116         ;WRONG DATA WAS FETCHED
10440 073766      5$:      ;THE FOLLOWING WILL TEST DSTM=3 MFPD. BELOW ARE THE
;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
;INTO THE F/F'S ON SSRB.
;      * D30.00      (003)
;      * D30.10      (221)
;      * D10.20      (233)
;      * D10.50      (311)
;      * D10.60      (177)
;      * MFP.00      (066)
;      * MFP.10      (250)
10450

```

```
10451 : SVC.80 (222)
10452 : SVC.90 (300)
10453 : FET.00 (217)
10454 073766 012737 073774 001112 : MOV #15$,SLPERR ;SET LOOP ON ERROR POINTER TO 15$
10455 073774 012737 010340 177776 15$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
10456 074002 000240 : NOP ;THIS IS A SYNC POINT FOR SCOPING
10457 074004 106537 100000 : MFPD @#100000 ;READ FROM PHYSICAL 100000
10458 074010 012601 : MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
10459 074012 020001 : CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
10460 074014 001401 : BEQ 6$ ;BRANCH IF CORRECT DATA WAS FETCHED
10461 074016 104116 : ERROR 116 ;WRONG DATA WAS FETCHED
10462 074020 6$: :THE FOLLOWING WILL TEST DSTN=4 MFPD. BELOW ARE THE
10463 :ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10464 :THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
10465 :INTO THE F/F'S ON SSRB.
10466 : D45.00 (004)
10467 : * D10.30 (122)
10468 : * D10.60 (177)
10469 : * MFP.00 (066)
10470 : MFP.10 (250)
10471 : SVC.80 (222)
10472 : SVC.90 (300)
10473 : FET.00 (217)
10474 074020 012737 074026 001112 : MOV #16$,SLPERR ;SET LOOP ON ERROR POINTER TO 16$
10475 074026 012737 010340 177776 16$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
10476 074034 012702 100002 : MOV #100002,R2 ;LOAD VIRTUAL ADDRESS INTO R2
10477 074040 000240 : NOP ;THIS IS A SYNC POINT FOR SCOPING
10478 074042 106542 : MFPD -(R2) ;READ FROM PHYSICAL 100000
10479 074044 012601 : MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
10480 074046 020001 : CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
10481 074050 001401 : BEQ 7$ ;BRANCH IF CORRECT DATA WAS FETCHED
10482 074052 104116 : ERROR 116 ;WRONG DATA WAS FETCHED
10483 074054 012737 032226 000250 7$: MOV #MTRAP,MVEC ;SET M.M.VECTOR TO NORMAL ROUTINE
10484 074062 012737 073572 001112 : MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
10485 074070 042737 000002 172516 : BIC #BIT1,MPR3 ;DISABLE SUPERVISOR D-SPACE
10486 074076 000413 : BR TST115 ;:BRANCH TO NEXT TEST
10487
10488
10489 074100 013737 177572 001250 10$: MOV MPR0,MPR0 ;SAVE MPR0 FOR ERROR TYPEOUT
10490 074106 013737 177574 001252 : MOV MPR1,MPR1 ;SAVE MPR1 FOR ERROR TYPEOUT
10491 074114 013737 177576 001254 : MOV MPR2,MPR2 ;SAVE MPR2 FOR ERROR TYPEOUT
10492 074122 104117 : ERROR 117 ;TRIED TO READ NON-RESIDENT PAGE
10493 074124 000002 : RTI ;RETURN TO TEST
10494
10495
10496
10497
10498
10499
10500
10501
10502
10503
10504
10505
10506
```

```
*****
*TEST 115 MFPD (USER/PREV.USER) WITH USER D-SPACE ENABLED
*
* THIS TEST CHECKS THAT, IF THE INSTRUCTION IS EITHER MFPD OR
* MTPD AND BOTH THE PRESENT AND PREVIOUS MODES ARE USER
* (PS=17XXXX), THEN 'SSRB I SPACEB L' IS NOT ASSERTED AND
* D-SPACE IS USED IF IT WAS ENABLED. [IN THIS WAY AN OPERATING
* SYSTEM CAN MAKE SOME PROPRIETARY CODE 'EXECUTE ONLY' FOR
* THE USER.]
* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
* WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
```

```

10507 :* THEIR ADDRESSES).
10508 :* IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
10509 :* WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
10510 :*
10511 :*****
10512 074126 000004 TST115: SCOPE
10513 074130 012737 074552 001316 MOV #TST116,NXTTST ;SET ESCAPE POINTER TO EOP ROUTINE
10514 074136 012700 077406 MOV #77406,R0 ;MAKE ALL USER I-SPACE PAGES RESIDENT
10515 ;READ/WRITE, LENGTH 200 BLOCKS
10516 074142 012702 000010 MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
10517 074146 012701 177600 MOV #UIPDRO,R1 ;PUT ADDRESS OF FIRST PDR IN R1
10518 074152 010021 19$: MOV R0,(R1)+ ;LOAD R0 INTO PDR ADDRESSED BY R1
10519 074154 077202 SOB R2,19$ ;BRANCH BACK TO 19$ IF R2 IS NOT ZERO
10520 074156 012737 074172 001110 MOV #20$,SLPADR ;SET LOOP POINTER TO 20$
10521 074164 012737 074172 001112 MOV #20$,SLPERR ;SET LOOP ON ERROR TO 20$
10522 074172 012737 077406 172310 20$: MOV #77406,KIPDR4 ;MAKE KERNEL I PAGE 4 R/W, 200 BLOCKS
10523 074200 012700 077400 MOV #77400,R0 ;MAKE PAGE 4 IN ALL BUT USER D
10524 ;AND USER I NON-RESIDENT
10525 074204 010037 172330 MOV R0,KDPDR4 ;KERNEL D-SPACE PAGE 4
10526 074210 010037 172210 MOV R0,SIPDR4 ;SUPERVISOR I-SPACE PAGE 4
10527 074214 010037 172230 MOV R0,SDPDR4 ;SUPERVISOR D-SPACE PAGE 4
10528 074220 012737 077406 177630 MOV #77406,UDPDR4 ;USER D-SPACE PAGE 4 READ/WRITE
10529 074226 012737 001000 172350 MOV #1000,KIPAR4 ;MAP KERNEL I PAGE 4 TO 16K
10530 074234 012737 001000 177650 MOV #1000,UIPAR4 ;MAP USER I PAGE 4 TO 16K
10531 074242 012737 001000 177670 MOV #1000,UDPAR4 ;MAP USER D PAGE 4 TO 16K
10532 074250 012700 036514 MOV #36514,R0 ;LOAD DATA PATTERN INTO R0

```

```

10533 074254 010037 100000      MOV     R0,#100000      ;LOAD DATA PATTERN INTO PHY 100000
10534 074260 012737 074514 000250  MOV     #108,MFVEC      ;SET M.M. VECTOR TO 108
10535 074266 052737 000001 172516  BIS     #BIT0,MFR3      ;ENABLE USER D-SPACE
10536 074274 105037 172310      CLRB    KIPDR4         ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
10537 074300 105037 177610      CLRB    UIPDR4         ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT
10538 074304      2$:      ;THE FOLLOWING WILL TEST DSTM=1 MFPI. BELOW ARE THE
10539      ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10540      ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
10541      ;INTO THE F/F'S ON SSRB.
10542      ; * D12.00 (001)
10543      ; * D12.10 (175)
10544      ; * MFP.00 (066)
10545      ; * MFP.10 (250)
10546      ; * SVC.80 (222)
10547      ; * SVC.90 (300)
10548      ; * FET.00 (217)
10549 074304 012737 074312 001112  MOV     #12$,SLPERR     ;SET LOOP ON ERROR POINTER TO 12$
10550 074312 012737 170340 177776 12$:    MOV     #170340,PSW     ;MAKE PREVIOUS MODE USER
10551 074320 012702 100000      MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
10552 074324 000240      NOP                     ;THIS IS A SYNC POINT FOR SCOPING
10553 074326 006512      MFPI    (R2)            ;READ FROM PHYSICAL 100000
10554 074330 012601      MOV     (USP)+,R1       ;POP USER STACK INTO R1
10555 074332 020001      CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
10556 074334 001401      BEQ     4$              ;BRANCH IF CORRECT DATA WAS FETCHED
10557 074336 104116      ERROR   116            ;WRONG DATA WAS FETCHED
10558 074340      4$:      ;THE FOLLOWING WILL TEST DSTM=2 MFPI. BELOW ARE THE
10559      ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10560      ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
10561      ;INTO THE F/F'S ON SSRB.
10562      ; * D12.01 (002)
10563      ; * D12.10 (175)
10564      ; * MFP.00 (066)
10565      ; * MFP.10 (250)
10566      ; * SVC.80 (222)
10567      ; * SVC.90 (300)
10568      ; * FET.00 (217)
10569 074340 012737 074346 001112  MOV     #14$,SLPERR     ;SET LOOP ON ERROR POINTER TO 14$
10570 074346 012737 170340 177776 14$:    MOV     #170340,PSW     ;MAKE PREVIOUS MODE USER
10571 074354 012702 100000      MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
10572 074360 000240      NOP                     ;THIS IS A SYNC POINT FOR SCOPING
10573 074362 006522      MFPI    (R2)+          ;READ FROM PHYSICAL 100000
10574 074364 012601      MOV     (USP)+,R1       ;POP USER STACK INTO R1
10575 074366 020001      CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
10576 074370 001401      BEQ     5$              ;BRANCH IF CORRECT DATA WAS FETCHED
10577 074372 104116      ERROR   116            ;WRONG DATA WAS FETCHED
10578 074374      5$:      ;THE FOLLOWING WILL TEST DSTM=3 MFPI. BELOW ARE THE
10579      ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10580      ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
10581      ;INTO THE F/F'S ON SSRB.
10582      ; * D30.00 (003)
10583      ; * D30.10 (221)
10584      ; * D10.20 (233)
10585      ; * D10.50 (311)
10586      ; * D10.60 (177)
10587      ; * MFP.00 (066)
10588      ; * MFP.10 (250)
    
```

```
10589      :      SVC.80      (222)
10590      :      SVC.90      (300)
10591      :      FET.00      (217)
10592 074374 012737 074402 001112      :      MOV      #15$,SLPERR      ;SET LOOP ON ERROR POINTER TO 15$
10593 074402 012737 170340 177776 15$:      MOV      #170340,PSW      ;MAKE PREVIOUS MODE USER
10594 074410 000240      :      NOP      ;THIS IS A SYNC POINT FOR SCOPING
10595 074412 006537 100000      :      MFPI     @#100000      ;READ FROM PHYSICAL 100000
10596 074416 012601      :      MOV      (USP)+,R1      ;POP USER STACK INTO R1
10597 074420 020001      :      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
10598 074422 001401      :      BEQ      6$,          ;BRANCH IF CORRECT DATA WAS FETCHED
10599 074424 104116      :      ERROR   116      ;WRONG DATA WAS FETCHED
10600 074426      6$:      ;THE FOLLOWING WILL TEST DSTM=4 MFPI. BELOW ARE THE
10601      :      ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10602      :      ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
10603      :      ;INTO THE F/F'S ON SSRB.
10604      :      D45.00      (004)
10605      :      * D10.30      (122)
10606      :      * D10.60      (177)
10607      :      * MFP.00      (066)
10608      :      MFP.10      (250)
10609      :      SVC.80      (222)
10610      :      SVC.90      (300)
10611      :      FET.00      (217)
10612 074426 012737 074434 001112      :      MOV      #16$,SLPERR      ;SET LOOP ON ERROR POINTER TO 16$
10613 074434 012737 170340 177776 16$:      MOV      #170340,PSW      ;MAKE PREVIOUS MODE USER
10614 074442 012702 100002      :      MOV      #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
10615 074446 000240      :      NOP      ;THIS IS A SYNC POINT FOR SCOPING
10616 074450 006542      :      MFPI     -(R2)      ;READ FROM PHYSICAL 100000
10617 074452 012601      :      MOV      (USP)+,R1      ;POP USER STACK INTO R1
10618 074454 020001      :      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
10619 074456 001401      :      BEQ      7$,          ;BRANCH IF CORRECT DATA WAS FETCHED
10620 074460 104116      :      ERROR   116      ;WRONG DATA WAS FETCHED
10621 074462 012737 032226 000250 7$:      MOV      #MTRAP,MVEC      ;SET M.M.VECTOR TO NORMAL ROUTINE
10622 074470 012737 074172 001112      :      MOV      #20$,SLPERR      ;SET LOOP POINTER TO START OF TEST
10623 074476 042737 000001 172516      :      BIC      #BIT0,MPR3      ;DISABLE USER D-SPACE
10624 074504 012737 000340 177776      :      MOV      #340,PSW      ;MAKE PRESENT MODE KERNEL
10625 074512 000413      :      BR      21$,          ;BRANCH TO EXIT
10626
10627
10628 074514 013737 177572 001250 10$:      MOV      MPR0,MPR0      ;SAVE MPR0 FOR ERROR TYPEOUT
10629 074522 013737 177574 001252      :      MOV      MPR1,MPR1      ;SAVE MPR1 FOR ERROR TYPEOUT
10630 074530 013737 177576 001254      :      MOV      MPR2,MPR2      ;SAVE MPR2 FOR ERROR TYPEOUT
10631 074536 104117      :      ERROR   117      ;TRIED TO READ NON-RESIDENT PAGE
10632 074540 000002      :      RTI      ;RETURN TO TEST
10633
10634 074542 005037 177572      21$:      CLR      @MPR0      ;DISABLE KT
10635 074546 005037 172516      :      CLR      @MPR3
10636
10637
10638
10639      :*****
10640      :*TEST 116      CHECK DPARS READ BACK CORRECTLY
10641      :*
10642      :*      DATA ERRORS MAY OCCUR WHEN A PAR,PD%, OR MEMORY MANAGEMENT
10643      :*      REGISTER TO BE READ IS ADDRESSED USING MEMORY MANAGEMENT.
10644      :*      DATA FROM AN INTERNAL REGISTER (SYS SIZE-LO, SYS SIZE-HI, SYS
```

10645
 10646
 10647
 10648
 10649
 10650
 10651
 10652
 10653 074552 000004
 10654 074554 013746 177776
 10655 074560 042716 000020
 10656 074564 012746 074572
 10657 074570 000002
 10658 074572
 10659 074572 012737 075330 001316
 10660 074600
 10661 074600 005037 177572
 10662 074604 005037 172516
 10663 074610 005037 177776
 10664 074614 012737 001014 177746
 10665 074622 005037 172340
 10666 074626 012737 177406 172300
 10667 074634 012737 000200 172342
 10668 074642 012737 177406 172302
 10669 074650 012737 000400 172344
 10670 074656 012737 177406 172304
 10671 074664 012737 000600 172346
 10672 074672 012737 177406 172306
 10673 074700 012737 001000 172350
 10674 074706 012737 177406 172310
 10675 074714 012737 001200 172352
 10676 074722 012737 177406 172312
 10677 074730 012737 001400 172354
 10678 074736 012737 177406 172314
 10679 074744 012737 007600 172356
 10680 074752 012737 177406 172316
 10681 074760 005037 172360
 10682 074764 012737 177406 172320
 10683 074772 012737 000200 172362
 10684 075000 012737 177406 172322
 10685 075006 012737 000400 172364
 10686 075014 012737 177406 172324
 10687 075022 012737 000600 172366
 10688 075030 012737 177406 172326
 10689 075036 012737 001000 172370
 10690 075044 012737 177406 172330
 10691 075052 012737 001200 172372
 10692 075060 012737 177406 172332
 10693 075066 012737 001400 172374
 10694 075074 012737 177406 172334
 10695 075102 012737 007600 172376
 10696 075110 012737 177406 172336
 10697 075116 012737 000014 172516
 10698 075124 012737 075312 075326
 10699 075132 022737 075326 075326
 10700 075140 101461

```

    ** ID, OR CPU ERROR) MAY BE ENABLED ONTO THE INTERNAL DATA BUS AT
    ** THE SAME TIME AS DATA FROM THE ADDRESSED PAR, PDR, OR
    ** MEMORY MANAGEMENT REGISTER. SYMPTOM: INCORRECT DATA READ
    ** FROM A PAR, PDR, OR MEMORY MANAGEMENT REGISTER WHEN
    ** ADDRESSED USING MEMORY MANAGEMENT. CORRECTION: INSTALL
    ** ECO M8140-00002.
    *****
    TST116: SCOPE
    MOV @MPS, -(SP)
    BIC #20, (SP)
    MOV #BS, -(SP)
    RTI
    8$:
    MOV #DONE, NXTTST ;POINT TO NEXT TEST
    20$:
    CLR MMR0 ;SET CONDITIONS FOR THIS TEST
    CLR MMR3 ;MEM MAN OFF
    CLR PS ;KERNAL MODE
    MOV #1014, @#CONTRL ;DISABLE CACHE
    CLR KIPAR0 ;KI VIRTUAL=PHYSICAL
    MOV #177406, KIPDR0
    MOV #200, KIPAR1
    MOV #177406, KIPDR1
    MOV #400, KIPAR2
    MOV #177406, KIPDR2
    MOV #600, KIPAR3
    MOV #177406, KIPDR3
    MOV #1000, KIPAR4
    MOV #177406, KIPDR4
    MOV #1200, KIPAR5
    MOV #177406, KIPDR5
    MOV #1400, KIPAR6
    MOV #177406, KIPDR6
    MOV #7600, KIPAR7
    MOV #177406, KIPDR7
    CLR KDPAR0 ;KD VIRTUAL=PHYSICAL
    MOV #177406, KDPDR0
    MOV #200, KDPAR1
    MOV #177406, KDPDR1
    MOV #400, KDPAR2
    MOV #177406, KDPDR2
    MOV #600, KDPAR3
    MOV #177406, KDPDR3
    MOV #1000, KDPAR4
    MOV #177406, KDPDR4
    MOV #1200, KDPAR5
    MOV #177406, KDPDR5
    MOV #1400, KDPAR6
    MOV #177406, KDPDR6
    MOV #7600, KDPAR7
    MOV #177406, KDPDR7
    MOV #14, MMR3 ;ENABLE DPARS
    MOV #MAGIC, WORK ;BEGIN WITH USER DPARS
    CMP #WORK, WORK ;TESTED ALL THE DPARS?
    BLOS 7$ ;BRANCH IF YES
    
```

```

10701 075142 012737 000001 177572 MOV #1, MMR0 ;TURN ON MEM MAN
10702 075150 017703 000152 MOV @WORK, R3 ;GET ADDRESS OF FIRST DPAR
10703 075154 017700 000146 MOV @WORK, R0
10704 075160 017701 000142 MOV @WORK, R1
10705 075164 062701 000016 ADD #16, R1 ;POINT TO LAST DPAR
10706 075170 062737 000002 075326 ADD #2, WORK ;POINT TO MAGIC NUMBER
10707 075176 017702 000124 MOV @WORK, R2 ;GET THE MAGIC NUMBER
10708 075202 020103 2$: CMP R1, R3 ;ARE ALL DPARS LOADED?
10709 075204 101402 BLOS 3$, ;BRANCH IF YES
10710 075206 010223 MOV R2, (R3)+ ;LOAD DPAR USING KDPAR7
10711 075210 000774 BR 2$, ;DO DPAR0 THRU DPAR6
10712 075212 010003 3$: MOV R0, R3 ;POINT BACK TO DPAR0
10713 075214 020103 4$: CMP R1, R3 ;ARE ALL DPARS READ?
10714 075216 101421 BLOS 5$, ;BRANCH IF YES
10715 075220 020213 CMP R2, (R3) ;READ THE MAGIC NUMBER BACK
10716 075222 001425 BEQ 6$, ;BRANCH IF OK
10717 075224 011301 MOV (R3), R1 ;GET THE RECEIVED DATA
10718 075226 005037 177572 CLR MMR0 ;TURN OFF MEM MAN
10719 075232 010137 001176 MOV R1, STMP2 ;GET THE RECEIVED DATA
10720 075236 010337 001172 MOV R3, STMP0 ;GET THE DPAR ADDRESS
10721 075242 010237 001174 MOV R2, STMP1 ;GET THE MAGIC NUMBER
10722 075246 005037 172516 CLR MMR3 ;DISABLE DPAR
10723 075252 104145 ERROR 145 ;DPAR DID NOT READ CORRECTLY
10724 075254 104400 014365 TYPE ,ECO ;TYPE ECO MESSAGE
10725 075260 000423 BR DONE ;REPORT ONLY FIRST ERROR
10726 075262 005037 177572 5$: CLR MMR0 ;TURN OFF MEM MAN
10727 075266 062737 000002 075326 ADD #2, WORK ;POINT TO NEXT PARO
10728 075274 000716 BR 1$,
10729 075276 062703 000002 6$: ADD #2, R3 ;POINT TO NEXT DPAR
10730 075302 000744 BR 4$,
10731 075304 005037 172516 7$: CLR MMR3 ;DISABLE DPARS
10732 075310 000407 BR DONE
10733
10734 075312 177660 MAGIC: .WORD UDPAR0 ;DPARS AND MAGIC NUMBER TABLE
10735 075314 007601 .WORD 7601
10736 075316 172260 .WORD SDPAR0
10737 075320 007655 .WORD 7655
10738 075322 172360 .WORD KDPAR0
10739 075324 007654 .WORD 7654
10740 075326 000000 WORK: .WORD 0 ;WORK LOCATION
10741
10742 075330 DONE:
10743
10744
10745 .SBTTL MEMORY MANAGEMENT SETUP
10746
10747
10748
10749
10750 MMSSET:
10751 075330 012703 172340 MOV #KIPAR0,R3 ;GET ADDRESS OF KIPAR0
10752 075334 012704 172300 MOV #KIPDR0,R4
10753 075340 005005 CLR R5
10754 075342 010300 4$: MOV R3,R0 ;GET ADDRESS OF KIPAR0 OR SIPAR0
10755 075344 012720 000000 MOV #0,(R0)+ ;SETUP
10756 075350 012720 000200 MOV #200,(R0)+ ;THE

```

```
10757 075354 012720 000400      MOV      #400,(R0)+      ;KERNEL OR SUPERVISOR
10758 075360 012720 000600      MOV      #600,(R0)+      ;PAR'S
10759 075364 012720 001000      MOV      #1000,(R0)+     ;TO MAP
10760 075370 012720 001200      MOV      #1200,(R0)+     ;THE PROGRAM
10761 075374 012720 001400      MOV      #1400,(R0)+     ;TO
10762 075400 012710 177600      MOV      #177600,(R0)    ;ITSELF
10763 075404 010400              MOV      R4,R0           ;GET ADDRESS OF KIPDRO OR SIPDRO
10764 075406 012701 077406      MOV      #77406,R1       ;GET PDR DATA
10765 075412 012702 000010      MOV      #10,R2          ;SETUP SOB COUNT
10766 075416 010120 2S:      MOV      R1,(R0)+        ;INSTRUCTION TO SETUP PDR'S
10767 075420 077202              SOB      R2,2S           ;EXECUTE EIGHT TIMES
10768 075422 005705              TST      R5              ;IS SETUP COMPLETE?
10769 075424 001006              BNE     TST117           ;BRANCH IF YES
10770 075426 012703 172240      MOV      #SIPARO,R3      ;GET ADDRESS OF SUPER PARO
10771 075432 012704 172200      MOV      #SIPDRO,R4      ;GET ADDRESS OF SUPER PDR0
10772 075436 005205              INC     R5              ;SET PASS COUNT
10773 075440 000740              BR      4S              ;GO SETUP SUPER PAR'S AND PDR'S
```

```
*****
*TEST 117      MEMORY MANAGEMENT ABORT
*
* THIS TEST SETS UP PDR6 TO CAUSE A MEMORY MANAGEMENT ABORT.
* IF TMCC AERF(1) DOES NOT GO HIGH IT WILL LOOK LIKE THE TRAP FAILED.
*
* IF TMCC ABORT DOES NOT GO HIGH THE TEST WILL NOT TRAP AT ALL.
* IF TMCB SEGT DOES NOT GO LOW A TRAP TO 4 WILL OCCUR.
*
* IF TMCE CACHE BEND DOES NOT GO HIGH A TRAP TO 350 WILL OCCUR.
*****
```

```
10785 075442 000004      TST117: SCOPE
10786 075444 012737 075636 001210      MOV      #TST120,$ESCAPE ;SAVE START ADDRESS OF NEXT TEST
10787 075452 012737 075636 001316      MOV      #TST120,NXTTST  ;SAVE START ADDRESS OF NEXT TEST
10788 075460 012737 075574 000004      MOV      #3S,@ERRVEC     ;SETUP LOCATION 4
10789 075466 012737 000340 000006      MOV      #PR7,@ERRVEC+2  ;RESTOR ERROR VEC PSW
10790 075474 012737 075624 000250      MOV      #4S,@ERRVEC     ;SETUP ABORT VECTOR
10791 075502 012737 075602 000010      MOV      #5S,@RESVEC     ;SETUP LOCATION 10
10792 075510 012737 000340 000012      MOV      #PR7,@RESVEC+2  ;RESTORE RESVEC PSW
10793 075516 012737 075610 000240      MOV      #6S,@240        ;SETUP LOCATION 240
10794 075524 012737 075616 000350      MOV      #7S,@350        ;SETUP LOCATION 350
10795 075532 012737 075546 001112      MOV      #1S,$LPERR      ;SETUP ERROR LOOP
10796 075540 012737 077401 172314      MOV      #77401,@KIPDR6  ;MAKE PAGE 6 CAUSE ABORT
10797 075546 012706 001100 1S:      MOV      #STACK,SP       ;SETUP THE SP
10798 075552 012737 000001 177572      MOV      #BIT0,@MMRO     ;TURN RELOCATION ON
10799 075560 012737 177777 140000      MOV      #-1,@140000     ;EXECUTE ABORT INSTRUCTION
10800 ;ABORT FAILED
10801 075566 005037 177572      CLR     @MMRO            ;TURN RELOCATION OFF
10802 075572 104146      ERROR   146             ;NO KT ABORT
10803 ;TRAPPED TO LOCATION 4
10804 075574 005037 177572 3S:      CLR     @MMRO            ;TURN OFF MM
10805 075600 104150      ERROR   150             ;TRAPPED TO 4
10806 075602 005037 177572 5S:      CLR     @MMRO            ;TURN OFF MM
10807 075606 104151      ERROR   151             ;TRAPPED TO 10
10808 075610 005037 177572 6S:      CLR     @MMRO            ;TURN OFF MM
10809 075614 104152      ERROR   152             ;TRAPPED TO 240
10810 075616 005037 177572 7S:      CLR     @MMRO            ;TURN RELOCATION OFF
10811 075622 104166      ERROR   166
10812 075624 005037 177572 4S:      CLR     @MMRO            ;TURN MM OFF
```


10813 075630 012737 000012 000010
 10814
 10815
 10816
 10817
 10818
 10819
 10820
 10821
 10822
 10823
 10824
 10825
 10826
 10827
 10828
 10829 075636 000004
 10830 075640 012737 075776 001210
 10831 075646 012737 075776 001316
 10832 075654 012737 026252 000004
 10833 075662 012737 075730 000250
 10834 075670 012737 075704 001112
 10835 075676 012737 077404 172314
 10836 075704 012706 001100
 10837 075710 012737 001001 177572
 10838 075716 005037 140000
 10839
 10840 075722 005037 177572
 10841 075726 104153
 10842
 10843 075730 012737 075772 000250
 10844 075736 012737 075744 001112
 10845 075744 012706 001100
 10846 075750 012737 001001 177572
 10847 075756 013737 140000 001172
 10848
 10849 075764 005037 177572
 10850 075770 104155
 10851
 10852 075772 005037 177572
 10853
 10854
 10855
 10856
 10857
 10858
 10859
 10860
 10861
 10862
 10863
 10864 075776 000004
 10865 076000 012737 076156 001316
 10866 076006 005037 177766
 10867 076012 022737 167777 177760
 10868 076020 003456

```

MOV #12,@PRESVEC ;CONTINUE
:*****
*TEST 120 MEMORY MANAGEMENT TRAP
*
* THIS TEST ENSURES THAT THE MEMORY MANAGEMENT TRAP LOGIC WORKS.
* IF TMCA HONOR SEGTF DOES NOT GO LOW OR DOES NOT GET THRU
* TO TMCB BRQ TRUE THE TRAP WILL NOT OCCUR.
*
* IF TMCB SEGT DOES NOT GO LOW BEN13 WILL FAIL TO BRK.20.
* THE FLOW WILL THEN GO TO RTI.60 WHICH MEANS AN ACKNOWLEDGE
* IS NEVER GIVEN AND THE PROCESSOR WILL HANG JF.
*
* AN INSTRUCTION IS THEN EXECUTED THAT CAUSES A MEMORY MANAGEMENT
* TRAP ON THE SOURCE OPERAND BUT NOT ON THE DESTINATION.
:*****
TST120: SCOPE
MOV #TST121,SESCAPE ;SAVE START ADDRESS OF NEXT TEST
MOV #TST121,NXTTST ;SAVE START ADDRESS OF NEXT TEST
MOV #CPUSPUR,@ERRVEC ;RESTORE LOCATION 4
MOV #2$,@PERRVEC ;SETUP LOCATION 250
MOV #3$,SLPERR ;SETUP ERROR LOOP
MOV #77404,@KIPDR6 ;SET ACF 4 IN PDR6
3$: MOV #STACK,SP ;INITIALIZE THE SP
MOV #1001,@MMRO ;TURN RELOCATION ON
CLR @140000 ;EXECUTE TRAP TYPE INSTRUCTION
:NO TRAP
CLR @MMRO ;TURN RELOCATION OFF
ERROR 153 ;NO KT TRAP
:TRAP OK, NOW TRY TRAP ON SOURCE NO TRAP ON DESTINATION
2$: MOV #4$,@PERRVEC ;SETUP VECTOR
MOV #5$,SLPERR ;SETUP ERROR LOOP
5$: MOV #STACK,SP ;INITIALIZE THE SP
MOV #1001,@MMRO ;INITIALIZE MMRO
MOV @140000,$TMPO ;EXECUTE TRAP ON SOURCE
:NO TRAP
CLR @MMRO ;TURN RELOCATION OFF
ERROR 155 ;NO TRAP ON SOURCE
:TRAP OK
4$: CLR @MMRO ;TURN RELOCATION OFF
;CONTINUE
:*****
*TEST 121 NON EXISTANT MEMORY ABORT
*
* THIS TEST ENSURES THAT A NON EXISTANT MEMORY
* REFERENCE FUNCTIONS PROPERLY.
*
* IF TMCC AERF(1) DOES NOT GO HIGH IT WILL LOOK LIKE THE ABORT FAILED.
* IF TMCC ABORT DOES NOT GO HIGH THE ABORT WILL STILL OCCUR,
* BUT THE ERROR REGISTER WILL NOT BE LOADED.
:*****
TST121: SCOPE
MOV #TST122,NXTTST ;SAVE ADDRESS OF NEXT TEST
CLR @CPUERR ;CLEAR ERROR REG
CMP #167777,@SIZELO;2 MILLION WORDS ON SYSTEM?
BLE TST122 ;BRANCH IF YES
    
```

```

10869 076022 012737 076052 001112      MOV      #1$,SLPERR      ;SETUP ERROR LOOP
10870 076030 012737 076104 000004      MOV      #3$,@ERRVEC    ;SETUP LOCATION 4
10871 076036 013737 177760 172354      MOV      @SIZELO,@KIPAR6 ;SETUP PAR6
10872 076044 012737 077406 172314      MOV      #77406,@KIPDR6 ;PUT 6 IN PDR6 ACF FIELD
10873 076052 012706 001100      1$:      MOV      #STACK,SP      ;SETUP THE SP
10874 076056 012737 000020 172516      MOV      #20,@MPR3      ;SETUP FOR 22 BIT MODE
10875 076064 012737 000001 177572      MOV      #BIT0,@MPRO     ;TURN RELOCATION ON
10876 076072 005037 140100      CLR      @#140100       ;MAKE REFERENCE TO NEXM
10877
10878 076076 005037 177572      ;NO ABORT
10879 076102 104156      CLR      @MPRO          ;TURN RELOCATION OFF
                                ERROR 156      ;NO ABORT ON NEXM
10880
10881 076104 005037 177572      ;ABORT OK
10882 076110 022737 000040 177766      3$:      CLR      @MPRO          ;TURN RELOCATION OFF
                                CMP      #BIT5,@CPUERR  ;IS ERROR REGISTER OK?
10883 076116 001411      BEQ      4$            ;BRANCH IF YES
10884 076120 013737 177766 001174      MOV      @CPUERR,$TMP1  ;SAVE ERROR REG FOR TYPEOUT
10885 076126 012737 000040 001172      MOV      #BIT5,$TMP0    ;SAVE EXPECTED VALUE
10886 076134 005037 177766      CLR      @CPUERR       ;CLEAR ERROR REG
10887 076140 104160      ERROR 160      ;NEXM BIT DID NOT SET IN CPU ERROR
10888 076142 005037 177766      4$:      CLR      @CPUERR       ;CLEAR ERROR REG
10889 076146 005737 177766      TST      @CPUERR       ;DID REGISTER CLEAR?
10890 076152 001401      BEQ      TST122        ;:BRANCH IF YES
10891 076154 104161      ERROR 161      ;NEXM BIT DID NOT CLEAR
10892
10893      ;*****
10894      ;*TEST 122      KT BEND
10895
10896      ;*
10897      ;*      THIS TEST ENSURES THAT TMCE KT BEND GOES LOW ON AN ODD
10898      ;*      ADDRESS ERROR, SL RED, AND NEXM. THIS IS DONE BY EXECUTING
10899      ;*      AN INSTRUCTION FOR EACH OF THESE THREE CASES THAT ALSO
10900      ;*      CAUSES A KT ABORT. THE ABORT SHOULD NOT BE HONORED.
10901
10902      ;*
10903      ;*      NOTE: ON A KB11-E OR KB11-EM THE KT ABORT SHOULD BE HONORED OVER A
10904      ;*      NEXM TRAP. IF THIS IS A KB11-E/EM THEN THIS FEATURE IS TESTED.
10905      ;*
10906      ;*****
10907      TST122: SCOPE
10908      MOV      #TST123,NXTTST ;SAVE ADDRESS OF NEXT TEST
10909      CMP      #167777,@SIZELO ;2 MILLION WORDS ON SYSTEM?
10910      BLE      3$          ;BRANCH IF YES
10911      MOV      #1$,SLPERR    ;SETUP ERROR LOOP
10912      MOV      #2$,@ERRVEC    ;SETUP MEMORY VECTOR
10913      MOV      #3$,@ERRVEC    ;SETUP LOCATION 4
10914      TST      @KB11E        ;IS THIS A KB11-E OR KB11-EM?
10915      BEQ      20$         ;BR IF NOT
10916      MOV      #3$,@ERRVEC    ;SETUP MEMORY VECTOR
10917      MOV      #21$,ERRVEC    ;SETUP LOCATION 4
10918      20$:      BIS      #7,@KIPDR6    ;SET ACF FIELD TO 7 IN PDR6
10919      MOV      #60,@MPR3      ;SETUP MPR3
10920      1$:      MOV      #STACK,SP    ;INITIALIZE THE SP
10921      MOV      #BIT0,@MPRO     ;TURN RELOCATION ON
10922      CLR      @#140100       ;MAKE A REFERENCE TO NEXM
10923      ;FAILURE, KT ABORT CAME IN (KB11-B/C)
10924      2$:      CLR      @MPRO          ;TURN RELOCATION OFF
                                ERROR 162      ;KT ABORT OCCURRED ON NEXM
                                BR      3$          ;SKIP KB11-E ERROR MESSAGE
10925      ;FAILURE, KT ABORT OVER-RIDDEN BY NEXM (KB11-E/EM ONLY)
10926      21$:     CLR      @MPRO          ;TURN RELOCATION OFF
    
```

```

10925 076310 104165          ERROR 165
10926          :NEXM OK, TRY SL RED
10927 076312 005037 172516 3$: CLR @MMR3 ;GO BACK TO 18 BIT MODE
10928 076316 052737 000007 172314 BIS #7,@KIPDR6 ;MAKE KERNEL PAGE 6 NON RESIDENT
10929 076324 012737 076370 000250 MOV #5$,@ERRVEC ;SETUP MEM VECTOR
10930 076332 012737 076370 000004 MOV #5$,@ERRVEC ;SETUP LOCATION 4
10931 076340 012737 076346 001112 MOV #6$,SLPERR ;SETUP ERROR LOOP
10932 076346 012737 000001 177572 6$: MOV #BIT0,@MMRO ;TURN RELOCATION ON
10933 076354 012706 140336 MOV #140336,KSP ;PUT SP IN RED ZONE PAGE 6
10934 076360 012737 140000 177774 MOV #140000,@STKLMT ;SET STACK BOUNDARY AT 24K + 400
10935 076366 005016          CLR (SP) ;EXECUT INSTRUCTION TO RED ZONE NON-RESIDENT
10936 076370 032737 100000 177572 5$: BIT #BIT15,@MMRO ;DID KT ABORT FLAG COME ON?
10937 076376 001407          BEQ 10$ ;BRANCH IF NO
10938 076400 005037 177572          CLR @MMRO ;TURN RELOCATION OFF
10939 076404 005037 177774          CLR @STKLMT ;
10940 076410 012706 001100          MOV #STACK,SP ;RESTORE THE SP
10941 076414 104163          ERROR 163
10942          :SL RED OK, TRY ODD ADDRESS
10943 076416 005037 177774 10$: CLR @STKLMT ;CLEAR THE SL REG
10944 076422 012737 076462 000250 MOV #7$,@ERRVEC ;SETUP MM VECTOR
10945 076430 012737 076470 000004 MOV #8$,@ERRVEC ;SETUP LOCATION 4
10946 076436 012737 076444 001112 MOV #9$,SLPERR ;SETUP ERROR LOOP
10947 076444 012737 000001 177572 9$: MOV #BIT0,@MMRO ;TURN RELOCATION ON
10948 076452 012706 001100          MOV #STACK,SP ;INITIALIZE THE SP
10949 076456 005037 140001          CLR @140001 ;EXECUTE ODD ADDRESS AND KT ABORT
10950          :FAILURE, KT ABORT CAME IN
10951 076462 005037 177572 7$: CLR @MMRO ;TURN RELOCATION OFF
10952 076466 104164          ERROR 164 ;KT ABORT OCCURRED ON ODD ADDRESS
10953          :ODD ADDRESS OK
10954 076470 005037 177572 8$: CLR @MMRO ;TURN RELOCATION OFF
10955 076474 005037 177766          CLR @CPUERR ;ENSURE ERROR REG CLEAR
10956          ;CONTINUE
10957          :*****
10958          :*TEST 123 SL REGISTER COMPARATOR TEST 2
10959          :*
10960          :* THIS TEST IS THE SAME AS TEST 153 EXCEPT IT TESTS THE ADDRESSES
10961          :* ON EVERY PAGE. THIS IS DONE BY MAPPING I/O PAGE ADDRESSES TO
10962          :* MEMORY IN KERNEL MODE. THIS MAKES THE I/O PAGE INACCESSABLE
10963          :* IN KERNEL MODE SO AN IOT INSTRUCTION IS USED TO RETURN TO
10964          :* SUPERVISOR MODE WHEN THE I/O PAGE IS NEEDED.
10965          :*****
10966 076500 000004          TST123: SCOPE
10967 076502 012737 077242 001316 MOV #TST124,NXTTST ;SAVE ADDRESS OF NEXT TEST
10968 076510 005037 001176          CLR $TMP2 ;CLEAR BUFFER OVERFLOW FLAG
10969 076514 012737 003706 001106 MOV #D1990,$ICNT ;SETUP ITERATION COUNT
10970 076522 012737 076530 001110 MOV #23$,SLPADR ;SETUP LOOP ADDRESS
10971 076530 012737 077214 001210 23$: MOV #22$,SESCAPE ;SETUP ESCAPE
10972 076536 012737 077210 000020 MOV #12$,@IOTVEC ;SETUP THE IOT VECTOR
10973 076544 012737 040340 000022 MOV #40340,@IOTVEC+2 ;SETUP THE IOT VEC PSW
10974 076552 012737 077406 172314 MOV #77406,@KIPDR6 ;ENSURE PDR6 OK
10975 076560 012737 001400 172354 MOV #1400,@KIPAR6 ;ENSURE PAR6 OK
10976 076566 012737 076712 000004 MOV #1$,@ERRVEC ;SETUP ERROR VECTOR
10977 076574 012737 040340 000006 MOV #40340,@ERRVEC+2 ;SETUP ERRVEC PSW
10978 076602 012737 001600 172356 MOV #1600,@KIPAR7 ;MAP KERNEL PAGE 7 TO 28x
10979 076610 052737 040000 177776 BIS #BIT14,@PSW ;GO TO SUPER MODE
10980 076616 012737 000001 177572 MOV #BIT0,@MMRO ;TURN RELOCATION ON
    
```



```

11037 077042 001011      BNE 5$          ;BRANCH IF YES
11038 077044 005200      INC R0          ;SET POINTER TO HIGH BYTE
11039 077046 113720 177775  MOVB @STKLMT+1,(R0)+ ;SAVE ERROR STACK LIMIT
11040 077052 010520      MOV R5,(R0)+   ;SAVE ERROR SP
11041 077054 020027 157774  CMP R0,#157774 ;HAS BUFFER REACHED PAGE ??
11042 077060 001002      BNE 5$          ;BRANCH IF NO
11043 077062 005237 001176  INC $TMP2      ;SET BUFFER OVERFLOW FLAG
11044
11045      ;CONTINUE TEST
11046 077066 062705 000400  5$: ADD #400,R5    ;GO TO NEXT STACK ADDRESS
11047 077072 000004      IOT           ;GO TO SUPERVISOR MODE
11048 077074 012706 001100  21$: MOV #STACK,SSP ;RESET THE SSP
11049 077100 005037 177766  CLR @CPUERR   ;CLEAR ERROR REGISTER
11050 077104 105037 177777  CLRB @PSW+1   ;GO BACK TO KERNEL
11051 077110 005302      DEC R2        ;REPLACES A
11052 077112 001264      BNE 3$        ;SOB
11053 077114 000004      IOT           ;GO TO SUPERVISOR MODE
11054 077116 062701 000400  16$: ADD #400,R1    ;SET NEXT YELLOW ZONE ADDRESS
11055 077122 062737 000400 177774  ADD #400,@STKLMT ;GO TO NEXT SL ADDRESS
11056 077130 105037 177777  CLRB @PSW+1   ;GO BACK TO KERNEL
11057 077134 005303      DEC R3        ;THIS REPLACES
11058 077136 001246      BNE 2$        ;A SOB
11059
11060      ;DONE WITH TEST. WAS THERE AN ERROR?
11061 077140 000004      IOT           ;GO TO SUPERVISOR MODE
11062 077142 012737 177600 172356  MOV #177600,@KIPAR7 ;RESTORE KERNEL I/O PAGE
11063 077150 105037 177777  CLRB @PSW+1   ;GO TO KERNEL MODE
11064 077154 005037 177774  18$: CLR @STKLMT   ;RESET THE SL REG
11065 077160 012706 001100  MOV #STACK,SP ;AND SP
11066 077164 012737 026252 000004  MOV #CPUSPUR,@ERRVEC ;RESTORE ERRVEC
11067 077172 005737 120002  TST @#120002  ;WAS THERE AN ERROR?
11068 077176 001406      BEQ 22$       ;BRANCH IF NO
11069 077200 010037 001156  MOV R0,$REGO  ;SAVE ERROR DATA POINTER
11070 077204 104167      ERROR 167     ;STACK LIMIT COMPARATORS FAILED
11071 077206 000402      BR 22$
11072
11073 077210 000176 000000  ;IOT ROUTINE
11074      12$: JMP @($SSP) ;RETURN IN SUPER MODE
11075
11076      ;TEST FINISHED, CLEAN UP VECTORS
11077 077214 005037 177572  22$: CLR @PRO      ;TURN RELOCATION OFF
11078 077220 012737 025474 000020  MOV #SCOPE,@IOTVEC ;RESTORE IOT VECTOR
11079 077226 012737 000340 000022  MOV #PR7,@IOTVEC+2
11080 077234 012737 000340 000006  MOV #PR7,@ERRVEC+2
11081
11082      ;*****
11083      ;*TEST 124 PS RESTORE
11084      ;*
11085      ;* THIS TEST ENSURES THAT BEN6 WORKS ON A PS RESTORE.
11086      ;* THIS IS DONE BY SETTING THE STACK TO A NON RESIDENT PAGE
11087      ;* AND DOING A TRAP INSTRUCTION. WHEN THE PROCESSOR TRYS TO
11088      ;* PUSH THE OLD PSW ON THE STACK A KT ABORT WILL OCCUR.
11089      ;* SINCE IT WAS A KERNEL R6 OPERATION THIS WILL CAUSE BEN13
11090      ;* TO GO TO STATE SER.00 WHICH WILL PUSH THE PSW AND PC INTO
11091      ;* LOCATIONS 2 AND 0, AND THEN TRAP TO LOCATION 4.
11092      ;* THE PSW IN LOCATION 2 SHOULD BE THE PSW BEFORE THE
11093      ;* TRAP INSTRUCTION AND NOT THE PSW IN THE TRAP VECTOR.
    
```

```

11093
11094 077242 000004
11095 077244 012737 077346 001316
11096 077252 012737 077324 000004
11097 077260 012737 000340 000036
11098 077266 042737 000007 172314
11099 077274 012737 077302 001112
11100 077302 005037 000002
11101 077306 012737 000001 177572
11102 077314 012706 150004
11103 077320 000236
11104 077322 104400
11105 077324 005037 177572
11106 077330 012706 001100
11107 077334 022737 000310 000002
11108 077342 001401
11109 077344 104170
11110 077346
11111
11112
11113 077346 000004
11114 077350 000240
11115
11116 077352 000137 025204
11117
11118 077356
11119 077354
11120 077360
11121 077360
11122
11123 077360 001000
11124 000001

:*****
TST124: SCOPE
MOV #CACHE,NXTTST
MOV #1$,@ERRVEC ;SETUP ERROR VECTOR
MOV #PR7,@TRAPVEC+2 ;ENSURE PR7 IN LOCATION 36
BIC #7,@PKIPDR6 ;MAKE PAGE 6 NON-RESIDENT
MOV #2$,SLPERR ;SETUP ERROR LOOP
2$: CLR @#2 ;ENSURE LOCATION 2 CLEAR
MOV #BIT0,@MMRO ;TURN RELOCATION ON
MOV #150004,SP ;SETUP THE SP
SPL 6 ;SET CPU PRIORITY AT 6
TRAP ;EXECUTE TRAP INSTRUCTION
1$: CLR @MMRO ;TURN RELOCATION OFF
MOV #STACK,SP ;RETSORE SP
CMP #310,@#2 ;DID CORRECT PSW GET STACKED?
BEQ CACHE ;:BRANCH IF YES
ERROR 170 ;BENG FAILED ON PS RESTORE

CACHE:
:*****
END: SCOPE ;LOOP BACK FOR LAST TEST
NOP ;THIS CAN BE ANYTHING YOU WANT
; (AS LONG AS IT IS ONLY ONE WORD)
JMP $EOP ;JUMP TO END-OF-PASS ROUTINE

TSLOC=. ;GET PC TO AN EVEN WORD BOUNDARY
TSLOC=-4&TSLOC
TSLOC=TSLOC+4
.=TSLOC

TSTDAT: .BLKW 512.
.END
    
```


EM131	013335	1185	2154#
EM132	013416	1191	2163#
EM133	013462	1197	2169#
EM134	013530	1202	2176#
EM135	013572	1208	2182#
EM136	013632	1214	2188#
EM137	013763	1221	2203#
EM14	004503	713	1529#
EM140	014112	1227	2219#
EM141	014160	1233	2226#
EM142	014222	1239	2232#
EM143	014270	1245	2239#
EM145	014332	1257	2245#
EM146	014461	1263	2261#
EM147	014637	1269	2280#
EM15	004537	719	1534#
EM150	014671	1275	2285#
EM151	014775	1281	2297#
EM152	015044	1287	2304#
EM153	015110	1293	2310#
EM155	015217	1305	2323#
EM156	015337	1311	2338#
EM16	004613	725	1542#
EM160	015472	1323	2354#
EM161	015600	1329	2366#
EM162	015651	1335	2373#
EM163	015730	1341	2381#
EM164	016005	1347	2389#
EM165	016070	1353	2398#
EM166	016154	1359	2407#
EM167	016231	1365	2416#
EM17	004646	731	1547#
EM170	016425	1371	2439#
EM171	016521	1377	2450#
EM2	003537	649	1440#
EM20	004714	737	1554#
EM201	016546	1384	2454#
EM202	016623	1392	2462#
EM203	016714	1401	2472#
EM21	004755	743	1560#
EM22	005017	750	1566#
EM23	005061	756	1572#
EM24	005125	762	1579#
EM25	005164	768	1585#
EM26	005242	774	1593#
EM27	005310	780	1600#
EM3	003620	655	1449#
EM30	005347	786	1606#
EM31	005436	792	1616#
EM32	005477	798	1622#
EM32M	005606	1251	1634#
EM33	005711	804	1646#
EM34	005770	810	1654#
EM35	006054	816	1663#
EM36	006142	822	1672#
EM37	006211	829	1679#

ER200	003274	1381#	3458	3461													
FCAC =	000400	424#															
FSTTST	001222	550#	4597*	4599*	4601*	4603*	4605*	4607*	4609*	4611*	4691						
FVPE =	002000	422#															
GNC =	***** U	450	3174	3181	3361	3367	3389	3395	3872	3873	3874	3875	3876	3877			
		3878	3879	3880	4555	4559	4563	4572	4580	4668							
HIADRS=	177742	154#	4313														
HITMIS=	177752	158#	439#	4317													
HOLFLG	001304	578#	6627*	6645	6650*	6663	6750	6753*	6758*	6797*	6808	6813*	6826	6909			
		6912*	6917*														
HT =	000011	42#	3663	3701													
IOTVEC=	000020	140#	4622*	4623*	10972*	10973*	11077*	11078*									
IVSS =	040000	418#															
KBTST	032602	4486#	4643														
KB11CM	001362	618#	4486*	4534*	4676	6013	6062	6110									
KB11E	001360	616#	4487*	4491*	4530	4532*	4674	4682	4841	4850	6015	6064	6112	10910			
KB11EM	001361	617#															
KDPARO=	172360	312#	8052*	8191*	8478*	8484*	8661*	10681*	10738								
KDPAR1=	172362	313#	8053*	8662*	10683*												
KDPAR2=	172364	314#	8663*	10685*													
KDPAR3=	172366	315#	8664*	10687*													
KDPAR4=	172370	316#	8054*	8192*	8476*	8665*	10689*										
KDPAR5=	172372	317#	8666*	10691*													
KDPAR6=	172374	318#	8667*	10693*													
KDPAR7=	172376	319#	5298	5515	5530	5889	8056*	8194*	8668*	10695*							
KDPARO=	172320	290#	8047*	8186*	8730	8847	8988	9107	9134	9138*	9141*	9154*	9157*	9174*			
		9177*	9194*	9197*	9253	9273*	9289*	9309*	9329*	9396	9408*	9424*	9444*	9464*			
		10682*															
KDPDR1=	172322	291#	8048*	8187*	9139*	9155*	9175*	9195*	9274*	9290*	9510*	9330*	9409*	9425*			
		9445*	9465*	10684*													
KDPDR2=	172324	292#	8050*	8189*	10686*												
KDPDR3=	172326	293#	8051*	8190*	10688*												
KDPDR4=	172330	294#	8049*	8188*	8475*	9546*	9685*	9831*	9962*	10091*	10237*	10387*	10525*	10690*			
KDPDR5=	172332	295#	10692*														
KDPDR6=	172334	296#	10694*														
KDPDR7=	172336	297#	5394	5702	5717	6023	8055*	8193*	9140*	9142*	9156*	9158*	9176*	9178*			
		9196*	9198*	9275*	9291*	9311*	9331*	9410*	9426*	9446*	9466*	10696*					
KERSTK=	001100	29#	4616	4661	4956	4963	7473	8485	9563	10106	10256						
KERVEC	032456	4445#	7728	7918													
KIPARO=	172340	301#	3563	3934	3968	5294	5496	5498	5512	5513	5527	5880	6156	6157*			
		6160	6165	6169	6422	6437*	6970*	8039*	8541*	8653*	9528*	10665*	10751				
KIPAR1=	172342	302#	6438*	6971*	8040*	8542*	8654*	9529*	10667*								
KIPAR2=	172344	303#	6439*	6972*	8041*	8543*	8655*	9530*	10669*								
KIPAR3=	172346	304#	6440*	6973*	8042*	8544*	8656*	9531*	10671*								
KIPAR4=	172350	305#	2985	6444*	6458*	6472*	6486*	6500*	6514*	6528*	6542*	6556*	6570*	6584*			
		6598*	6630*	6658*	6683*	6707*	6719*	6737	6800*	6822*	6841*	6865*	6877*	6896			
		6978*	7031*	7085*	7138	7161*	7206*	7269*	7324*	7373*	7469*	7516*	8185*	8551*			
		8592*	8657*	9552*	9691*	9837*	9968*	10097*	10243*	10393*	10529*	10673*					
KIPAR5=	172352	306#	6631*	6647	6659*	6661	6752	6801*	6810	6823*	6824	6842*	6911	7086*			
		7128*	7160*	7205*	7268*	7323*	7372*	7470*	8658*	10675*							
KIPAR6=	172354	307#	8659*	10677*	10871*	10975*											
KIPAR7=	172356	308#	3982	6441*	6974*	7414	8043*	8545*	8660*	9532*	10679*	10978*	11015*	11062*			
KIPARO=	172300	279#	4511*	4512	4514*	5390	5683	5685	5699	5700	5714	6008	6261	6262*			
		6265	6270	6274	6420	6434	6965*	8034*	8536*	8707	8734	8738*	8741*	8746			
		8754*	8757*	8774*	8777*	8794*	8797*	8851	8871*	8879	8887*	8907*	8927*	8992			
		9004*	9012	9020*	9040*	9060*	9130	9146	9249	9281	9392	9416	9523*	9539			

TST111	072040	9819	9930	9956#	
TST112	072470	9958	10061	10085#	
TST113	073062	10087	10200	10222#	
TST114	073526	10224	10348	10373#	
TST115	074126	10375	10486	10512#	
TST116	074552	10513	10653#		
TST117	075442	10769	10785#		
TST12	042006	4993	5054#		
TST120	075636	10786	10787	10829#	
TST121	075776	10830	10831	10864#	
TST122	076156	10865	10868	10890	10903#
TST123	076500	10904	10966#		
TST124	077242	10967	11094#		
TST13	042440	5056	5167#		
TST14	042614	5169	5202	5215#	
TST15	042760	5217	5282#		
TST16	043114	5284	5303	5319#	
TST17	043220	5321	5335	5351#	
TST2	040766	4726	4738	4755#	
TST20	043324	5353	5367	5383#	
TST21	043430	5385	5399	5415#	
TST22	043534	5417	5431	5447#	
TST23	043640	5449	5463	5490#	
TST24	044050	5492	5534	5553#	
TST25	044260	5555	5597	5615#	
TST26	044470	5617	5659	5677#	
TST27	044700	5679	5721	5739#	
TST3	041122	4757	4797#		
TST30	045110	5741	5783	5801#	
TST31	045320	5803	5845	5872#	
TST32	045450	5874	5897	5915#	
TST33	045600	5917	5940	5958#	
TST34	045730	5960	5983	6000#	
TST35	046106	6002	6031	6049#	
TST36	046264	6051	6080	6097#	
TST37	046442	6099	6128	6150#	
TST4	041166	4799	4825#		
TST40	046560	6152	6185#		
TST41	046676	6187	6220#		
TST42	047014	6222	6255#		
TST43	047132	6257	6290#		
TST44	047250	6292	6325#		
TST45	047366	6327	6365#		
TST46	047470	6367	6406#		
TST47	051104	6408	6623#		
TST5	041310	4827	4869#		
TST50	051734	6625	6728	6784#	
TST51	052610	6786	6886	6955#	
TST52	053142	6957	7025#		
TST53	053334	7027	7078#		
TST54	053620	7080	7155#		
TST55	053750	7157	7200#		
TST56	054214	7202	7263#		
TST57	054434	7265	7318#		
TST6	041366	4871	4897#		
TST60	054632	7320	7367#		

Symbol	Value	Value	Value	Value	Value	Value	Value	Value	Value	Value	Value	Value	Value	Value
SOVER	025742	3238	3256	3264	3274	3282#	3270	3286						
SPASS	001100	506#	3164*	3165*	3176	3216								
SPWRAD	030512	3916#												
SPWRDN	030370	3887#	3911	4628										
SPWRMG	030506	3914#												
SPWRUP	030436	3896	3901#											
SQUES	001216	547#	3351											
SRDCHR=	*****	U	3877											
SRDEEC=	*****	U	3877											
SRDLIN=	*****	U	3877											
SRDOCT=	*****	U	3877											
SREGAD	001154	530#												
SREGO	001156	532#	2952	2960	2962	2964	2966	2968	2970	2972	2978	2983	2990	3018
		3022	3028	3033	3035	3043	3044	3046	3305*	11069*				
SREG1	001160	533#	2939	2941	2947	2960	2964	2966	2970	2972	2981	2987	2990	3000
		3003	3010	3016	3020	3022	3026	3033	3035	3044	3046	3306*		
SREG2	001162	534#	2941	2943	2945	2948	2952	2960	2966	2970	2972	2978	3012	3020
		3035	3046	3307*										
SREG3	001164	535#	2945	2978	2987	3016	3018	3035	3308*					
SREG4	001166	536#	3035	3046	3309*									
SREG5	001170	537#	3310*											
SRESRE	027406	3617#	3878											
SRTRN	025460	3192	3211#	4634	4636*									
SSAVRE	027350	3601#	3877											
SSAVR6	030522	3895*	3901	3902*	3903*	3920#								
SSCOPE	025474	3234#	4622	11077										
SSETUP=	000037	443#	3162	3328	4664									
SSIZE	030570	3950#	4549											
SSIZEX	031036	3986	3996#											
SSTUP =	177777	443#												
SSVLAD	025714	3246	3277#											
SSVPC =	000240	490#	495											
SSWR =	177400	1#	10	15	16	17	18	19	20	21	22	544	545	546
		3156	3163	3190	3203	3216	3226	3227	3228	3229	3230	3237	3249	3251
		3252	3257	3258	3259	3266	3267	3268	3279	3282	3285	3293	3294	3295
		3296	3297	3298	3314	3317	3324	3328	3334	3351	4726	4759	4801	4829
		4873	4901	4927	4955	4995	5058	5171	5219	5286	5323	5355	5387	5419
		5451	5494	5557	5619	5681	5743	5805	5876	5919	5962	6004	6053	6101
		6154	6189	6224	6259	6294	6329	6363	6410	6627	6788	6959	7029	7082
		7159	7204	7267	7322	7371	7408	7467	7516	7566	7607	7669	7764	7861
		7946	8027	8184	8265	8368	8465	8529	8649	8838	8971	9104	9239	9374
		9516	9683	9821	9960	10089	10226	10377	10513	10654	10786	10830	10865	10904
SSWRK=	000200	1#	22	23	3230	3231	3253	3254						
STBIT	025466	3207*	3216#	4637*										
STIMES	001206	544#	3163*	3266*	3273	3276*	3285	4631*	5876*	5919*	5962*	6004*	6053*	6101*
		6959*	7029*	7946*										
STKB	001142	523#												
STKS	001140	522#	4620											
STMPO	001172	538#	2994	3005	3014	3038	3041	3369*	3379	3397*	3407	5125*	5135	5139*
		5149	5518*	5525*	5581*	5588*	5643*	5650*	5705*	5712*	5767*	5774*	5829*	5836*
		6443*	6455	6457*	6469	6471*	6483	6485*	6497	6499*	6511	6513*	6525	6527*
		6539	6541*	6553	6555*	6567	6569*	6581	6583*	6595	6597*	6609	6642*	6657
		6805*	6820	6845*	6849	7484*	7633*	7642	8556*	8557	8563*	8564	8570*	8571
		8578*	8579	8586*	8587	8593*	8594	8600*	8601	8608*	8609	8712*	8762*	8782*
		8802*	8805	8845*	8895*	8915*	8935*	8938	8978*	9028*	9048*	9068*	9071	9112*

		9162*	9182*	9202*	9205	9247*	9297*	9317*	9337*	9340	9382*	9432*	9452*	9472*
STMP1	001174	9475	10720*	10847*	10885*									
STMP2	001176	539#	2974	2976	3005	3038	3041	6752*	6911*	7634*	7639	10721*	10884*	
STMP3	001200	540#	2974	3005	3038	6647*	6810*	7635*	7636	10719*	10968*	11036	11043*	
STMP4	001202	541#	7632*	7638*	7641*	7644*	7645	7805*	7810*	7817*	7821*	7824*	7825	
STMP5	001204	542#												
		543#	5304*	5336*	5368*	5400*	5432*	5464*	5535*	5598*	5660*	5722*	5784*	5846*
STN =	000125	5898*	5941*	5984*	6032*	6081*	6129*							
		1#	10	4692	4710	4726#	4738	4746	4756	4759#	4789	4798	4801#	4812
		4826	4829#	4859	4870	4873#	4888	4898	4901#	4915	4924	4927#	4942	4952
		4955#	4964	4979	4992	4995#	5037	5055	5058#	5155	5168	5171#	5202	5205
		5216	5219#	5270	5283	5286#	5303	5307	5320	5323#	5335	5339	5352	5355#
		5367	5371	5384	5387#	5379	5403	5416	5419#	5431	5435	5448	5451#	5463
		5476	5491	5494#	5534	5538	5554	5557#	5597	5601	5616	5619#	5659	5663
		5678	5681#	5721	5725	5740	5743#	5783	5787	5802	5805#	5845	5858	5873
		5876#	5897	5901	5916	5919#	5940	5944	5959	5962#	5983	5987	6001	6004#
		6031	6036	6050	6053#	6080	6084	6098	6101#	6128	6141	6151	6154#	6176
		6186	6189#	6211	6221	6224#	6246	6256	6259#	6281	6291	6294#	6316	6326
		6329#	6352	6366	6369#	6397	6407	6410#	6415	6612	6624	6627#	6728	6773
		6785	6788#	6886	6944	6956	6959#	6960	7014	7026	7029#	7068	7079	7082#
		7145	7156	7159#	7181	7201	7204#	7251	7264	7267#	7308	7319	7322#	7357
		7368	7371#	7391	7405	7408#	7449	7464	7467#	7501	7513	7516#	7549	7563
		7566#	7591	7604	7607#	7653	7666	7669#	7722	7743	7761	7764#	7838	7858
		7861	7912	7932	7943	7946#	8012	8024	8027#	8029	8169	8181	8184#	8249
		8262	8265#	8352	8365	8368#	8452	8462	8465#	8515	8526	8529#	8531	8614
		8630	8646	8649#	8819	8835	8838#	8952	8968	8971#	9085	9101	9104#	9220
		9236	9239#	9355	9371	9374#	9500	9513	9516#	9518	9657	9667	9680	9683#
		9793	9803	9818	9821#	9930	9942	9957	9960#	10061	10072	10086	10089#	10200
		10209	10223	10226#	10348	10359	10374	10377#	10486	10496	10513#	10639	10654#	10769
		10774	10786#	10815	10830#	10854	10865#	10890	10892	10904#	10957	10967#	11081	11095#
STPB	001146	525#	3690*	3701										
STPFLG	001153	529#	3651	3701										
STPS	001144	524#	3688	3701										
STRAP	030326	3856#	4626											
STRP =	000022	3863#	3873#	3874#	3875#	3876#	3877#	3878#	3879#	3880#	3881#			
STRPAD	030346	3860	3871#											
STSTN	001102	507#	3162*	3225	3255	3277*	3282	3286	3303	3313	3351	3370	3398	4724*
		4725	4998*	4999	5289*	5290	6418*	6419	6963*	6964	8032*	8033	8534*	8535
		9521*	9522											
STYPEN=	***** U	3877												
STYPDS	030102	3792#	3876											
STYPE	027444	3651#	3863	3872										
STYPEC	027604	3670	3677	3683	3688#	3689								
STYPEX	027652	3694	3696	3699#										
STYPOC	027700	3732#	3873											
STYPON	027714	3731	3734#	3875										
STYPOS	027654	3727#	3874											
SXTSTR	025512	3240#												
SSGET4=	000001	3190#	3195											
SSTRP =	000002	3862#	3873	3874	3875	3876	3877	3878	3879	3880	3881			
SSSTN	001104	509#	3370*	3375	3398*	3403								
SOF ILL	030077	3728*	3732*	3742	3777#									
		446#	450	452#	490	491#	493#	495#	503#	550	2925#	3175#	3216	3218#
		3285	3286	3351	3362#	3368#	3390#	3396#	3542#	3701	3846#	3898	3919	4060#
		4556#	4560#	4573#	4581#	4595#	4638	4639	4669#	8144*	8150	11118	11121#	11123#

MSG15	5270#	5272
MSG16	5307#	5309
MSG161	10774#	10776
MSG162	10815#	10817
MSG163	10854#	10856
MSG164	10892#	10894
MSG165	10957#	10959
MSG166	11081#	11083
MSG17	5339#	5341
MSG2	4746#	4748
MSG20	5371#	5373
MSG200	10639#	10641
MSG21	5403#	5405
MSG22	5435#	5437
MSG23	5476#	5478
MSG24	5538#	5540
MSG25	5601#	5603
MSG26	5663#	5665
MSG27	5725#	5727
MSG3	4789#	4791
MSG30	5787#	5789
MSG31	5858#	5860
MSG32	5901#	5903
MSG33	5944#	5946
MSG34	5987#	5989
MSG35	6036#	6038
MSG36	6084#	6086
MSG37	6141#	6143
MSG4	4812#	4814
MSG40	6176#	6178
MSG41	6211#	6213
MSG42	6246#	6248
MSG43	6281#	6283
MSG44	6316#	6318
MSG45	6352#	6354
MSG46	6397#	6399
MSG47	6612#	6614
MSG5	4859#	4861
MSG50	6773#	6775
MSG512	6944#	6946
MSG53	7068#	7070
MSG54	7145#	7147
MSG55	7181#	7183
MSG56	7251#	7253
MSG57	7308#	7310
MSG6	4888#	4890
MSG60	7357#	7359
MSG61	7391#	7393
MSG62	7449#	7451
MSG63	7501#	7503
MSG64	7549#	7551
MSG65	7591#	7593
MSG66	7653#	7655
MSG67	7743#	7745
MSG7	4915#	4917
MSG70	7838#	7840

7016

MSG71	7932#	7934													
MSG72	8012#	8014													
MSG73	8169#	8171													
MSG74	8249#	8251													
MSG75	8352#	8354													
MSG76	8452#	8454													
MSG77	8515#	8517													
MULT	1#	414#													
NEWTST	1#	414#	4710	4746	4789	4812	4859	4888	4915	4942	4979	5037	5155	5205	5270
	5307	5339	5371	5403	5435	5476	5538	5601	5663	5725	5787	5858	5901	5944	5987
	6036	6084	6141	6176	6211	6246	6281	6316	6352	6397	6612	6773	6944	7014	7068
	7145	7181	7251	7308	7357	7391	7449	7501	7549	7591	7653	7743	7838	7932	8012
	8169	8249	8352	8452	8515	8630	8819	8952	9085	9220	9355	9500	9667	9803	9942
	10072	10209	10359	10496	10639	10774	10815	10854	10892	10957	11081				
PAGEF	1#	6977	7030												
PARMSG	5901#	5903	5946												
PDRMSG	6036#	6038	6086												
POP	1#	414#	3622	3833	3905										
PUSH	1#	414#	3602	3792	3889										
RESPMG	5307#	5309	5341	5373	5405	5437									
RESPON	1#	5291	5323	5355	5387	5419	5451								
SAVEAD	1#	10786	10830												
SAVTST	1#	3303													
SCOPE	33#	4756	4798	4826	4870	4898	4924	4952	4992	5055	5168	5216	5283	5320	5352
	5384	5416	5448	5491	5554	5616	5678	5740	5802	5873	5916	5959	6001	6050	6098
	6151	6186	6221	6256	6291	6326	6366	6407	6624	6785	6956	7026	7079	7156	7201
	7264	7319	7368	7405	7464	7513	7563	7604	7666	7761	7858	7943	8024	8181	8262
	8365	8462	8526	8646	8835	8968	9101	9236	9371	9513	9680	9818	9957	10086	10223
	10374	10512	10653	10785	10829	10864	10903	10966	11094	11113					
SETACF	1#	7159	7204	7267	7322	7371									
SETTRA	3863#	3873	3874	3875	3876	3877	3878	3879	3880						
SETUP	1#	414#													
SKIP	1#	414#	4692	4738	4964	5202	5303	5335	5367	5399	5431	5463	5534	5597	5659
	5721	5783	5845	5897	5940	5983	6031	6080	6128	6728	6886	7722	7912	8614	9657
	9793	9930	10061	10200	10348	10486	10769	10890	11108						
SLASH	1#	414#													
SPACE	414#														
SSCOPE	1#	3235													
STARS	1#	414#	442	470	496	623	3148	3219	3286	3356	3583	3629	3701	3779	3847
	3882	3927	4021	4541	4546	4710	4720	4746	4754	4789	4796	4812	4824	4859	4868
	4888	4896	4915	4922	4942	4950	4979	4990	5037	5053	5155	5166	5205	5214	5270
	5281	5307	5318	5339	5350	5371	5382	5403	5414	5435	5446	5476	5489	5538	5552
	5601	5614	5663	5676	5725	5738	5787	5800	5858	5871	5901	5914	5944	5957	5987
	5999	6036	6048	6084	6096	6141	6149	6176	6184	6211	6219	6246	6254	6281	6289
	6316	6324	6352	6364	6397	6405	6612	6622	6773	6783	6944	6954	7014	7024	7068
	7077	7145	7154	7181	7199	7251	7262	7308	7317	7357	7366	7391	7403	7449	7462
	7501	7511	7549	7561	7591	7602	7653	7664	7743	7759	7838	7856	7932	7941	8012
	8022	8169	8179	8249	8260	8352	8363	8452	8460	8515	8524	8630	8644	8819	8833
	8952	8966	9085	9099	9220	9234	9355	9369	9500	9511	9667	9678	9803	9816	9942
	9955	10072	10084	10209	10221	10359	10372	10496	10511	10639	10652	10774	10784	10815	10828
	10854	10863	10892	10902	10957	10965	11081	11093	11112						
TRACK	1#	4755	4797	4825	4869	4897	4923	4951	4991	5054	5167	5215	5282	5319	5351
	5383	5415	5447	5490	5553	5615	5677	5739	5801	5872	5915	5958	6000	6049	6097
	6150	6185	6220	6255	6290	6325	6365	6406	6623	6784	6955	7025	7078	7155	7200
	7263	7318	7367	7404	7463	7512	7562	7603	7665	7760	7857	7942	8023	8180	8261
	8364	8461	8525	8645	8834	8967	9100	9235	9370	9512	9679	9817	9956	10085	10222

.SSIZE	1#	3927
.SSUPR	1#	
.STRAP	1#	3847
.STYPB	1#	
.STYPD	1#	3779
.STYPE	1#	3629
.STYPO	1#	3701
.1170	1#	24

. ABS. 101360 000

ERRORS DETECTED: 0

DSKZ:CEKBEE.BIN,DSKZ:CEKBEE.LST/CRF/SOL/NL:TOC=DSKZ:CEKBEE.SML,DSKZ:CEKBEE.P11
RUN-TIME: 82 123 10 SECONDS
RUN-TIME RATIO: 465/217=2.1
CORE USED: 41K (81 PAGES)