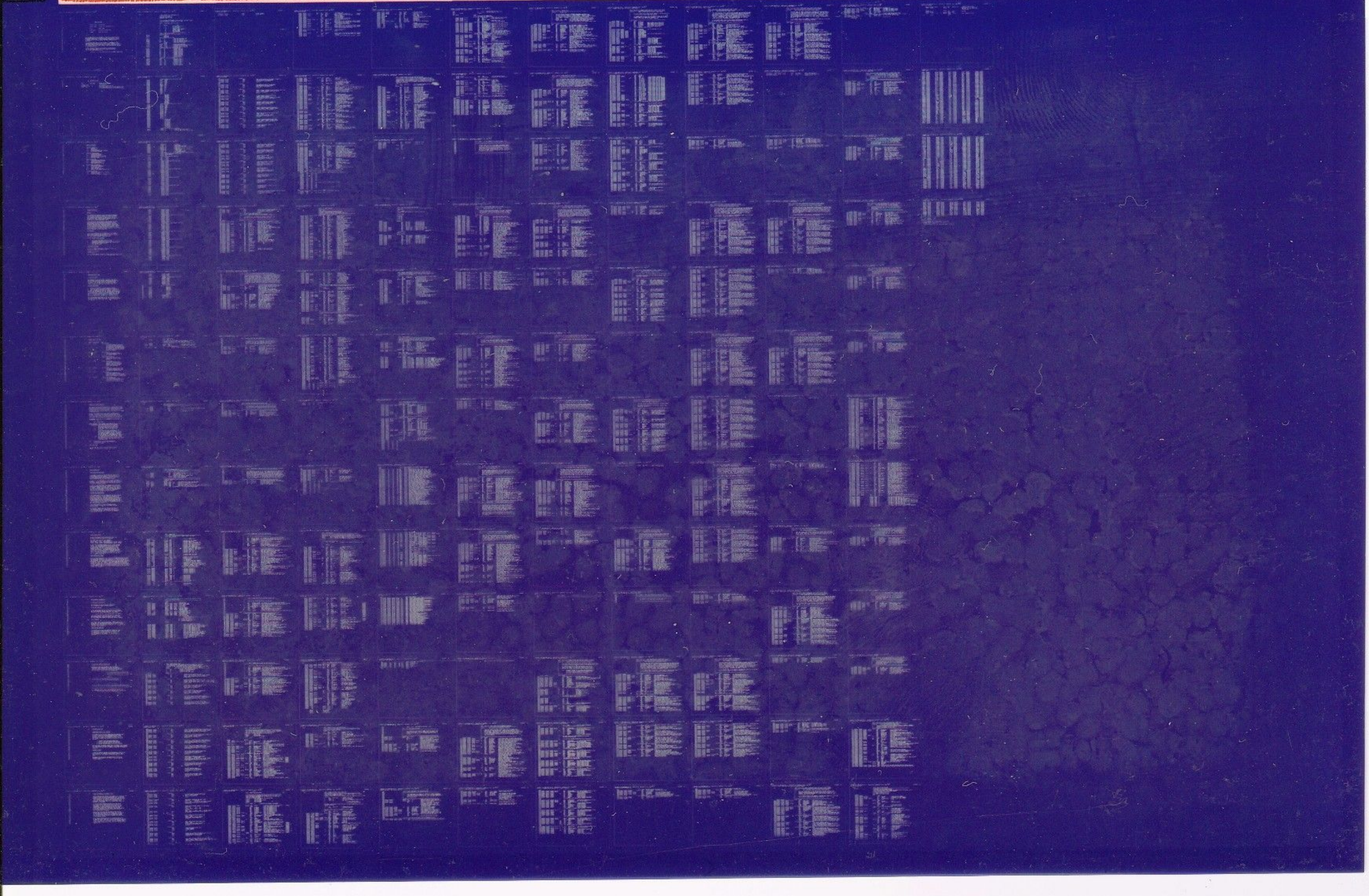
KD11-Z 11/44 MEM MGMT PRT B AH-F614C-MC
CKKTBCO FICHE 1 OF 1



```
1                                       .REM    @
2
3
4
5
6
7
8
9                                               IDENTIFICATION
10                                              --------------
11
12                                      PRODUCT CODE:   AC-F612C-MC
13
14                                      PRODJCT NAME:   CKKTBC0 11/44 MEM MGMT PRT B
15
16                                      DATE:           APRIL, 1981
17
18                                      MAINTAINER:     DIAGNOSTIC ENGINEERING
19
20                                      AUTHOR:         DAN P. MILLEVILLE
21
22
23
24
25
26                      THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT
27                      NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL
28                      EQUIPMENT CORPORATION.  DIGITAL EQUIPMENT CORPORATION ASSUMES
29                      NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS
30                      DOCUMENT.
31
32                      THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE
33                      PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM
34                      AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE)
35                      ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED
36                      IN WRITING BY DIGITAL.
37
38                      DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY
39                      FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT
40                      THAT IS NOT SUPPLIED BY DIGITAL.
41
42                              COPYRIGHT   (C) 1979, 1981 BY DIGITAL EQUIPMENT CORPORATION
```

43
44
45
46
47
48
49
50
51
52

PROGRAM HISTORY

DATE	REVISION	REASON FOR REVISION
OCTOBER, 1979	A	FIRST RELEASE
DECEMBER, 1979	B	ADDITION OF 'CSM' TEST 35
APRIL, 1981	C	USING NEW SYSMAC WITH ^Q CHECKS AND BIT CHECK OF THE POWER MONITOR BIT OF CPUERR REGISTER

TABLE OF CONTENTS

1.0 PROGRAM INFORMATION

1.1 ABSTRACT

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

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

1.2 REQUIREMENTS

A PDP 11/44 PROCESSOR WITH A MINIMUM OF 16K OF MEMORY AND A CONSOLE TERMINAL ARE REQUIRED TO RUN THE PROGRAM UNLESS THE PROGRAM IS RUNNING UNDER APT OR ACT IN WHICH CASE THE CONSOLE TERMINAL IS NOT NECESSARY.

1.3 RELATED DOCUMENTS AND STANDARDS

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

1.4 PRELIMINARY PROGRAMS

BEFORE THIS MEMORY MANAGEMENT DIAGNOSITC IS RUN, THE FOLLOWING DIAGNOSTICS SHOULD BE RUN:

CKKAAA0 11/4 CPU/EIS
CKKABA0 TRAPS

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

2.0 OPERATING INSTRUCTIONS

2.1 LOADING PROCEDURES

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

2.2 STARTING PROCEDURES

T PROGRAM IS STARTED BY LOADING ADDRESS 200 AND
S TING. THE SWITCH REGISTER SHOULD BE SET ACCORDING TO
S ION 2.3 BEFORE THE PROGRAM IS STARTED. HOWEVER, IF
DESIRED, THE PROGRAM WILL USE THE SOFTWARE SWITCH REGISTER
AT LOCATION 176 (LOCATION 174 WILL BE USED AS THE SOFTWARE
DISPLAY REGISTER). IN THAT CASE, THE PROGRAM WILL ASK FOR
THE INITIAL SWITCH REGISTER VALUE BY TYPING ''SWR= XXXXXX
NEW= '' AFTER TYPING THE NAME OF THE PROGRAM (XXXXXX =
THE OCTAL CONTENTS OF LOCATION 176). (SEE SECTION 2.4)

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

2.3 CONTROL SWITCH SETTINGS

SWITCH	OCTAL VALUE	USE
SW15	100000	HALT ON ERROR THIS SWITCH WHEN SET WILL HALT THE PROCESSOR WHEN AN ERROR IS DETECTED AFTER THE ERROR MESSAGE HAS BEEN TYPED. PRESSING CONTINUE WILL RESUME TESTING (SEE SECTION 3.1 ABOUT LOADING THE SWITCH REG BEFORE CONTINUING).
SW14	040000	LOOP ON TEST THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE CURRENT SUBTEST.
SW13	020000	INHIBIT ERROR TYPEOUTS THIS SWITCH WHEN SET WILL INHIBIT THE TYPING OF ERROR MESSAGES.
SW12	010000	INHIBIT TRACE TRAP THIS SWITCH WHEN SET WILL INHIBIT T-BIT TRAPPING WHICH NORMALLY TAKES PLACE DURING EVERY OTHER PASS STARTING WITH THE THIRD PASS.
SW10	002000	BELL ON ERROR THIS SWITCH WHEN SET WILL RING THE CONSOLE TERMINAL BELL WHEN AN ERROR HAS BEEN DETECTED.
SW9	001000	LOOP ON ERROR THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE FIRST FAILURE WHICH IS ENCOUNTERED EVEN IF THE FAILURE IS INTERMITTANT
SW8	000400	LOOP ON TEST IN SWR<7:0> THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE TEST WHOSE TEST NUMBER IS SET IN BITS 7-0 OF THE SWITCH REG.

2.4 LOADING THE SWITCH REGISTER

THE CONSOLE SWITCH REGISTER PROVIDED IS LOADED DIRECTLY FROM THE CONSOLE BY TYPING A CONTROL P (^P), THEN WHEN THE CONSOLE PROMPT IS RECIEVED, TYPE 'D SW XXXXXX', WHERE 'XXXXXX' IS THE INTENDED VALUE OF THE SWITCH REGISTER. THE VALUE OF THE CONSOLE SWITCH REG. CAN BE CHANGED ANY TIME WHETHER THE PROGRAM IS RUNNING OR NOT.

TO LOAD THE SOFTWARE SWITCH REG. WHILE THE PROGRAM IS RUNNING, A CONTROL G (^G) SHOULD BE TYPED ON THE CONSOLE TERMINAL. (THE 'SCOPE' AND 'ERROR' ROUTINES CHECK TO SEE IF A ^C HAS BEEN TYPED.) THE ORIGINAL VALUE OF THE SOFTWARE SWTICH REG. WILL BE REQUESTED AS MENTIONED IN SECTION 2.2.

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

```
SWR = XXXXXX  NEW =
```

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

XXXXXX<CR>	ONE TO SIX OCTAL DIGITS FOLLOWED BY A CARRIAGE RETURN WHICH WILL BE LOADED AS THE NEW VALUE FOR THE SWITCH REG.
<CR>	JUST A <CR>, LEAVES THE SWITCH REG. AS IT IS.
XXX^U	A CONTROL-U (^U) WILL CAUSE ALL OF THE DIGITS TYPED SO FAR TO BE IGNORED.
^C	WILL CAUSE THE PROGRAM TO TYPE THE PRESENT TEST AND PASS NUMBERS, REQUEST A NEW VALUE FOR THE SWITCH REG., AND JUMP TO THE END-OF-PASS ROUTINE SO THE PROGRAM WILL GO DIRECTLY TO THE NEXT PASS WITH A NEW SW. REG. VALUE
<ILL.CHAR>	ANY CHARACTER TYPED WHICH IS NOT ANY OF THE ABOVE OR AN OCTAL DIGIT WILL CAUSE THE PROGRAM TO TYPE A '?<CRLF>' AND REACT AS THOUGH A ^U HAD BEEN TYPED.

NOTE: RECOGNITION OF A ^G MAY BE HAMPERED BY EXECUTION OF A COUPLE 'RESET' INSTRUCTIONS WITHIN THE PROGRAM.

2.5 EXECUTION TIMES

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

3.0 ERROR INFORMATION

3.1 ERROR REPORTING PROCEDURES

IF AN ERROR IS DETECTED, THE PROGRAM WILL TRAP TO THE ERROR HANDLING ROUTINE ($ERROR). THE VALUE OF BITS 15,13,10, AND 9 IN THE SWITCH REGISTER ARE CONSIDERED IN REPORTING AN ERROR (SEE SECTION 2.3). THE ERROR INFORMATION WILL BE TYPED UNLESS SW13 = 1.

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

IF SW9 = 1 (LOOP ON ERROR), THE PROGRAM WILL GO TO THE ADDRESS CONTAINED IN LOCATION "$LPERR". AFTER REPORTING THE ERROR. "$LPERR" IS SET BY EACH "SCOPE" CALL AND IS SET DIRECTLY DURING SOME SUBTESTS TO PROVIDE THE SMALLEST LOOP FOR LOOPING ON ERROR. IF SW9 = 0, THE PROGRAM WILL RETURN TO THE INSTRUCTION FOLLOWING THE ERROR CALL. (SEE SECTION 5.3 FOR MORE ON 'LOOP ON ERROR').

3.2 INTERPRETING ERROR REPORTS

EVERY ERROR REPORT TYPES THE NUMBER OF THE TEST IN WHICH THE ERROR TOOK PLACE (TESTNO) AND THE LOCATION OF THE ERROR CALL (ERRORPC). THESE TWO VALUES PINPOINT THE PLACE IN THE CODE THAT THE ERROR OCCURRED. BY REFERRING TO THE PROGRAM LISTING, THE OPERATOR CAN THEN READ THE COMMENTS ASSOCIATED WITH THAT PARTICULAR ERROR AND SUBTEST. A DESCRIPTION OF THE TEST FOUND IN THE PROGRAM LISTING WILL ALSO PROVIDE THE OPERATOR WITH INFORMATION ON THE LOGIC AND FUNCTIONS BEING TESTED.

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

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

3.3 SAMPLE ERROR REPORT

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

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

WE SEE THAT THE ERROR OCCURRED IN TEST 12 AT LOCATION
022060. THE "REGISTER ADDRESS" TELLS US THAT WE WERE
TESTING MEMORY MANAGEMENT'S STATUS RGISTER 0 (SR0).
IN THE LISTING, THE TEST DESCRIPTION SAYS THAT THE
ERROR BITS (BITS <15:13>) OF SR0 WERE BEING SET AND
CLEARED INDIVIDUALLY. THE ERROR REPORT SAYS WE TRIED
TO SET BIT 14 BY WRITING "040000" TO SR0 BUT WHEN WE
READ IT BACK WE READ "060000". IT APPEARS THAT BIT 13 IS
STUCK AT "1" OR IT IS GETTING SET WHEN BIT 14 IS SET
TO "1". ERROR REPORTS BEFORE AND AFTER THIS ONE COULD
TELL US WHICH IS THE CASE.

3.4 POWER MONITOR BIT ERRORS

WHEN THE POWER MONITOR BIT (BIT 0 OF THE CPU ERROR REGISTER) BECOMES
SET DURING THE RUNNING OF THIS DIAGNOSTIC, THE $SCOPE ROUTINE WILL
CALL AN ERROR. IF THE BIT SHOULD BECOME SET AFTER THE $SCOPE ROUTINE
AND BEFORE AN ERROR, AND THE ERROR IS CALLED FOR ANY REASON, THE ERROR
ROUTINE WILL CALL *TWO* ERRORS. THE FIRST ERROR WILL BE THE POWER
MONITOR BIT ERROR CALL, THEN THE ERROR CALL ORIGINALLY CALLED WILL BE
PRINTED. IN ANY CASE, LOOP-ON-ERROR IS DISABLED FOR THIS ERROR ONLY.
IT IS RECOMMENDED THAT IF THIS ERROR SHOULD BE CALLED, THAT THE PROBLEM
CAUSING THE BIT TO BE SET BE REPAIRED BEFORE RELYING ON THE RESULTS OF
RUNNING THIS DIAGNOSTIC.

4.0 MISCELLANEOUS INFORMATION

4.1 ACT/APT/XXDP COMPATABILITY

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

4.2 END-OF-PASS MESSAGE

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

```
        END OF PASS #2  TOTAL ERRORS SINCE LAST REPORT 0
```

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

4.3 T-BIT TRAPPING

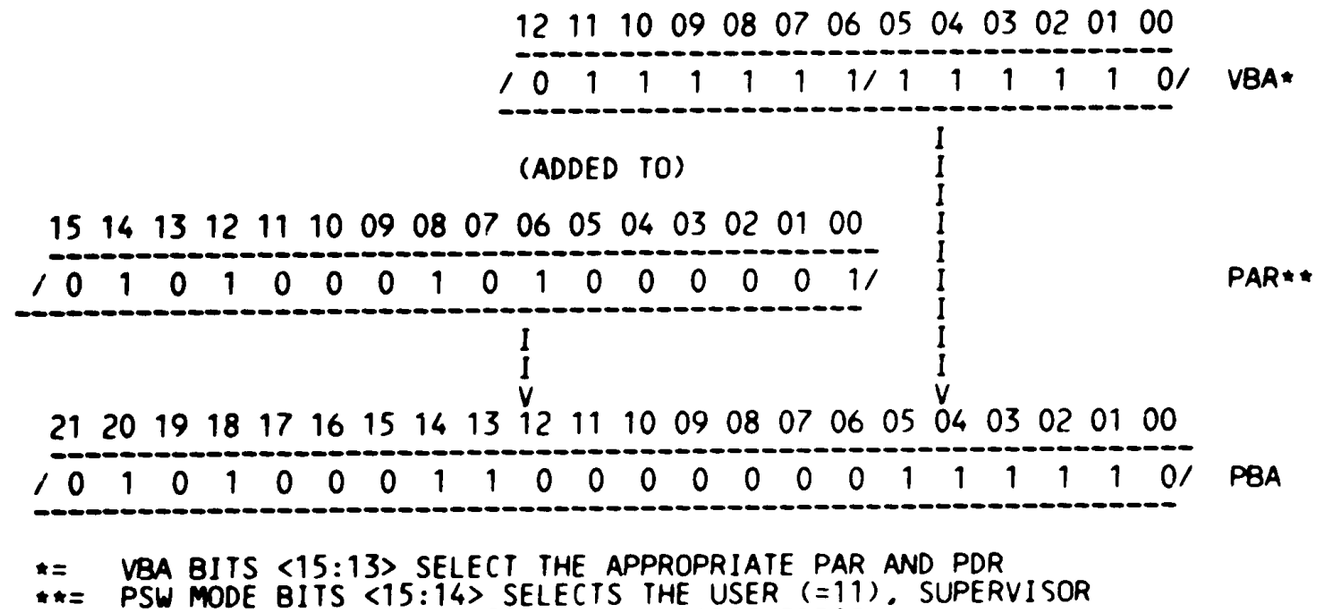
THE "T-BIT" (BIT 4) IN THE PROCESSOR STATUS WORD IS SET
BY AN 'RTI" IN THE END-OF-PASS ROUTINE FOR EVERY OTHER PASS
BEGINNING WITH THE THIRD PASS (PASSES 3,5,7,9...). T-BIT
TRAPPING CAN BE INHIBITED BY SETTING BIT 12 = 1 IN THE SWITCH
REGISTER (SEE SECTION 2.4).

4.4 POWER FAILURE HANDLING

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

4.5 PHYSICAL BUS ADDRESS CONSTRUCTION

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



*= VBA BITS <15:13> SELECT THE APPROPRIATE PAR AND PDR

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

5.0 PROGRAM DESCRIPTION

5.1 SUBROUTINES USED BY THIS PROGRAM

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

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

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

5.2 PROGRAM LISTING

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

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

5.3 USING THE PROGRAM TO DIAGNOSE A FAULT

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

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

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

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

@

```
956
957                             .TITLE  CKKTBC0 11/44 MEM MGMT PRT B
                                ;*COPYRIGHT (C) 1981
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.
                                ;*
958                             .SBTTL  OPERATIONAL SWITCH SETTINGS
                                ;*
                                ;*      SWITCH                  USE
                                ;*      ------          --------------------
                                ;*        15            HALT ON ERROR
                                ;*        14            LOOP ON TEST
                                ;*        13            INHIBIT ERROR TYPEOUTS
                                ;*        12            INHIBIT TRACE TRAP
                                ;*        10            BELL ON ERROR
                                ;*         9            LOOP ON ERROR
                                ;*         8            LOOP ON TEST IN SWR<7:0>
959                             .SBTTL  BASIC DEFINITIONS
                                ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                          STACK=  1100
104000                                  ERROR=EMT
000004                                  SCOPE=IOT
                                ;*MISCELLANEOUS DEFINITIONS
000011                          HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                          LF=     12              ;;CODE FOR LINE FEED
000015                          CR=     15              ;;CODE FOR CARRIAGE RETURN
000200                          CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776                          PS=     177776          ;;PROCESSOR STATUS WORD
177776                                  PSW=PS
177774                          STKLMT= 177774          ;;STACK LIMIT REGISTER
177772                          PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                          DSWR=   177570          ;;HARDWARE SWITCH REGISTER
177570                          DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
                                ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                          R0=     %0              ;;GENERAL REGISTER
000001                          R1=     %1              ;;GENERAL REGISTER
000002                          R2=     %2              ;;GENERAL REGISTER
000003                          R3=     %3              ;;GENERAL REGISTER
000004                          R4=     %4              ;;GENERAL REGISTER
000005                          R5=     %5              ;;GENERAL REGISTER
000006                          R6=     %6              ;;GENERAL REGISTER
000007                          R7=     %7              ;;GENERAL REGISTER
000006                          SP=     %6              ;;STACK POINTER
000007                          PC=     %7              ;;PROGRAM COUNTER
                                ;*PRIORITY LEVEL DEFINITIONS
000000                          PR0=    0               ;;PRIORITY LEVEL 0
000040                          PR1=    40              ;;PRIORITY LEVEL 1
000100                          PR2=    100             ;;PRIORITY LEVEL 2
000140                          PR3     140             ;;PRIORITY LEVEL 3
000200                          PR4=    200             ;;PRIORITY LEVEL 4
000240                          PR5=    240             ;;PRIORITY LEVEL 5
000300                          PR6=    300             ;;PRIORITY LEVEL 6
000340                          PR7=    340             ;;PRIORITY LEVEL 7
```

```
                        ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW09=   1000
000400                  SW08=   400
000200                  SW07=   200
000100                  SW06=   100
000040                  SW05=   40
000020                  SW04=   20
000010                  SW03=   10
000004                  SW02=   4
000002                  SW01=   2
000001                  SW00=   1
001000                          SW9=SW09
000400                          SW8=SW08
000200                          SW7=SW07
000100                          SW6=SW06
000040                          SW5=SW05
000020                          SW4=SW04
000010                          SW3=SW03
000004                          SW2=SW02
000002                          SW1=SW01
000001                          SW0=SW00
                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                  BIT15=  100000
040000                  BIT14=  40000
020000                  BIT13=  20000
010000                  BIT12=  10000
004000                  BIT11=  4000
002000                  BIT10=  2000
001000                  BIT09=  1000
000400                  BIT08=  400
000200                  BIT07=  200
000100                  BIT06=  100
000040                  BIT05=  40
000020                  BIT04=  20
000010                  BIT03=  10
000004                  BIT02=  4
000002                  BIT01=  2
000001                  BIT00=  1
001000                          BIT9=BIT09
000400                          BIT8=BIT08
000200                          BIT7=BIT07
000100                          BIT6=BIT06
000040                          BIT5=BIT05
000020                          BIT4=BIT04
000010                          BIT3=BIT03
000004                          BIT2=BIT02
000002                          BIT1=BIT01
000001                          BIT0=BIT00
                        ;*BASIC ''CPU'' TRAP VECTOR ADDRESSES
000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
```

BASIC DEFINITIONS

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

MEMORY MANAGEMENT DEFINITIONS

```
172204                  SIPDR2= 172204
172206                  SIPDR3= 172206
172210                  SIPDR4= 172210
172212                  SIPDR5= 172212
172214                  SIPDR6= 172214
172216                  SIPDR7= 172216
                        ;*SUPERVISOR ''D'' PAGE DESCRIPTOR REGISTERS
172220                  SDPDR0= 172220
172222                  SDPDR1= 172222
172224                  SDPDR2= 172224
172226                  SDPDR3= 172226
172230                  SDPDR4= 172230
172232                  SDPDR5= 172232
172234                  SDPDR6= 172234
172236                  SDPDR7= 172236
                        ;*SUPERVISOR ''I'' PAGE ADDRESS REGISTERS
172240                  SIPAR0= 172240
172242                  SIPAR1= 172242
172244                  SIPAR2= 172244
172246                  SIPAR3= 172246
172250                  SIPAR4= 172250
172252                  SIPAR5= 172252
172254                  SIPAR6= 172254
172256                  SIPAR7= 172256
                        ;*SUPERVISOR ''D'' PAGE ADDRESS REGISTERS
172260                  SDPAR0= 172260
172262                  SDPAR1= 172262
172264                  SDPAR2= 172264
172266                  SDPAR3= 172266
172270                  SDPAR4= 172270
172272                  SDPAR5= 172272
172274                  SDPAR6= 172274
172276                  SDPAR7= 172276
                        ;*KERNEL ''I'' PAGE DESCRIPTOR REGISTERS
172300                  KIPDR0= 172300
172302                  KIPDR1= 172302
172304                  KIPDR2= 172304
172306                  KIPDR3= 172306
172310                  KIPDR4= 172310
172312                  KIPDR5= 172312
172314                  KIPDR6= 172314
172316                  KIPDR7= 172316
                        ;*KERNEL ''D'' PAGE DESCRIPTOR REGISTERS
172320                  K PDR0= 172320
172322                  KDPDR1= 172322
172324                  KDPDR2= 172324
172326                  KDPDR3= 172326
172330                  KDPDR4= 172330
172332                  KDPDR5= 172332
172334                  KDPDR6= 172334
172336                  KDPDR7= 172336
                        ;*KERNEL ''I'' PAGE ADDRESS REGISTERS
172340                  KIPAR0= 172340
172342                  KIPAR1= 172342
172344                  KIPAR2= 172344
172346                  KIPAR3= 172346
172350                  KIPAR4- 172350
```

MEMORY MANAGEMENT DEFINITIONS

```
                172352                  KIPAR5= 172352
                172354                  KIPAR6= 172354
                172356                  KIPAR7= 172356
                                        ;*KERNEL "D" PAGE ADDRESS REGISTERS
                172360                  KDPAR0= 172360
                172362                  KDPAR1= 172362
                172364                  KDPAR2= 172364
                172366                  KDPAR3= 172366
                172370                  KDPAR4= 172370
                172372                  KDPAR5= 172372
                172374                  KDPAR6= 172374
                172376                  KDPAR7= 172376
    961                                 ;*ADDITIONAL DEFINITIONS
    962                                 ;*
    963         177572                          MMR0=SR0
    964         177574                          MMR1=SR1
    965         177576                          MMR2=SR2
    966         172516                          MMR3-SR3
    967         000006                          KSP-SP
    968         000006                          SSP=SP
    969         000006                          USP=SP
    970         000020                          TBIT=BIT4
    971         000100                          WBIT=BIT6
    972         177766                  CPUERR=177766
    973         001100                  KERSTK=STACK
    974         000700                  SUPSTK=STACK-200
    975         000600                  USESTK=STACK-300
    976
```

```
1008
                000000                  .SBTTL  TRAP CATCHER
                                                .=0
                                        ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ''.+2,HALT''
                                        ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                        ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                000174                          .=174
        000174  000000                  DISPREG: .WORD  0                ;;SOFTWARE DISPLAY REGISTER
        000176  000000                  SWREG:   .WORD  0                ;;SOFTWARE SWITCH REGISTER
                                        .SBTTL  STARTING ADDRESS(ES)
        000200  000137  020000                  JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
```

ACT11 HOOKS

```
1009                                    .SBTTL  ACT11 HOOKS
                                        ;;**********************************************************************
                                        ;HOOKS REQUIRED BY ACT11
000204                                          $SVPC=.                 ;SAVE PC
000046                                          .=46
000046  036120                                  $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
000052                                          .=52
000052  000000                                  .WORD   0               ;;2)SET LOC.52 TO ZERO
000204                                          .-$SVPC                 ;; RESTORE PC
```

```
1010                                    .SBTTL  APT PARAMETER BLOCK
                                        ;;**********************************************************************
                                        ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                        ;;**********************************************************************
                 000204                         .$X=.    ;;SAVE CURRENT LOCATION
                 000024                         .=24     ;;SET POWER FAIL TO POINT TO START OF PROGRAM
          000024 000200                         200      ;;FOR APT START UP
                 000044                         . 44     ;;POINT TO APT INDIRECT ADDRESS PNTR.
          000044 000204                         $APTHDR  ;;POINT TO APT HEADER BLOCK
                 000204                         .=.$X    ;;RESET LOCATION COUNTER
                                        ;;**********************************************************************
                                        ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                        ;INTERFACE SPEC.
          000204                        $APTHD:
          000204 000000                 $HIBTS: .WORD   0        ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
          000206 001224                 $MBADR: .WORD   $MAIL    ;;ADDRESS OF APT MAILBOX (BITS 0-15)
          000210 000002                 $TSTM:  .WORD   2        ;;RUN TIM OF LONGEST TEST
          000212 000005                 $PASTM: .WORD   5        ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
          000214 000005                 $UNITM: .WORD   5        ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
          000216 000014                         .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
1011

                                .SBTTL  COMMON TAGS
                                ;;*************************************************************
                                ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                                ;*USED IN THE PROGRAM.
                 001100             .=1100
001100                          $CMTAG:                         ;;START OF COMMON TAGS
001100  000000                          .WORD   0
001102     000                  $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103     000                  $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000                  $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000                  $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000                  $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000                  $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114     000                  $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115     001                  $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000                  $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000                  $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000                  $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000                  $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000                  $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                          .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                          .WORD   0
001134     000                  $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135     000                  $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                          .WORD   0
001140  177570                  SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570                  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560                  $TKS:   177560                  ;;TTY KBD STATUS
001146  177562                  $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566                  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154     000                  $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155     002                  $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156     012                  $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A 'LINE FEED'
001157     000                  $TPFLG: .BYTE   0               ;;'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
001160  000000                  $REGAD: .WORD   0               ;;CONTAINS THE ADDRESS FROM
                                                                ;;WHICH  ($REG0) WAS OBTAINED
                 000006                 .REPT   $CM3
001162  000000                  $REG0:  .WORD   0               ;;CONTAINS (($REGAD)+0)
001164  000000                  $REG1:  .WORD   0               ;;CONTAINS (($REGAD)+2)
001166  000000                  $REG2:  .WORD   0               ;;CONTAINS (($REGAD)+4)
001170  000000                  $REG3:  .WORD   0               ;;CONTAINS (($REGAD)+6)
001172  000000                  $REG4:  .WORD   0               ;;CONTAINS (($REGAD)+10)
001174  000000                  $REG5:  .WORD   0               ;;CONTAINS (($REGAD)+12)
                 000006         .REPT   6
001176  000000                  $TMP0:  .WORD   0               ;;USER DEFINED
001200  000000                  $TMP1:  .WORD   0               ;;USER DEFINED
001202  000000                  $TMP2:  .WORD   0               ;;USER DEFINED
001204  000000                  $TMP3:  .WORD   0               ;;USER DEFINED
001206  000000                  $TMP4:  .WORD   0               ;;USER DEFINED
001210  000000                  $TMP5:  .WORD   0               ;;USER DEFINED
001212  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001214     207      377     377 $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001217     000
001220     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
001221     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001222     012      000         $LF:    .ASCIZ  <12>            ;;LINE FEED
```

```
                                ;;*******************************************************************
                                .SBTTL  APT MAILBOX-ETABLE
                                ;;*******************************************************************
                                .EVEN
001224                          $MAIL:                          ;;APT MAILBOX
001224  000000                  $MSGTY: .WORD   AMSGTY   ;;MESSAGE TYPE CODE
001226  000000                  $FATAL: .WORD   AFATAL   ;;FATAL ERROR NUMBER
001230  000000                  $TESTN: .WORD   ATESTN   ;;TEST NUMBER
001232  000000                  $PASS:  .WORD   APASS    ;;PASS COUNT
001234  000000                  $DEVCT: .WORD   ADEVCT   ;;DEVICE COUNT
001236  000000                  $UNIT:  .WORD   AUNIT    ;;I/O UNIT NUMBER
001240  000000                  $MSGAD: .WORD   AMSGAD   ;;MESSAGE ADDRESS
001242  000000                  $MSGLG: .WORD   AMSGLG   ;;MESSAGE LENGTH
001244                          $ETABLE:                        ;;APT ENVIRONMENT TABLE
001244     000                  $ENV:   .BYTE   AENV     ;;ENVIRONMENT BYTE
001245     000                  $ENVM:  .BYTE   AENVM    ;;ENVIRONMENT MODE BITS
001246  000000                  $SWREG: .WORD   ASWREG   ;;APT SWITCH REGISTER
001250  000000                  $USWR:  .WORD   AUSWR    ;;USER SWITCHES
001252  000000                  $CPUOP: .WORD   ACPUOP   ;;CPU TYPE,OPTIONS
                                ;*                              BITS 15-11=CPU TYPE
                                ;*                                      11/04=01,11/05=02,11/20=03,11/40-04,11/45 05
                                ;*                                      11/70=06,PDQ=07,Q-10
                                ;*                              BIT 10=REAL TIME CLOCK
                                ;*                              BIT  9=FLOATING POINT PROCESSOR
                                ;*                              BIT  8=MEMORY MANAGEMENT
001254                          $ETEND:
                                .MEXIT

001254  000000                  TESTNO: .WORD   0               ;HOLDS TEST NUMBER FOR TYPEOUTS
001256  000000                  WASR6:  .WORD   0               ;USED TO STORE THE STACK POINTER AFTER A TRAP
001260  000000                  TRAPPC: .WORD   0               ;USED TO STORE THE PC OF A TRAP OR ABORT
001262  000000                  TRAPPS: .WORD   0               ;USED TO STORE THE PS OF A TRAP OR ABORT
001264  000000                  WASSR0: .WORD   0               ;USED TO STORE CONTENTS OF SR0
001266  000000                  WASSR1: .WORD   0               ;USED TO STORE CONTENTS OF SR1
001270  000000                  WASSR2: .WORD   0               ;USED TO STORE CONTENTS OF SR2
001272  000000                  WASSR3: .WORD   0               ;USED TO STORE CONTENTS OF SR3
001274  000000                  TBITPS: .WORD   0               ;SAVES THE PSW THAT MAY HAVE ITS T-BIT ON
001276  000000                  VIRT1:  .WORD   0               ;HOLDS VIRTUAL ADDRESS TO BE CONVERTED
001300  000000                  PBALO:  .WORD   0               ;HOLDS BITS <15:00> OF PHYSICAL ADDRESS
001302  000000                  PBAHI:  .WORD   0               ;HOLDS BITS <21:16> OF PHYSICAL ADDRESS
001304  000000                  BADPC:  .WORD   0               ;HOLDS PC FROM ABORT OR TRAP
001306  000200                  $MXCNT: .WORD   200             ;HOLD MAX. NUMBER OF LOOP ITERATIONS
001310  000000                  $TBIT:  .WORD   0               ;'T' BIT STATE INDICATOR
001312     136  103     015     $CNTLC: .ASCIZ  /^C/<15><12>    ;CONTROL C
001315     012  000
                                .EVEN
```

```
                                .SBTTL  ERROR POINTER TABLE
                                ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                                ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                                ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
                                ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                                ;*      DH              ;;POINTS TO THE DATA HEADER
                                ;*      DT              ;;POINTS TO THE DATA
                                ;*      DF              ;;POINTS TO THE DATA FORMAT
001320                          $ERRTB:
1012
1013 001320  007512             ;*ITEM 1
1014 001322  012456                     EM1                     ;UNEXPECTED CPU TRAP TO LOC. 004
1015 001324  015072                     DH1                     ;OLD PC  OLD PSW R6 WAS  CPUERR  TESTNO  ERRORPC
1016 001326  015643                     DT1                     ;TRAPPC,TRAPPS,WASR6,CPUERR,TESTNO,$ERPPC,0
1017                                    DF12                    ;0,0,0,0,0,0
1018
1019 001330  007552             ;*ITEM 2
1020 001332  012536                     EM2                     ;UNEXPECTED MEM. MGMT. TRAP TO LOC. 250
1021 001334  015110                     DH2                     ;OLD PC  OLD PSW R6 WAS  SR0     SR2     TESTNO  ERRORPC
1022 001336  015625                     DT2                     ;TRAPPC, TRAPPS, WASR6, WASSR0, WASSR2, TESTNO, $ERRPC, 0
1023                                    DF2                     ;0,0,0,0,0,0,0
1024
1025 001340  007621             ;*ITEM 3
1026 001342  012626                     EM10                    ;MEMORY MGMT. ACCESS ABORT DID NOT OCCUR
1027 001344  015130                     DH10                    ;PDR 4   PSW     TESTNO  ERRORPC
1028 001346  015634                     DT10                    ;$REG2,$TMP0,TESTNO,$ERRPC,0
1029                                    DF3                     ;0,0,0,0
1030
1031 001350  007671             ;*ITEM 4
1032 001352  012626                     EM11                    ;ACCESS ERROR DID NOT ABORT INSTRUCTION
1033 001354  015130                     DH10                    ;PDR 4   PSW     TESTNO  ERRORPC
1034 001356  015634                     DT10                    ;$REG2,$TMP0,TESTNO,$ERRPC,0
1035                                    DF3                     ;0,0,0,0
1036
1037 001360  007740             ;*ITEM 5
1038 001362  012666                     EM12                    ;SR0 DID NOT REPORT ACCESS ERROR CORRECTLY
1039 001364  015142                     DH12                    ;SR0 WAS EXPECTD PDR 4   PSW     TESTNO ERRORPC
1040 001366  015643                     DT12                    ;WASSR0,$REG3,$REG2,$TMP0,TESTNO,$ERRPC,0
1041                                    DF12                    ;0,0,0,0,0,0
1042
1043 001370  010012             ;*ITEM 6
1044 001372  012746                     EM13                    ;SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.
1045 001374  015160                     DH13                    ;SR2 WAS EXPECTD PDR 4   PSW     TESTNO ERRORPC
1046 001376  015643                     DT13                    ;WASSR2,$REG4,$REG2,$TMP0,TESTNO,$ERRPC,0
                                        DF12                    ;0,0,0,0,0,0
```

ERROR POINTER TABLE

```
1047                                    ;*ITEM 7
1048 001400  010063                             EM14    ;PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE
1049 001402  013026                             DH14    ;V.B.A.  KIPDR4  SR0 WAS SR2 WAS TESTNO  ERRORPC
1050 001404  015176                             DT14    ;$REG0,$REG4,WASSR0,WASSR2,TESTNO,$ERRPC,0
1051 001406  015643                             DF12    ;0,0,0,0,0,0
1052
1053                                    ;*ITEM 10
1054 001410  010144                             EM15    ;PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE
1055 001412  013106                             DH15    ;V.B.A.  KIPDR4  TESTNO  ERRORPC
1056 001414  015214                             DT15    ;$REG0,$REG4,TESTNO,$ERRPC.0
1057 001416  015634                             DF3     ;0,0,0,0
1058
1059                                    ;*ITEM 11
1060 001420  010227                             EM16    ;SR0 DID NOT REPORT PAGE LGTH. ABORT CORRECTLY
1061 001422  013146                             DH16    ;V.B.A.  KIPDR4  SR0 WAS EXPECTD TESTNO  ERRORPC
1062 001424  015226                             DT16    ;$REG0,$REG4,WASSR0,$REG2,TESTNO,$ERRPC,0
1063 001426  015643                             DF12    ;0,0,0,0,0,0
1064
1065                                    ;*ITEM 12
1066 C01430  010012                             EM13    ;SR2 DID NOT LOCKUP CORRECT VIRUAL ADDR.
1067 001432  013226                             DH17    ;V.B.A.  KIPDR4  SR2 WAS EXPECTD TESTNO  ERRORPC
1068 001434  015244                             DT17    ;$REG0,$REG4,WASSR2,$REG3,TESTNO,$ERRPC,0
1069 001436  015643                             DF12    ;0,0,0,0,0,0
1070
1071                                    ;*ITEM 13
1072 001440  010012                             EM13    ;SR2 DID NOT LOCKUP CORRECT VIRUAL ADDR.
1073 001442  013306                             DH20    ;SR2 WAS EXPECTD TESTNO  ERRORPC
1074 001444  015262                             DT20    ;WASSR2,$REG1,TESTNO,$ERRPC,0
1075 001446  015634                             DF3     ;0,0,0,0
1076
1077                                    ;*ITEM 14
1078 001450  010305                             EM21    ;SR0 OR SR2 CHANGED BY A SECOND ABORT
1079 001452  013346                             DH21    ;FIRST ABORT     SECOND ABORT
1080                                                     ;SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO  ERRORPC
1081 001454  015274                             DT21    ;$TMP0,$TMP2,WASSR0,WASSR2,TESTNO,$ERRPC,0
1082 001456  015643                             DF12    ;0,0,0,0,0,0
1083
1084                                    ;*ITEM 15
1085 001460  010352                             EM22    ;SR0 OR SR2 WAS NOT 'RESET'' BY A RESET
1086 001462  013463                             DH22    ;SR0 WAS SR2 WAS TESTNO  ERRORPC
1087 001464  015312                             DT22    ;WASSR0,WASSR2,TESTNO,$ERRPC,0
1088 001466  015634                             DF3     ;0,0,0,0
1089
1090                                    ;*ITEM 16
1091 001470  010421                             EM23    ;SR2 NOT TRACKING CORRECTLY
1092 001472  013306                             DH20    ;SR2 WAS EXPECTD TESTNO  ERROPC
1093 001474  015262                             DT20    ;WASSR2,$REG1,TESTNO,$ERRPC,0
1094 001476  015634                             DF3     ;0,0,0,0
1095
1096                                    ;*ITEM 17
1097 001500  010454                             EM24    ;DID NOT TRAP THRU KERNEL SPACE
1098 001502  013523                             DH24    ;PSW WAS R6 WAS  TESTNO  ERRORPC
1099 001504  015324                             DT24    ;$REG1,$REG2,TESTNO,$ERRPC,0
1100 001506  015634                             DF3     ;0,0,0,0
```

ERROR POINTER TABLE

```
1101                                    ;*ITEM 20
1102 001510  010513                             EM25        ;KT ERROR SERVICED ON ODD ADDR. ERROR
1103 001512  013306                             DH20        ;PDR     TESTNO  ERRORPC
1104 001514  015262                             DT20        ;$REG5,TESTNO,$ERRPC,0
1105 001516  015640                             DF5         ;0,0,0
1106
1107                                    ;*ITEM 21
1108 001520  010560                             EM26        ;SR0 OR SR2 CHANGED BY ODD ADDR. ERROR
1109 001522  013563                             DH26        ;EXPECTED        RECEIVED
1110                                                        ;SR0     SR2     SR0 WAS SR2 WAS TESTNO  ERRORPC
1111 001524  015336                             DT26        ;$REG0,$REG1,WASSR0,WASSR2,TESTNO,$ERRPC,0
1112 001526  015643                             DF12        ;0,0,0,0,0,0
1113
1114                                    ;*ITEM 22
1115 001530  010626                             EM27        ;ERROR DURING 'DOUBLE ERROR'' (KT & ODD ADDR.)
1116 001532  013675                             DH27        ;EXPECTED:
1117                                                        ;PSW     PC        SR0     SR2
1118                                                        ;170017  (3$+4)  020147  (3$)
1119                                                        ;RECEIVED
1120                                                        ;PSW     PC        SR0     SR2     TESTNO  ERRORPC
1121 001534  015354                             DT27        ;$REG1,$REG3,WASSR0,WASSR2,TESTNO,$ERRPC,0
1122 001536  015643                             DF12        ;0,0,0,0,0,0
1123
1124                                    ;*ITEM 23
1125 001540  010703                             EM30        ;MFPI INSTRUCTION PUSHED WRONG DATA
1126 001542  014072                             DH30        ;DATA    DATA
1127                                                        ;EXPECTD RECEIVD TESTNO  ERRORPC
1128 001544  015372                             DT30        ;$REG0,$REG1,TESTNO,$ERRPC,0
1129 001546  015634                             DF3         ;0,0,0,0
1130
1131                                    ;*ITEM 24
1132 001550  010746                             EM31        ;MTPI INSTRUCTION LOADED WRONG DATA
1133 001552  014072                             DH30        ;DATA    DATA
1134                                                        ;EXPECTD RECEIVD TESTNO  ERRORPC
1135 001554  015372                             DT30        ;$REG0,$REG1,TESTNO,$ERRPC,0
1136 001556  015634                             DF3         ;0,0,0,0
1137
1138                                    ;*ITEM 25
1139 001560  011011                             EM32        ;STACK NOT PUSHED BY MFPI-MTPI
1140 001562  014146                             DH32        ;TESTNO  ERRORPC
1141 001564  015404                             DT32        ;TESTNO,$ERRPC,0
1142 001566  015651                             DF32        ;0,0
1143
1144                                    ;*ITEM 26
1145 001570  011047                             EM33        ;KERNEL PAGE ACCESSED INSTEAD OF USER: MFPI-MTPI
1146 001572  014165                             DH33        ;SR0 WAS SR2 WAS TESTNO  ERRORPC
1147 001574  015312                             DT22        ;WASSR0,WASSR2,TESTNO,$ERRPC,0
1148 001576  015634                             DF3         ;0,0,0,0
1149
1150                                    ;*ITEM 27
1151 001600  011125                             EM34        ;M.M. ABORT IN KERNAL D-SPACE HAD WRONG CONDITION
1152 001602  014225                             DH34        ;(MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC EXPECTING 020031
1153 001604  015412                             DT34        ;$REG1,$REG2,$REG3,TESTNO,$ERRPC,0
1154 001606  015620                             DF1         ;0,0,0,0,0
1155
1156                                    ;*ITEM 30
1157 001610  011206                             EM35        ;ILLEGAL MODE 10 NOT ABORTED
```

ERROR POINTER TABLE

```
1158 001612  014146                    DH32                    ;TESTNO  ERRORPC
1159 001614  015404                    DT32                    ;TESTNO,$ERRPC,0
1160 001616  015651                    DF32                    ;0,0
```

ERROR POINTER TABLE

```
1161                                    ;*ITEM 31
1162 001620  011242                             EM36    ;SR0 DID NOT REPORT ILLEGAL MODE 10 CORRECTLY
1163 001622  014316                             DH36    ;SR0 WAS EXPECTD TESTNO  ERRORPC
1164 001624  015426                             DT36    ;WASSR0,$REG1,TESTNO,$ERRPC,0
1165 001626  015634                             DF3     ;0,0,0,0
1166
1167                                    ;*ITEM 32
1168 001630  011317                             EM37    ;PSW CHANGED BY AN RTI IN USER MODE
1169 001632  014356                             DH37    ;PSW WAS EXPECTD TESTNO  ERRORPC
1170 001634  015324                             DT24    ;$REG1,$REG2,TESTNO,$ERRPC,0
1171 001636  015634                             DF3     ;0,0,0,0
1172
1173                                    ;*ITEM 33
1174 001640  011362                             EM40    ;ABORT IN KERNAL D-SPACE PICKED UP VECTOR FROM I-SPACE
1175 001642  014416                             DH40    ;(PSW)   TESTNO  ERRORPC EXPECTING XXX340
1176 001644  015440                             DT40    ;$REG0,TESTNO,$ERRPC,0
1177 001646  015640                             DF5     ;0,0,0
1178
1179                                    ;*ITEM 34
1180 001650  011447                             EM41    ;D SPACE ENABLE CIRCUITRY HAS FAILED
1181 001652  014467                             DH41    ;ERROR   AUTOI/D VIRTUAL
1182                                                     ;REGISTR REGISTR ADDRESS TESTNO  PC AT ABORT
1183 001654  015450                             DT41    ;WASSR0,WASSR1,WASSR2,TESTNO,BADPC,0
1184 001656  015620                             DF1     ;0,0,0,0,0
1185
1186                                    ;*ITEM 35
1187 001660  011513                             EM42    ;INCORRECT STORE BY MTP INSTRUCTION
1188 001662  014573                             DH42    ;GDDATA  STORED  TESTNO  ERRORPC
1189 001664  015464                             DT42    ;$REG3,$REG4,TESTNO,$ERRPC,0
1190 001666  015634                             DF3     ;0,0,0,0
1191
1192                                    ;*ITEM 36
1193 001670  011556                             EM43    ;TRIED TO REFERENCE NON-RESIDENT PAGE
1194 001672  014633                             DH43    ;(MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC
1195 001674  015476                             DT43    ;$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
1196 001676  015620                             DF1
1197
1198                                    ;*ITEM 37
1199 001700  011623                             EM44    ;WRONG DATA FETCHED BY MFP INSTRUCTION
1200 001702  014072                             DH30    ;DATA    DATA
1201                                                     ;EXPECTD RECEIVD TESTNO  ERRORPC
1202 001704  015372                             DT30    ;$REG0,$REG1,TESTNO,$ERRPC,0
1203 001706  015634                             DF3     ;0,0,0,0
1204
1205                                    ;*ITEM 40
1206 001710  011556                             EM43    ;TRIED TO REFERENCE NON-RESIDENT PAGE
1207 001712  014633                             DH43    ;(MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC
1208 001714  015512                             DT45    ;WASSR0,WASSR1,WASSR2,TESTNO,$ERRPC,0
1209 001716  015620                             DF1     ;0,0,0,0,0
1210
1211                                    ;*ITEM 41
1212 001720  011671                             EM45    ;ILLEGAL CSM DID NOT TRAP TO 10
1213 001722  014146                             DH32    ;TESTNO  ERRORPC
1214 001724  015404                             DT32    ;TESTNO,$ERRPC,0
1215 001726  015625                             DF2     ;0,0
```

ERROR POINTER TABLE

```
1216
1217 001730  011730                ;*ITEM 42
1218 001732  014703                        EM46    ;CSM DID NOT ENTER SUPERVISOR MODE
1219 001734  015526                        DH44    ;EXPECTD (PSW)   TESTNO  ERR PC
1220 001736  015620                        DT46    ;$REG3,ACSMPS,TESTNO,$ERRPC,0
1221                                       DF1     ;0,0,0,0,0
1222                                ;*ITEM 43
1223 001740  011772                        EM47    ;CSM SET UP WRONG PREVIOUS MODE
1224 001742  014703                        DH44    ;EXPECTD (PSW)   TESTNO  ERR PC
1225 001744  015526                        DT46    ;$REG3,ACSMPS,TESTNO,$ERRPC,0
1226 001746  015620                        DF1     ;0,0,0,0,0
1227
1228                                ;*ITEM 44
1229 001750  012031                        EM50    ;CSM SET UP STACK WRONG
1230 001752  014072                        DH30    ;DATA    DATA
1231                                               ;EXPECTD RECEIVD TESTNO  ERR PC
1232 001754  015372                        DT30    ;ACSMSP,$TMP0,TESTNO,$ERRPC,0
1233 001756  015620                        DF1     ;0,0,0,0,0
1234
1235                                ;*ITEM 45
1236 001760  012060                        EM51    ;CSM PUSHED INCORRECT ARGUMENT
1237 001762  014072                        DH30    ;DATA    DATA
1238                                               ;EXPECTD RECEIVD TESTNO  ERR PC
1239 001764  015540                        DT47    ;$REG0,CSM1ST,TESTNO,$ERRPC,0
1240 001766  015620                        DF1     ;0,0,0,0,0
1241
1242                                ;*ITEM 46
1243 001770  012116                        EM52    ;CSM PUSHED WRONG PC
1244 001772  014072                        DH30    ;DATA    DATA
1245                                               ;EXPECTD RECEIVD TESTNO  ERR PC
1246 001774  015552                        DT50    ;$TMP0,CSM2ND,TESTNO,$ERRPC,0
1247 001776  015620                        DF1     ;0,0,0,0,0
1248
1249                                ;*ITEM 47
1250 002000  012142                        EM53    ;CSM DID NOT CLEAR OLD PSW BITS <3:0>
1251 002002  014703                        DH44    ;OLDPSW  TESTNO  ERR PC
1252 002004  015564                        DT52    ;CSM3RD,TESTNO,$ERRPC,0
1253 002006  015634                        DF3     ;0,0,0,0
1254
1255                                ;*ITEM 50
1256 002010  012207                        EM54    ;CSM ACCESSED WRONG SUPERVISOR SPACE
1257 002012  014146                        DH32    ;TESTNO  ERR PC
1258 002014  015404                        DT32    ;TESTNO,$ERRPC,0
1259 002016  015640                        DF5     ;0,0,0
1260
1261                                ;*ITEM 51
1262 002020  012253                        EM55    ;CSM ABORTED WHEN IT SHOULD NOT HAVE
1263 002022  014146                        DH32    ;TESTNO  ERR PC
1264 002024  015404                        DT32    ;TESTNO,$ERRPC,0
1265 002026  015640                        DF5     ;0,0,0
1266
1267                                ;*ITEM 52
1268 002030  012317                        EM56    ;CSM FAILED TO INCREMENT/DECREMENT REGISTER PROPERLY
1269 002032  014773                        DH55    ;TESTNO  ERR PC  R0 EXP  R0 RCV
1270 002034  015574                        DT55    ;TESTNO,$ERRPC,$TMP0,$REG0,0
1271 002036  015634                        DF3     ;0,0,0,0
```

ERROR POINTER TABLE

```
1272
1273 002040  012403
1274 002042  015032
1275 002044  015606
1276 002046  015634
1277
1278
1279
1280
1281
1282
1283
1284
1285 002050
1286 002050  012700  077406
1287
1288 002054  012702  172300
1289 002060  012701  000020
1290 002064  010022
1291 002066  077102
1292 002070  020227  172340
1293 002074  001003
1294 002076  012702  172200
1295 002102  000766
1296 002104  020227  172240
1297 002110  001003
1298 002112  012702  177600
1299 002116  000760
1300 002120  012701  172340
1301 002124  012702  172360
1302 002130  012703  000007
1303 002134  005000
1304 002136  010021
1305 002140  010022
1306 002142  062700  000200
1307 002146  077305
1308 002150  012711  177600
1309 002154  012712  177600
1310 002160  020127  172356
1311 002164  001005
1312 002166  012701  172240
1313 002172  012702  172260
1314 002176  000754
1315 002200  020127  172256
1316 002204  001401
1317 002206  000207
1318 002210  012701  177640
1319 002214  012702  177660
1320 002220  000743
```

```
;*ITEM 53
        EM57                    ;CSM FAILED TO PUT PROPER ARGUMENT ON STACK
        DH57                    ;TESTNO  ERR PC  ARGEXP  ARGRCV
        DT57                    ;TESTNO,$ERRPC,$TMP1,$TMP2,0
        DF3                     ;0,0,0,0
.SBTTL  INITIALIZE ALL PAR'S AND PDR'S
;*      *****   SUBROUTINES UNIQUE TO THIS PROGRAM    *****
;;**************************************************************
;*
;*      THIS ROUTINE WILL INITIALIZE ALL KERNAL, SUPERVISOR, AND
;*      USER PAR'S AND PDR'S TO THEIR USUAL INITIAL VALUE
;*
;;**************************************************************
APRINIT:
        MOV     #77406,R0       ;MAKE ALL PDR'S 4K, READ/WRITE, UPWARDS
                                ;EXPANDING, 200 BLOCKS
        MOV     #KIPDR0,R2      ;LOAD THE ADDRESS OF THE FIRST KERNAL PDR
1$:     MOV     #20,R1          ;LOAD R1 WITH 16
2$:     MOV     R0,(R2)+        ;LOAD EACH PDR IN TURN
        SOB     R1,2$           ;LOOP UNTIL ALL ARE LOADED
        CMP     R2,#KDPDR7+2    ;HAVE WE LOADED ALL KERNAL PDR'S
        BNE     3$              ;BRANCH IF KERNAL & SUPER HAVE BEEN LOADED
        MOV     #SIPDR0,R2      ;LOAD ALL SUPERVISOR PDR'S
        BR      1$              ;BRANCH TO LOOP
3$:     CMP     R2,#SDPDR7+2    ;HAVE USER PDR'S BEEN DONE
        BNE     4$              ;BRANCH IF THEY HAVE
        MOV     #UIPDR0,R2      ;LOAD ALL USER PDR'S
        BR      1$              ;BRANCH TO LOOP
4$:     MOV     #KIPAR0,R1      ;LOAD R1 WITH ADDRESS OF KIPAR0
        MOV     #KDPAR0,R2      ;LOAD R2 WITH ADDRESS OF KDPAR0
5$:     MOV     #7,R3           ;LOAD LOOP COUNTER WITH 7
        CLR     R0              ;CLEAR PAR VALUE REGISTER
6$:     MOV     R0,(R1)+        ;LOAD AN I-SPACE PAR
        MOV     R0,(R2)+        ;LOAD A D-SPACE PAR
        ADD     #200,R0         ;INCREASE THE PAR VALUE BY 200
        SOB     R3,6$           ;LOOP UNTIL 7 PAR'S ARE LOADED
        MOV     #177600,(R1)    ;MAP I-SPACE PAR7 TO I/O PAGE
        MOV     #177600,(R2)    ;MAP D-SPACE PAR7 TO I/O PAGE
        CMP     R1,#KIPAR7
        BNE     7$
        MOV     #SIPAR0,R1
        MOV     #SDPAR0,R2
        BR      5$
7$:     CMP     R1,#SIPAR7
        BEQ     8$              ;BRANCH TO USER LOAD ROUTINE
        RTS     PC              ;RETURN TO CALLING ROUTINE
8$:     MOV     #UIPAR0,R1
        MOV     #UDPAR0,R2
        BR      5$
```

```
                                        .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.  IN
                                        ;*      ALL OF THE D-SPACE ENABLE TESTS, D-SPACE PAGES 1 & 3 ARE MAPPED
                                        ;*      NON-RESIDENT AND I-SPACE PAGE 3 IS MAPPED NON-RESIDENT.  ALL
                                        ;*      OTHER PAGES ARE MAPPED RESIDENT, 4K, READ/WRITE.  THEREFORE, IF
                                        ;*      THE NON-RESIDENT PAGE IS 1 OR 3 YOU ARE NOT FORCING I-SPACE WHEN
                                        ;*      YOU SHOULD.  IF THE NON-RESIDENT PAGE IS 3, AND YOU ARE IN TEST
                                        ;*      15, YOU ARE PROBABLY FORCING I-SPACE WHEN YOU SHOULD BE ALLOWING
                                        ;*      D-SPACE.
                                        ;;*****************************************************************
002222                                  NODSPAC:                        ;STARTING ADDRESS FOR ABORT SERVICE ROUTINE
002222  042737  000004  172516                BIC     #BIT2,MMR3      ;TURN OFF D-SPACE BEFORE DOING ROUTINE
002230  005227                          NDFLAG: INC     (PC)+           ;MAKE FLAG ZERO IF THE FIRST TIME
002232  177777                                .WORD   -1              ;FLAG SHOULD BE -1
002234  001401                                BEQ     10$             ;BRANCH IF FIRST TIME IN ROUTINE
002236  000000                                HALT                    ;I HAVE ENTERED THIS ROUTINE BEFORE
                                                                        ;THE FIRST ERROR IS REPORTED; THE SECOND
                                                                        ;ENTRY ADDRESS IS ON THE STACK, AND THE
                                                                        ;FIRST ERROR CONDITION IS PROBABLY STILL
                                                                        ;LOCKED UP.
002240  011637  001304                  10$:    MOV     (KSP),BADPC     ;SAVE PC AT TIME OF ABORT OR TRAP
002244  012637  001260                          MOV     (KSP)+,TRAPPC   ;SAVE RETURN ADDRESS IN CASE OF LOOP
002250  012637  001262                          MOV     (KSP)+,TRAPPS   ;SAVE OLD PSW IN CASE OF LOOP
002254  013737  177572  001264                MOV     MMR0,WASSR0     ;SAVE STATUS REGISTER
002262  013737  177574  001266                MOV     MMR1,WASSR1     ;SAVE AUTO INCR/DECR REGISTER
002270  013737  177576  001270                MOV     MMR2,WASSR2     ;SAVE VIRTUAL ADDRESS REGISTER
002276  005737  001264                        TST     WASSR0          ;WAS ABORT NON-RESIDENT?
002302  100002                                BPL     1$              ;BRANCH IF ABORT NOT EXPECTED
002304  104034                                ERROR   +34             ;D-SPACE ENABLE FAULTY
002306  000401                                BR      2$              ;BRANCH TO EXIT
002310  104002                          1$:   ERROR   +2              ;UNEXPECTED M.M. ABORT
002312  042737  177376  177572        2$:   BIC     #177376,MMR0    ;CLEAR ALL BITS EXCEPT 0 AND 8
002320  012737  177777  002232                MOV     #-1,NDFLAG      ;MOVE A -1 TO THE FLAG
002326  013746  001262                        MOV     TRAPPS,-(KSP)   ;PUSH OLD PSW ONTO STACK
002332  013746  001260                        MOV     TRAPPC,-(KSP)   ;PUSH OLD PC ONTO STACK
002336  052737  000004  172516                BIS     #BIT2,MMR3      ;TURN D-SPACE BACK ON
002344  000006                                RTT                     ;RETURN TO MAIN PROGRAM
```

```
                                        .SBTTL  TURN OFF T-BIT AND SAVE CURRENT PSW
                                        ;;**********************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE IS USED TO TURN OFF THE TRACE TRAP BIT IN
                                        ;*      THE PSW IF IT IS ON.  THE PROCESSOR STATUS IS SAVED IN
                                        ;*      ''TBITPS'' SO THAT THE PSW CAN BE RESTORED TO ITS PREVIOUS
                                        ;*      CONDITION WHEN CONDITIONS WARRANT T-BIT TRAPPING.
                                        ;*
                                        ;;**********************************************************************
1370 002346  033727  177776  000020  TOFF:   BIT     PSW,#TBIT       ;IS THE T-BIT SET IN THE PSW?
1371 002354  001411                          BEQ     1$              ;EXIT IF NO
1372 002356  013746  177776                  MOV     PSW,-(SP)       ;PUSH PRESENT PSW ON THE STACK
1373 002362  011637  001274                  MOV     (SP),TBITPS     ;ALSO SAVE IT IN ''TBITPS'' FOR
1374                                                                  ;RESTORING LATER
1375 002366  042716  000020                  BIC     #TBIT,(SP)      ;CLEAR THE T-BIT (BIT 4) IN THE PSW
1376 002372  012746  002400                  MOV     #1$,-(SP)       ;PUSH PC OF 'RTS'' ON STACK
1377 002376  000006                          RTT                     ;''RETURN'' TO 1$ WITH T-BIT OFF
1378 002400  000207                  1$:     RTS     PC              ;RETURN TO PROGRAM
```

```
                                                .SBTTL  TURN ON T-BIT AND RESTORE PREVIOUS PSW
                                                ;*****************************************************************
                                                ;*
                                                ;*      THIS SUBROUTINE IS USED TO RESTORE THE PROCESSOR STATUS
                                                ;*      TO ITS PREVIOUS CONDITION BY RESTORING THE ''T-BIT PSW''
                                                ;*      SAVED BY THE ''TOFF'' SUBROUTINE IN THE ''TBITPS'' LOCATION.
                                                ;*
                                                ;*****************************************************************
1387 002402  033727  001274  000020  TON:      BIT     TBITPS,#TBIT    ;WAS T-BIT ON IN THE PREVIOUS PSW?
1388 002410  001410                             BEQ     1$              ;EXIT IF NO
1389 002412  013746  001274                     MOV     TBITPS,-(SP)    ;PUSH PREVIOUS PSW ON THE STACK
1390 002416  012737  000340  001274             MOV     #340,TBITPS     ;RESET THE ''TBITPS'' LOCATION
1391 002424  012746  002432                     MOV     #1$,-(SP)       ;PUSH PC OF ''RTS'' ON STACK
1392 002430  000006                             RTT                     ;''RETURN'' TO 1$ WITH T-BIT RESTORED
1393 002432  000207                     1$:     RTS     PC              ;RETURN TO PROGRAM
```

```
1394                                            .SBTTL  SUBROUTINE TO PREPARE ERLOOP, THE STACK AND EXIT TO ERROR
1395                                    ;;*****************************************************************
1396                                    ;*
1397                                    ;*      THIS SUBROUTINE IS USED BY THE THREE SUBROUTINES MFPITS, MTPITS AND
1398                                    ;*      MFPDTS WHEN AN ERROR IS CALLED.  THE RETURN ADDRESS IS POPPED AS
1399                                    ;*      RETURN IS NOT BACK TO THE SUBROUTINE, BUT TO THE TEST ERROR CALL.  THIS
1400                                    ;*      ROUTINE SETS LOCATION ERLOOP, DUPLICATES THE RETURN ADDRESS ON THE
1401                                    ;*      STACK FOR POSSIBLE LOOP ON ERROR, AND CORRECTS THE RETURN ADDRESS TO
1402                                    ;*      POINT TO THE ERROR CALL 4 LOCATIONS BACK FROM THE ''TEST PASSED'' LOCA-
1403                                    ;*      TION DUPLICATED.  NOTE THE ''TEST PASSED'' RETURN ON THE STACK IS NOT
1404                                    ;*      TOUCHED FOR PROBABLE LATER USE WHEN LOOPING IS NO LONGER ENABLED.
1405                                    ;*
1406                                    ;;*****************************************************************
1407 002434  005726                    ERPREP: TST     (SP)+           ;POP RETURN OF THIS ROUTINE - NOT USED
1408 002436  012737  000001  002614            MOV     #1,ERLOOP       ;SET ERROR LOOPING FLAG INDICATING AN ERROR
1409 002444  011646                            MOV     (SP),-(SP)      ;DUPLICATE RETURN PC AND
1410 002446  162716  000004                    SUB     #4,(SP)         ;FUDGE FOR ERROR RETURN
1411 002452  000207                            RTS     PC              ;RETURN TO THE ERROR CALL IN THE TEST
```

```
1412                                            .SBTTL  SUBROUTINE TO TEST MFPI INSTRUCTION
1413                                    ;;*****************************************************************
1414                                    ;*
1415                                    ;*      USAGE OF THE SUBROUTINE BELOW IS AS FOLLOWS:
1416                                    ;*
1417                                    ;*      MOV     #MFPILP,$LPERR  ;PUT ADDRESS OF LOOPING LOCATION IN $LPERR
1418                                    ;*      MOV     #(NUMB),MFPIPS  ;PUT PS PREVIOUS/CURRENT MODE VALUE IN THIS LOCATION
1419                                    ;*      MOV     #TRAPRTN,MFPIVC ;LOAD THE TEST EXCLUSIVE TRAP ROUTINE TO MFPIVC
1420                                    ;*      MOV     #(NUMB),R2      ;SETUP ADDRESS IN R2
1421                                    ;*    (IT IS ASSUMED THAT ALL PDR'S/PAR'S WILL ALSO BE SET UP PROPERLY)
1422                                    ;*      JSR     PC,MFPITS       ;GO DO THE TEST
1423                                    ;*      MFPI    (MODE)          ;MFPI INSTRUCTION TO TEST IS PUT HERE
1424                                    ;*      NOP                     ;NEEDED FOR MODES 1,2,4, & 5 *ONLY*
1425                                    ;*      ERROR   +(NUMB)         ;RETURN IS HERE FOR ERROR CALLS
1426                                    ;*      TST     (SP)+           ;POP STACK - EXTRA RETURN INSTALLED BY SUBRTN NOT NEEDED
1427                                    ;*
1428                                    ;*****************************************************************
1429 002454  005037  002614    MFPITS: CLR     ERLOOP          ;CLEAR ERROR LOOPING FLAG
1430 002460  011605                    MOV     (SP),R5         ;MOVE STACK POINTER TO R5 FOR SUBROUTINE LOADING
1431 002462  012537  002522            MOV     (R5)+,MFPILD    ;MOVE MFPI INSTRUCTION TO LOCATION
1432 002466  012537  002524            MOV     (R5)+,MFPILD+2  ;MOVE NEXT WORD TO LOCATION
1433 002472  062716  000010            ADD     #10,(SP)        ;FUDGE RETURN TO "TEST PASSED" LOCATION
1434 002476  010637  001176            MOV     SP,$TMP0        ;SAVE THE STACK POINTER AND
1435 002502  011637  001200            MOV     (SP),$TMP1      ;SAVE THE RETURN ADDRESS
1436 002506  012737            MFPILP: MOV     (PC)+,@(PC)+    ;SETUP PSW AS DEFINED BY LOADED VALUE IN PRVMD1
1437 002510  000000            MFPIPS: .WORD   0               ;LOCATION TO HOLD SUPER/USER PREVIOUS MODE
1438 002512  177776                    .WORD   PSW             ;LOAD THE PSW
1439 002514  017737  000072  000250    MOV     @MFPIVC,MMVEC   ;SET M.M. VECTOR TO MFPIV1
1440 002522  000000  000000    MFPILD: .WORD   0,0             ;LOCATIONS TO HOLD MFPI INSTRUCTION UNDER TEST
1441 002526  012737  015726  000250    MOV     #MGMERR,MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
1442 002534  012601                    MOV     (SP)+,R1        ;POP SUPERVISOR/USER/KERNEL STACK INTO R1
1443 002536  020001                    CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
1444 002540  001411                    BEQ     1$              ;BRANCH IF CORRECT DATA WAS FETCHED
1445 002542  012737  000340  177776    MOV     #340,PSW        ;GO TO KERNEL MODE
1446 002550  013706  001176            MOV     $TMP0,SP        ;RESET SP
1447 002554  013716  001200            MOV     $TMP1,(SP)      ;RESET RETURN ADDRESS
1448 002560  004737  002434            JSR     PC,ERPREP       ;GO PREPARE LOCATION ERLOOP, RETURN PC & EXIT TO ERROR
1449                            ;FOR TIGHTER SCOPE LOOP, REPLACE "JSR PC,ERPREP" WITH "BR MFPILP" = 000767
1450 002564  005737  002614    1$:     TST     ERLOOP          ;CHECK TO SEE IF THIS "PASSED" IS IN AN ERROR LOOP
1451 002570  001346                    BNE     MFPILP          ;BRANCH BACK IF SO
1452 002572  012737  000340  177776    MOV     #340,PSW        ;GET BACK TO KERNEL MODE
1453 002600  013706  001176            MOV     $TMP0,SP        ;RESET SP
1454 002604  013716  001200            MOV     $TMP1,(SP)      ;RESET RETURN ADDRESS
1455 002610  000207                    RTS     PC              ;EXIT - TEST PASSED
1456 002612  000000            MFPIVC: .WORD   0               ;LOCATION TO HOLD ADDRESS OF MM TRAP CATCHER
1457 002614  000000            ERLOOP: .WORD   0               ;LOCATION USED TO FLAG AN ERROR CONDITION
```

```
1458                                            .SBTTL  SUBROUTINE TO CHECK THE MTPI INSTRUCTION
1459                                    ;;*****************************************************************
1460                                    ;*
1461                                    ;*      USAGE OF THE SUBROUTINE BELOW IS AS FOLLOWS:
1462                                    ;*
1463                                    ;*      MOV     #MTPILP,$LPERR  ;PUT ADDRESS OF LOOPING LOCATION IN $LPERR
1464                                    ;*      MOV     #(NUMB),MTPIPS  ;PUT PS PREVIOUS/CURRENT MODE VALUE IN THIS LOCATION
1465                                    ;*      MOV     #TRAPRTN,MTPIVC ;LOAD THE TEST EXCLUSIVE TRAP ROUTINE TO MTPIVC
1466                                    ;*      MOV     #(NUMB),R2      ;SETUP ADDRESS IN R2
1467                                    ;*   (IT IS ASSUMED THAT ALL PDR'S/PAR'S WILL ALSO BE SET UP PROPERLY)
1468                                    ;*      JSR     PC,MTPITS       ;GO DO THE TEST
1469                                    ;*      MTPI    (MODE)          ;MTPI INSTRUCTION TO TEST IS PUT HERE
1470                                    ;*      NOP                     ;NEEDED FOR MODES 1,2,4, & 5 *ONLY*
1471                                    ;*      ERROR   +(NUMB)         ;RETURN IS HERE FOR ERROR CALLS
1472                                    ;*      TST     (SP)+           ;POP STACK - EXTRA RETURN INSTALLED BY SUBRTN NOT NEEDED
1473                                    ;*
1474                                    ;;*****************************************************************
1475 002616  005037  002614    MTPITS: CLR     ERLOOP          ;CLEAR ERROR LOOPING FLAG
1476 002622  011605                    MOV     (SP),R5         ;MOVE STACK POINTER TO R5 FOR LOADING
1477 002624  012537  002666            MOV     (R5)+,MTPILD    ;MOVE MTPI TO LOCATION
1478 002630  012537  002670            MOV     (R5)+,MTPILD+2  ;MOVE NEXT WORD TO LOCATION
1479 002634  012537  002710            MOV     (R5)+,MTPITA    ;MOVE NUMBER TO ADD TO R2 TO LOCATION
1480 002640  062716  000012            ADD     #12,(SP)        ;FUDGE RETURN TO ''TEST PASSED'' LOCATION
1481 002644  012737            MTPILP: MOV     (PC)+,@(PC)+    ;MAKE PREVIOUS MODE USER/SUPERVISOR
1482 002646  000000            MTPIPM: .WORD   0               ;LOCATION TO HOLD PREVIOUS MODE USER OR SUPER
1483 002650  177776                    .WORD   PSW             ;LOAD THE PSW
1484 002652  010046                    MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
1485 002654  105037  172310            CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
1486 002660  017737  000050  000250    MOV     @MTPIVC,MMVEC   ;SET MM TO VECTOR IN MTPIVC
1487 002666  000000  000000    MTPILD: .WORD   0,0             ;LOCATIONS USED TO PLACE THE MPTI INSTRUCTION
1488 002672  012737  015726  000250    MOV     #MGMERR,MMVEC   ;RESTORE MM VECTOR TO NORMAL ROUTINE
1489 002700  112737  000006  172310    MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
1490 002706  062702                    ADD     (PC)+,R2        ;ADD NUMBER TO R2 TO UNDO INCREMENT/DECREMENT
1491 002710  000000            MTPITA: .WORD   0               ;LOCATION TO HOLD -2, 0, +2 OR 100000
1492 002712  011201                    MOV     (R2),R1         ;READ FROM ADDRESS 60000
1493 002714  020001                    CMP     R0,R1           ;SEE IF DATA WAS STORED AT CORRECT PLACE
1494 002716  001402                    BEQ     1$              ;BRANCH IF IT WAS
1495 002720  004737  002434            JSR     PC,ERPREP       ;GO PREPARE LOCATION ERLOOP, RETURN PC & EXIT TO ERROR
1496                            ;FOR TIGHTER ERROR LOOP, REPLACE ''JSR PC,ERPREP'' WITH 'BR MTPILP''   000754
1497 002724  005737  002614    1$:     TST     ERLOOP          ;SEE IF THIS ''PASSED'' WAS IN AN ERROR LOOP
1498 002730  001345                    BNE     MTPILP          ;BRANCH BACK IF SO
1499 002732  000207                    RTS     PC              ;EXIT - TEST PASSED
1500 002734  000000            MTPIVC: .WORD   0               ;LOCATION TO HOLD MM VECTOR TO LOAD
```

```
                                        .SBTTL  SUBROUTINE TO CHECK THE MFPD INSTRUCTION
                                ;;*****************************************************************
                                ;*
                                ;*      USAGE OF THE SUBROUTINE BELOW IS AS FOLLOWS:
                                ;*
                                ;*      MOV     #MFPDLP,$LPERR  ;PUT ADDRESS OF LOOPING LOCATION IN $LPERR
                                ;*      MOV     #(NUMB),MFPDPS  ;PUT PS PREVIOUS/CURRENT MODE VALUE IN THIS LOCATION
                                ;*      MOV     #TRAPRTN,MFPDVC ;LOAD THE TEST EXCLUSIVE TRAP ROUTINE TO MFPDVC
                                ;*      MOV     #(NUMB),R2      ;SETUP ADDRESS IN R2
                                ;*    (IT IS ASSUMED THAT ALL PDR'S/PAR'S WILL ALSO BE SET UP PROPERLY)
                                ;*      JSR     PC,MFPDTS       ;GO DO THE TEST
                                ;*      MFPD    (MODE)          ;MFPD INSTRUCTION TO TEST IS PUT HERE
                                ;*      NOP                     ;NEEDED FOR MODES 1,2,4, & 5 *ONLY*
                                ;*      ERROR   +(NUMB)         ;RETURN IS HERE FOR ERROR CALLS
                                ;*      TST     (SP)+           ;POP STACK - EXTRA RETURN INSTALLED BY SUBRTN NOT NEEDED
                                ;*
                                ;;*****************************************************************
002736  005037  002614          MFPDTS: CLR     ERLOOP          ;CLEAR THE ERROR LOOPING FLAG
002742  011605                          MOV     (SP),R5         ;MOVE RETURN ADDRESS TO R5 FOR LOADING
002744  012537  002774                  MOV     (R5)+,MFPDLD    ;MOVE MFPD INSTRUCTION TO THE LOCATION BELOW
002750  012537  002776                  MOV     (R5)+,MFPDLD+2  ;MOVE NEXT WORD TO THE NEXT LOCATION
002754  062716  000010                  ADD     #10,(SP)        ;FUDGE RETURN TO THE ''TEST PASSED'' LOCATION
002760  012737                  MFPDLP: MOV     (PC)+,@(PC)+    ;SET UP PSW AS LOADED BY TEST RUNNING THIS SUBROUTINE
002762  000000                  MFPDPS: .WORD   0               ;LOCATION TO HOLD NUMBER TO LOAD THE PSW
002764  177776                          .WORD   PSW             ;LOAD THE PSW
002766  017737  000036  000250          MOV     @MFPDVC,MMVEC   ;SET M.M. VECTOR TO TRAP CATCHER OF THE TEST
002774  000000  000000          MFPDLD: .WORD   0,0             ;LOCATIONS TO HOLD THE MFPD INSTRUCTION
003000  012737  015726  000250          MOV     #MGMERR,MMVEC   ;RESTORE MM VECTOR TO NORMAL ROUTINE
003006  012601                          MOV     (SP)+,R1        ;POP K/S/U STACK INTO R1
003010  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS DATA STORED
003012  001402                          BEQ     1$              ;BRANCH IF CORRECT DATA WAS FETCHED
003014  004737  002434                  JSR     PC,ERPREP       ;GO PREPARE ERLOOP, STACK AND EXIT TO ERROR CALL
003020  005737  002614          1$:     TST     ERLOOP          ;SEE IF THIS ''PASSED'' IS IN AN ERROR LOOP
003024  001355                          BNE     MFPDLP          ;BRANCH BACK IF SO
003026  000207                          RTS     PC              ;EXIT - TEST PASSED
003030  000000                  MFPDVC: .WORD   0               ;LOCATION TO HOLD MM VECTOR OF TEST
```

```
1538                                    .SBTTL  SCOPE HANDLER ROUTINE

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

003032                                  $SCOPE:
003032  104410                                  CKSWR                           ;;TEST  FOR CHANGE IN SOFT-SWR
003034  032777  040000  176076  1$:     BIT     #BIT14,@SWR             ;;LOOP ON PRESENT TEST?
003042  001077                          BNE     $OVER                   ;;YES IF SW14=1
                                        ;#####START OF CODE FOR THE XOR TESTER#####
003044  000416                          $XTSTR: BR      6$                      ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
003046  013746  000004                          MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
003052  012737  003072  000004                  MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
003060  005737  177060                          TST     @#177060                ;;TIME OUT ON XOR?
003064  012637  000004                          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
003070  000446                                  BR      $SVLAD                  ;;GO TO THE NEXT TEST
003072  022626                          5$:     CMP     (SP)+,(SP)+             ;;CLEAR THE STACK AFTER A TIME OUT
003074  012637  000004                          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
003100  000434                                  BR      7$                      ;;LOOP ON THE PRESENT TEST
003102                                  6$:;#####END OF CODE FOR THE XOR TESTER#####
003102  032777  000400  176030                  BIT     #BIT08,@SWR             ;;LOOP ON SPEC. TEST?
003110  001404                                  BEQ     2$                      ;;BR IF NO
003112  127737  176022  001102                  CMPB    @SWR,$TSTNM             ;;ON THE RIGHT TEST?    SWR<7:0>
003120  001450                                  BEQ     $OVER                   ;;BR IF YES
003122  013737  177766  003256  2$:     MOV     177766,CPSAVE   ;MOVE CPU ERR REG VALUE TO LOC FOR TST  ;DPM001
003130  032737  000001  003256          BIT     #BIT00,CPSAVE   ;SEE IF THE POWER MONITOR BIT IS ON     ;DPM001
003136  001406                          BEQ     2000$           ;BRANCH TO CONTINUE ROUTINE IF CLEAR    ;DPM001
003140  042737  000001  177766          BIC     #BIT00,177766   ;CLEAR THE BIT FOUND TO BE SET           ;DPM001
003146  104177                          EMT     +177            ;CALL SPECIAL POWER FAIL BIT ERROR CALL ;DPM001
003150  105037  001103                  CLRB    $ERFLG          ;CLEAR THE ERROR FLAG                   ;DPM001
003154  105737  001103          2000$:  TSTB    $ERFLG                  ;;HAS AN ERROR OCCURRED?
003160  001412                          BEQ     $SVLAD                  ;;BR IF NO
003162  032777  001000  175750          BIT     #BIT09,@SWR             ;;LOOP ON ERROR?
003170  001404                          BEQ     4$                      ;;BR IF NO
003172  013737  001110  001106  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
003200  000420                          BR      $OVER
003202  105037  001103          4$:     CLRB    $ERFLG                  ;;ZERO THE ERROR FLAG
003206  105237  001102          $SVLAD: INCB    $TSTNM                  ;;COUNT TEST NUMBERS
003212  113737  001102  001230          MOVB    $TSTNM,$TESTN           ;;SET TEST NUMBER IN APT MAILBOX
003220  011637  001106                  MOV     (SP),$LPADR             ;;SAVE SCOPE LOOP ADDRESS
003224  011637  001110                  MOV     (SP),$LPERR             ;;SAVE ERROR LOOP ADDRESS
003230  005037  001212                  CLR     $ESCAPE                 ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
003234  112737  000001  001115          MOVB    #1,$ERMAX               ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
003242  013777  001102  175672  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
003250  013716  001106                  MOV     $LPADR,(SP)             ;;FUDGE RETURN ADDRESS
003254  000002                          RTI                             ;;FIXES PS
003256  000000                  CPSAVE: .WORD   0               ;LOCATION TO SAVE CPU ERR REG CONTENTS  ;DPM001
```

```
1540                                    .SBTTL  ERROR HANDLER ROUTINE

                                        ;;*****************************************************************
                                        ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                        ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                        ;*AND GO TO ERRTYP ON ERROR
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW15=1         HALT ON ERROR
                                        ;*SW13=1         INHIBIT ERROR TYPEOUTS
                                        ;*SW10=1         BELL ON ERROR
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*CALL
                                        ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER

003260  105037  003636         $ERROR: CLRB    IBSAVE          ;CLEAR THE ITEM BYTE SAVE LOCATION      ;DPM001
003264  104410                         CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
003266  010037  001162                 MOV     R0,$REG0        ;SAVE THE CONTENTS OF R0
003272  010137  001164                 MOV     R1,$REG1        ;SAVE THE CONTENTS OF R1
003276  010237  001166                 MOV     R2,$REG2        ;SAVE THE CONTENTS OF R2
003302  010337  001170                 MOV     R3,$REG3        ;SAVE THE CONTENTS OF R3
003306  010437  001172                 MOV     R4,$REG4        ;SAVE THE CONTENTS OF R4
003312  010537  001174                 MOV     R5,$REG5        ;SAVE THE CONTENTS OF R5
003316  113737  001102  001254         MOVB    $TSTNM,TESTNO   ;SAVE THE TEST NUMBER
003324  105237  001103         7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
003330  001775                         BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
003332  013777  001102  175602         MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
003340  032777  002000  175572         BIT     #BIT10,@SWR     ;;BELL ON ERROR?
003346  001402                         BEQ     1$              ;;NO - SKIP
003350  104401  001214                 TYPE    ,$BELL          ;;RING BELL
003354  005237  001112         1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
003360  011637  001116                 MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
003364  162737  000002  001116         SUB     #2,$ERRPC
003372  117737  175520  001114         MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
003400  122737  000177  001114         CMPB    #177,$ITEMB     ;SEE IF THIS IS THE POWER FAIL CALL      ;DPM001
003406  001426                         BEQ     1001$           ;BRANCH AROUND ROUTINE IF IT IS          ;DPM001
003410  105737  003636                 TSTB    IBSAVE          ;SEE IF THIS IS THE 2ND ERROR CALL       ;DPM001
003414  001021                         BNE     1000$           ;BRANCH IF SO                            ;DPM001
003416  013737  177766  003256         MOV     177766,CPSAVE   ;MOVE CPU ERR REG TO CPSAVE FOR TEST     ;DPM001
003424  032737  000001  003256         BIT     #BIT00,CPSAVE   ;SEE IF POWER MONITOR BIT IS SET         ;DPM001
003432  001414                         BEQ     1001$           ;BRANCH IF OK                            ;DPM001
003434  042737  000001  177766         BIC     #BIT00,177766   ;CLEAR THE BIT FOUND SET                 ;DPM001
003442  113737  001114  003636         MOVB    $ITEMB,IBSAVE   ;MAKE IBSAVE NON-ZERO FOR DUAL CALL      ;DPM001
003450  112737  000177  001114         MOVB    #177,$ITEMB     ;SET $ITEMB TO SPECIAL POWER FAIL PNTR   ;DPM001
003456  000402                         BR      1001$           ;BRANCH OVER IBSAVE CLEARING             ;DPM001
003460  105037  003636         1000$:  CLRB    IBSAVE          ;CLEAR IBSAVE SO AFTER 2ND ERROR, EXIT   ;DPM001
003464                         1001$:
003464  032777  020000  175446         BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
003472  001004                         BNE     20$             ;;SKIP TYPEOUTS
003474  004737  003640                 JSR     PC,ERRTYP       ;;GO TO USER ERROR ROUTINE
003500  104401  001221                 TYPE    ,$CRLF
003504                         20$:
003504  122737  000001  001244         CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
003512  001007                         BNE     2$              ;;NO,SKIP APT ERROR REPORT
003514  113737  001114  003526         MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
003522  004737  006004                 JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
003526     000         21$:     .BYTE   0
003527     000                  .BYTE   0
```

ERROR HANDLER ROUTINE

```
003530  000777                      22$:    BR      22$             ;;APT ERROR LOOP
003532  105737  003636              2$:     TSTB    IBSAVE          ;SEE IF POWER FAIL ERROR CALL           ;DPM001
003536  001005                              BNE     3$              ;BRANCH IF NOT - HALT NOT ALLOWED       ;DPM001
003540  005777  175374                      TST     @SWR            ;;HALT ON ERROR
003544  100002                              BPL     3$              ;;SKIP IF CONTINUE
003546  000000                              HALT                    ;;HALT ON ERROR!
003550  104410                              CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
003552  032777  001000  175360      3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
003560  001405                              BEQ     4$              ;;BR IF NO
003562  105737  003636                      TSTB    IBSAVE          ;SEE IF THIS IS THE PWR MNTR BIT ERROR  ;DPM001
003566  001256                              BNE     7$              ;BRANCH BACK IF SO - FUDGING NOT ALLOWED;DPM001
003570  013716  001110                      MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
003574  005737  001212              4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
003600  001405                              BEQ     5$              ;;BR IF NONE
003602  105737  003636                      TSTB    IBSAVE          ;SEE IF THIS IS THE PWR MNTR BIT ERROR  ;DPM001
003606  001246                              BNE     7$              ;BRANCH BACK IF SO - FUDGING NOT ALLOWED;DPM001
003610  013716  001212                      MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
003614                              5$:
003614  022737  036120  000042            CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
003622  001001                              BNE     6$              ;;BRANCH IF NO
003624  000000                              HALT                    ;;YES
003626                              6$:
003626  105737  003636                      TSTB    IBSAVE          ;SEE IF THIS IS THE PWR FAIL ERROR CALL ;DPM001
003632  001234                              BNE     7$              ;BRANCH BACK TO CALL ORIGINAL ERR IF SO ;DPM001
003634  000002                              RTI                     ;;RETURN
003636  000000                      IBSAVE: .WORD   0               ;LOC'N TO HOLD $ITEMB DURING DUAL ERR   ;DPM001
```

```
1542 003640  104401  001221          ERRTYP: TYPE    ,$CRLF          ;TYPE <CRLF>
1543 003644  010046                          MOV     R0,-(KSP)       ;SAVE R0.
1544 003646  005000                          CLR     R0              ;PICKUP THE ITEM INDEX
1545 003650  153700  001114                  BISB    @#$ITEMB,R0
1546 003654  001004                          BNE     1$              ;IF ITEM NUMBER IS ZERO, JUST
1547                                                                 ;TYPE THE PC OF THE ERROR
1548 003656  013746  001116                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
1549                                                                 ;;ERROR ADDRESS
1550 003662  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1551 003664  000530                          BR      13$             ;GET OUT
1552 003666  122700  000177          1$:     CMPB    #177,R0         ;SEE IF THIS ERROR CALL IS THE POWER MONITOR BIT CALL
1553 003672  001003                          BNE     100$            ;BRANCH IF NOT
1554 003674  012700  004162                  MOV     #PFECWS,R0      ;MOVE ADDRESS OF POWER MONITOR BIT ERROR TO R0
1555 003700  000406                          BR      110$            ;BRANCH TO CALL THE ERROR
1556 003702  005300                  100$:   DEC     R0              ;ADJUST THE INDEX SO THAT IT WILL
1557 003704  006300                          ASL     R0              ;WORK FOR THE ERROR TABLE.
1558 003706  006300                          ASL     R0
1559 003710  006300                          ASL     R0
1560 003712  062700  001320                  ADD     #$ERRTB,R0      ;FORM TABLE POINTER
1561 003716  012037  003726          110$:   MOV     (R0)+,2$        ;PICKUP "ERROR MESSAGE" POINTER
1562 003722  001404                          BEQ     3$              ;SKIP TYPEOUT IF NO POINTER
1563 003724  104401                          TYPE                    ;TYPE THE "ERROR MESSAGE"
1564 003726  000000                  2$:     .WORD   0               ;"ERROR MESSAGE" POINTER GOES HERE
1565 003730  104401  001221                  TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
1566 003734  012037  003744          3$:     MOV     (R0)+,4$        ;PICKUP "DATA HEADER" POINTER
1567 003740  001404                          BEQ     5$              ;SKIP TYPEOUT IF 0
1568 003742  104401                          TYPE                    ;TYPE THE "DATA HEADER"
1569 003744  000000                  4$:     .WORD   0               ;"DATA HEADER" POINTER GOES HERE
1570 003746  104401  001221                  TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
1571 003752  010146                  5$:     MOV     R1,-(KSP)       ;SAVE R1
1572 003754  012001                          MOV     (R0)+,R1        ;PICKUP "DATA TABLE" POINTER
1573 003756  001472                          BEQ     12$             ;BR IF NO DATA TO BE TYPED
1574 003760  012000                          MOV     (R0)+,R0        ;PICKUP "DATA FORMAT" POINTER
1575 003762  105710                  6$:     TSTB    (R0)            ;IS IT FORMAT 0?
1576 003764  001003                          BNE     7$              ;BR IF NO
1577                                  ;*THIS CODE IS FOR OCTAL (16-BIT) FORMAT (DF=0)
1578 003766  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FPR TYPEOUT
1579 003770  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1580 003772  000456                          BR      11$
1581                                  ;*THIS CODE IS FOR DECIMAL FORMAT (DF=1)
1582 003774  121027  000001          7$:     CMPB    (R0),#1         ;IS IT FORMAT 1?
1583 004000  001003                          BNE     8$              ;BRANCH IF NO
1584 004002  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
1585 004004  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
1586 004006  000450                          BR      11$
1587                                  ;*THIS CODE IS FOR BINARY FORMAT (DF=2)
1588 004010  121027  000002          8$:     CMPB    (R0),#2         ;IS IT FORMAT 2
1589 004014  001003                          BNE     9$              ;BRANCH IF NO
1590 004016  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
1591 004020  104406                          TYPBN                   ;;GO TYPE--BINARY ASCII
1592 004022  000442                          BR      11$
1593                                  ;*THIS CODE IS FOR OCTAL (22-BIT) FORMAT (DF=3)
1594 004024  121027  000003          9$:     CMPB    (R0),#3         ;IS IT FORMAT 3?
1595 004030  001011                          BNE     15$             ;BRANCH IF NO
1596 004032  012146                          MOV     (R1)+,-(KSP)    ;PUT ADDRESS OF FIRST LOC. ON STACK
1597 004034  004737  007056                  JSR     PC,$DB2O        ;CONVERT TWO LOCS. TO AN ASCII STRING
1598 004040  062716  000003                  ADD     #3,(KSP)        ;ONLY NEED 8 CHARACTERS NOT 11
```

```
                ;ERROR HANDLER ROUTINE
1599 004044 012637 004052              MOV     (KSP)+,10$      ;PUT ADDRESS OF ASCII CHARS. AT 10$
1600 004050 104401                     TYPE                    ;TYPE OCTAL VALUE OF 22-BIT BINARY NO.
1601 004052 000000              10$:   .WORD   0
1602                            ;*THIS CODE IS FOR OCTAL (22-BIT) FORMAT FOR A PAR LEFT SHIFTED 6 (DF-4)
1603 004054 010246              15$:   MOV     R2,-(KSP)       ;SAVE R2 ON STACK
1604 004056 010346                     MOV     R3,-(KSP)       ;SAVE R3 ON STACK
1605 004060 013103                     MOV     @(R1)+,R3       ;LOAD DATA WORD INTO R3
1606 004062 005002                     CLR     R2              ;R2 HOLDS UPPER SIX BITS OF NUMBER
1607 004064 073227 000006              ASHC    #6,R2           ;SHIFT VALUE LEFT 6 TIMES
1608 004070 010237 001206              MOV     R2,$TMP4        ;HOLDS LOWER 16 BITS OF ADDRESS
1609 004074 010337 001210              MOV     R3,$TMP5        ;HOLDS UPPER 6 BITS OF ADDRESS
1610 004100 012746 001206              MOV     #$TMP4,-(KSP)   ;PUT ADDRESS OF LOWER BITS ONTO STACK
1611 004104 004737 007056              JSR     PC,$DB2O        ;CONVERT TWO LOCS. TO AN ASCII STRING
1612 004110 062716 000003              ADD     #3,(KSP)        ;ONLY NEED 8 CHARACTERS NOT 11
1613 004114 012637 004122              MOV     (KSP)+,16$      ;PUT ADDRESS OF ASCII CHARS. AT 16$
1614 004120 104401                     TYPE                    ;TYPE OCTAL VALUE OF 22-BIT BINARY NO.
1615 004122 000000              16$:   .WORD   0
1616 004124 012603                     MOV     (KSP)+,R3       ;RESTORE R3
1617 004126 012602                     MOV     (KSP)+,R2       ;RESTORE R2
1618 004130 005711              11$:   TST     (R1)            ;IS THERE ANOTHER NUMBER?
1619 004132 001404                     BEQ     12$             ;BR IF NO
1620 004134 104401 004156              TYPE    ,14$            ;TYPE TWO(2) SPACES
1621 004140 105720                     TSTB    (R0)+           ;POINT TO NEW 'DATA FORMAT'
1622 004142 000707                     BR      6$              ;LOOP
1623 004144 012601              12$:   MOV     (KSP)+,R1       ;RESTORE R1
1624 004146 012600              13$:   MOV     (KSP)+,R0       ;RESTORE R0
1625 004150 104401 001221              TYPE    ,$CRLF          ;''CARRIAGE RETURN'' & 'LINE FEED''
1626 004154 000207                     RTS     PC              ;RETURN
1627 004156    040    040    000  14$:   .ASCIZ  /  /           ;TWO(2) SPACES
1628 004161    000                     .BYTE   0
1629 004162 004172 004232 004262 PFECWS: .WORD   PFECEM,PFECDH,PFECDT,PFECDF
     004170 004272
1630 004172    120    117    127 PFECEM: .ASCIZ  ?POWER MONITOR BIT WAS FOUND SET?
     004175    105    122    040
     004200    115    117    116
     004203    111    124    117
     004206    122    040    102
     004211    111    124    040
     004214    127    101    123
     004217    040    106    117
     004222    125    116    104
     004225    040    123    105
     004230    124    000
1631 004232    124    105    123 PFECDH: .ASCIZ  ?TESTNO  ERR PC  CPUERR?
     004235    124    116    117
     004240    040    040    105
     004243    122    122    040
     004246    120    103    040
     004251    040    103    120
     004254    125    105    122
     004257    122    000
1632                                   .EVEN
1633 004262 001230 001116 003256 PFECDT: .WORD   $TESTN,$ERRPC,CPSAVE,0
     004270 000000
1634 004272    000    000    000 PFECDF: .BYTE   0,0,0,0
     004275    000
```

```
1635                                    .SBTTL  TTY INPUT ROUTINE
                                        ;;*****************************************************************
                                        .ENABL  LSB
                                        ;;*****************************************************************
                                        ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                        ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
                                        ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
                                        ;*WHEN OPERATING IN TTY FLAG MODE.
004276  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
004304  001114                        BNE     15$             ;;BRANCH IF NO
004306  105777  174632                TSTB    @$TKS           ;;CHAR THERE?
004312  100111                        BPL     15$             ;;IF NO, DON'T WAIT AROUND
004314  117746  174626                MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
004320  042716  177600                BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
004324  022726  000007                CMP     #7,(SP)+        ;;IS IT A CONTROL G?
004330  001102                        BNE     15$             ;;NO, RETURN TO USER
004332  123727  001134  000001        CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
004340  001476                        BEQ     15$             ;;BRANCH IF YES
004342  104401  005243                TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
004346  104401  005250        $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
004352  013746  000176                MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
004356  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
004360  104401  005261                TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
004364  005046                19$:    CLR     -(SP)           ;;CLEAR COUNTER
004366  005046                        CLR     -(SP)           ;;THE NEW SWR
004370  105777  174550        7$:     TSTB    @$TKS           ;;CHAR THERE?
004374  100375                        BPL     7$              ;;IF NOT TRY AGAIN
004376  117746  174544                MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
004402  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
004406  021627  000003                CMP     (SP),#3         ;;IS IT A CONTROL-C?
004412  001015                        BNE     9$              ;;BRANCH IF NOT
004414  104401  001312                TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
004420  062706  000006                ADD     #6,SP           ;;CLEAN UP STACK
004424  123727  001135  000001        CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
004432  001003                        BNE     8$              ;;BRANCH IF NO
004434  012777  000100  174502        MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
004442  000137  005272        8$:     JMP     CNTRLC          ;;CONTROL-C RESTART
004446  021627  000025        9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
004452  001005                        BNE     10$             ;;BRANCH IF NOT
004454  104401  005236                TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
004460  062706  000006        20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
004464  000737                        BR      19$             ;;LET'S TRY IT AGAIN
004466  021627  000015        10$.    CMP     (SP),#15        ;;IS IT A <CR>?
004472  001022                        BNE     16$             ;;BRANCH IF NO
004474  005766  000004                TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
004500  001403                        BEQ     11$             ;;BRANCH IF YES
004502  016677  000002  174430        MOV     2(SP),@SWR      ;;SAVE NEW SWR
004510  062706  000006        11$:    ADD     #6,SP           ;;CLEAR UP STACK
004514  104401  001221        14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
004520  123727  001135  000001        CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
004526  001003                        BNE     15$             ;;BRANCH IF NOT
004530  012777  000100  174406        MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
004536  000002                15$:    RTI                     ;;RETURN
004540  004737  005634        16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
004544  021627  000060                CMP     (SP),#60        ;;CHAR < 0?
004550  002420                        BLT     18$             ;;BRANCH IF YES
004552  021627  000067                CMP     (SP),#67        ;;CHAR > 7?
```

```
004556  003015                          BGT     18$             ;;BRANCH IF YES
004560  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
004564  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
004570  001403                          BEQ     17$             ;;BRANCH IF YES
004572  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
004574  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
004576  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
004600  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
004604  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
004610  000667                          BR      7$              ;;GET THE NEXT ONE
004612  104401  001220          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
004616  000720                          BR      20$             ;;SIMULATE CONTROL-U
                                .DSABL  LSB
                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*CALL:
                                ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
                                ;
004620  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
004622  016666  000004  000002  1$:     MOV     4(SP),2(SP)     ;;SAVE THE PS
004630  105777  174310          1$:     TSTB    @$TKS           ;;WAIT FOR
004634  100375                          BPL     1$              ;;A CHARACTER
004636  117766  174304  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
004644  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
004652  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
004660  001013                          BNE     3$              ;;BRANCH IF NO
004662  105777  174256          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
004666  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
004670  117746  174252                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
004674  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
004700  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
004704  001366                          BNE     2$              ;;IF NOT DISCARD IT
004706  000750                          BR      1$              ;;YES, RESUME
004710  026627  000004  000021  3$:     CMP     4(SP),#$XON     ;;IS IT A RANDOM XON?                   ;RAN001
004716  001744                          BEQ     1$              ;;BRANCH IF YES                         ;RAN001
004720  026627  000004  000140          CMP     4(SP),#140      ;;IS IT UPPER CASE?
004726  002407                          BLT     4$              ;;BRANCH IF YES
004730  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
004736  003003                          BGT     4$              ;;BRANCH IF YES
004740  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
004746  000002                  4$:     RTI                     ;;GO BACK TO USER
                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                ;*CALL:
                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                ;*      '                       ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
004750  010346          $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
004752  005046                  CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
004754  012703  005226  1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
004760  022703  005236  2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
004764  101467                  BLOS    4$              ;;BR IF YES
004766  104411                  RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
004770  112613                  MOVB    (SP)+,(R3)      ;;GET CHARACTER
004772  122713  000003          CMPB    #3,(R3)         ;;IS IT A CONTROL-C?
```

```
004776  001006                          BNE     10$             ;;BRANCH IF NO
005000  104401  001312                  TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
005004  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY OFF OF THE STACK
005006  012603                          MOV     (SP)+,R3        ;;RESTORE R3
005010  000137  005272                  JMP     CNTRLC          ;;GOTO CONTROL-C RESTART
005014  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
005020  001022                          BNE     5$              ;;BR IF NO
005022  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
005024  001007                          BNE     6$              ;;BR IF NO
005026  112737  000134  005224          MOVB    #'\,9$          ;;TYPE A BACK SLASH
005034  104401  005224                  TYPE    ,9$
005040  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
005044  005303                  6$:     DEC     R3              ;;BACKUP BY ONE
005046  020327  005226                  CMP     R3,#$TTYIN      ;;STACK EMPTY?
005052  103434                          BLO     4$              ;;BR IF YES
005054  111337  005224                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
005060  104401  005224                  TYPE    ,9$             ;;GO TYPE
005064  000735                          BR      2$              ;;GO READ ANOTHER CHAR.
005066  005716                  5$:     TST     (SP)            ;;RUBOUT KEY SET?
005070  001406                          BEQ     7$              ;;BR IF NO
005072  112737  000134  005224          MOVB    #'\,9$          ;;TYPE A BACK SLASH
005100  104401  005224                  TYPE    ,9$
005104  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
005106  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
005112  001003                          BNE     8$              ;;BR IF NO
005114  104401  005236                  TYPE    ,$CNTLU         ;;TYPE A CONTROL 'U'
005120  000715                          BR      1$              ;;GO START OVER
005122  122713  000022          8$:     CMPB    #22,(R3)        ;;IS CHARACTER A '^R'?
005126  001011                          BNE     3$              ;;BRANCH IF NO
005130  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
005132  104401  001221                  TYPE    ,$CRLF          ;;TYPE A 'CR' & 'LF'
005136  104401  005226                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
005142  000706                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
005144  104401  001220          4$:     TYPE    ,$QUES          ;;TYPE A '?'
005150  000701                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
005152  111337  005224          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
005156  104401  005224                  TYPE    ,9$
005162  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
005166  001274                          BNE     2$              ;;LOOP IF NOT RETURN
005170  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
005174  104401  001222                  TYPE    ,$LF            ;;TYPE A LINE FEED
005200  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
005202  012603                          MOV     (SP)+,R3        ;;RESTORE R3
005204  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
005206  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
005214  012766  005226  000004          MOV     #$TTYIN,4(SP)
005222  000002                          RTI                     ;;RETURN
005224     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
005225     000                          .BYTE   0               ;;TERMINATOR
005226                          $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
005236     136     125     015  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL 'U'
005241     012     000
005243     136     107     015  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL 'G'
005246     012     000
005250     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR = /
005253     127     122     040
005256     075     040     000
```

```
005261  040  040  116  SMNEW:  .ASCIZ  /  NEW = /
005264  105  127  040
005267  075  040  000
```

```
1637                                        .SBTTL  CONTROL-C SERVICING ROUTINE
1638
1639 005272  013737  001232  001210  CNTRLC: MOV     $PASS,$TMP5     ;GET THE VALUE OF '$PASS'
1640 005300  005237  001210                  INC     $TMP5           ;FORM CURRENT PASS #
1641 005304  104401  005351                  TYPE    ,CMSG           ;TYPE THE TEST STOPS HERE
1642 005310  113737  001102  005344          MOVB    $TSTNM,1$       ;SAVE TEST NUMBER
1643 005316  013746  005344                  MOV     1$,-(SP)        ;SAVE 1$ FO TYPEOUT
1644 005322  104402                          TYPOC
1645 005324  104401  005346                  TYPE    ,2$
1646 005330  013746  001210                  MOV     $TMP5,-(SP)     ;SAVE $TMP5 FOR TYPEOUT
1647 005334  104405                          TYPDS                   ;TYPE ASCII DECIMAL WITH SIGN
1648 005336  104407                          GTSWR                   ;ASK FOR NEW SWR VALUE
1649 005340  000137  035720                  JMP     $EOP+2          ;JUMP TO END OF PASS + 2
1650 005344  000000                  1$:     .WORD   0               ;TEST # BUFFER
1651 005346     040     040     000  2$:     .ASCIZ  /  /            ;2 SPACES & STOP MESSAGE
1652 005351     112     125     115  CMSG:   .ASCII  /JUMPING TO END OF PASS/<15><12>
     005354     120     111     116
     005357     107     040     124
     005362     117     040     105
     005365     116     104     040
     005370     117     106     040
     005373     120     101     123
     005376     123     015     012
1653 005401     124     105     123          .ASCIZ  /TESTNO PASSNO/<15><12>
     005404     124     116     117
     005407     011     120     101
     005412     123     123     116
     005415     117     015     012
     005420     000
1654                                        .EVEN
```

```
1656                                    .SBTTL  TYPE ROUTINE
                                        ;;*****************************************************************
                                        ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                                        ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                                        ;*NOTE1:         $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                                        ;*NOTE2:         $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                        ;*NOTE3:         $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                                        ;*
                                        ;*CALL:
                                        ;*1) USING A TRAP INSTRUCTION
                                        ;*       TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                                        ;*OR
                                        ;*       TYPE
                                        ;*       MESADR
                                        ;*
005422  105737  001157          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
005426  100002                          BPL     1$              ;;BR IF YES
005430  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
005432  000430                          BR      3$              ;;LEAVE
005434  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
005436  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
005442  122737  000001  001244          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
005450  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
005452  132737  000100  001245          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
005460  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
005462  010037  005472                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
005466  004737  005774                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
005472  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
005474  132737  000040  001245  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
005502  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
005504  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
005506  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
005510  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
005512  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
005514  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
005520  000002                          RTI                     ;;RETURN
005522  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
005526  001430                          BEQ     8$
005530  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
005534  001006                          BNE     5$
005536  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
005540  104401                          TYPE                    ;;TYPE A CR AND LF
005542  001221                          $CRLF
005544  105037  005762                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
005550  000755                          BR      2$              ;;GET NEXT CHARACTER
005552  004737  00563'          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
005556  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
005562  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
005564  013746  001154                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
005570  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
005574  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
005576  004737  005634                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
005602  105337  005762                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
005606  000770                          BR      7$              ;;LOOP
                                        ;HORIZONTAL TAB PROCESSOR
005610  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
```

```
005614  004737  005634          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
005620  132737  000007  005762          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
005626  001372                          BNE     9$              ;;TAB STOP
005630  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
005632  000724                          BR      2$              ;;GET NEXT CHARACTER
005634                          $TYPEC:
005634  105777  173304                  TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?          ;MJD001
005640  100022                          BPL     10$             ;;BR IF NOT                     ;MJD001
005642  017746  173300                  MOV     @$TKB,-(SP)     ;;GET CHAR                      ;MJD001
005646  042716  177600                  BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS         ;MJD001
005652  122716  000023                  CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF                 ;MJD001
005656  001012                          BNE     102$            ;;BR IF NOT                     ;MJD001
005660                          101$:                                                           ;MJD001
005660  105777  173260                  TSTB    @$TKS           ;;WAIT FOR CHAR                 ;MJD001
005664  100375                          BPL     101$                                            ;MJD001
005666  117716  173254                  MOVB    @$TKB,(SP)      ;;GET CHAR                      ;MJD001
005672  042716  177600                  BIC     #177600,(SP)    ;;STRIP IT                      ;MJD001
005676  122716  000021                  CMPB    #$XON,(SP)      ;;WAS IT XON?                   ;MJD001
005702  001366                          BNE     101$            ;;BR IF NOT                     ;MJD001
005704                          102$:                                                           ;MJD001
005704  005726                          TST     (SP)+           ;;FIX STACK                     ;MJD001
005706                          10$:                                                            ;MJD001
005706  105777  173236                  TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
005712  100375                          BPL     10$                                             ;MJD001
005714  126627  000002  000021          CMPB    2(SP),#$XON     ;;IS CHARACTER A RANDOM XON?    ;RAN001
005722  001420                          BEQ     $TYPEX          ;;BRANCH IF YES                 ;RAN001
005724  116677  000002  173220          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
005732  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
005740  001003                          BNE     1$              ;;BRANCH IF NO
005742  105037  005762                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
005746  000406                          BR      $TYPEX          ;;EXIT
005750  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
005756  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
005760  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
005762  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
005764  000207          $TYPEX: RTS     PC
```

```
1657                                    .SBTTL  APT COMMUNICATIONS ROUTINE
                                        ;;*****************************************************************
005766  112737  000001  006232  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
005774  112737  000001  006230  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
006002  000403                        BR      $ATYC
006004  112737  000001  006232  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
006012                          $ATYC:
006012  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
006014  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
006016  105737  006230                TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
006022  001450                        BEQ     5$              ;;IF NOT:  BR
006024  122737  000001  001244        CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
006032  001031                        BNE     3$              ;;IF NOT:  BR
006034  132737  000100  001245        BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
006042  001425                        BEQ     3$              ;;IF NOT:  BR
006044  017600  000004                MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
006050  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
006056  005737  001224          1$:   TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
006062  001375                        BNE     1$              ;;IF NOT:  WAIT
006064  010037  001240                MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
006070  105720                  2$:   TSTB    (R0)+           ;;FIND END OF MESSAGE
006072  001376                        BNE     2$
006074  163700  001240                SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
006100  006200                        ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
006102  010037  001242                MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
006106  012737  000004  001224        MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
006114  000413                        BR      5$
006116  017637  000004  006142  3$:   MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
006124  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
006132  013746  177776                MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
006136  004737  005422                JSR     PC,$TYPE        ;;CALL TYPE MACRO
006142  000000                  4$:   .WORD   0
006144                          5$:
006144  105737  006232          10$:  TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
006150  001416                        BEQ     12$             ;;IF NOT:  BR
006152  005737  001244                TST     $ENV            ;;RUNNING UNDER APT?
006156  001413                        BEQ     12$             ;;IF NOT:  BR
006160  005737  001224          11$:  TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
006164  001375                        BNE     11$             ;;IF NOT:  WAIT
006166  017637  000004  001226        MOV     @4(SP),$FATAL   ;;GET ERROR #
006174  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
006202  005237  001224                INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
006206  105037  006232          12$:  CLRB    $FFLG           ;;CLEAR FATAL FLAG
006212  105037  006231                CLRB    $LFLG           ;;CLEAR LOG FLAG
006216  105037  006230                CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
006222  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
006224  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
006226  000207                        RTS     PC              ;;RETURN
006230     000                $MFLG:  .BYTE   0               ;;MESSG. FLAG
006231     000                $LFLG:  .BYTE   0               ;;LOG FLAG
006232     000                $FFLG:  .BYTE   0               ;;FATAL FLAG
                                        .EVEN
            000200                  APTSIZE=200
            000001                  APTENV=001
            000100                  APTSPOOL=100
            000040                  APTCSUP=040
```

```
 1658                                   .SBTTL  BINARY TO ASCII AND TYPE ROUTINE
                                        ;;*****************************************************************
                                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16-BIT
                                        ;*BINARY-ASCII NUMBER AND TYPE IT.
                                        ;*CALL:
                                        ;*      MOV     NUMBER,-(SP)    ;;NUMBER TO BE TYPED
                                        ;*      TYPBN                   ;;TYPE IT
006234  010146                          $TYPBN: MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
006236  016601  000006                          MOV     6(SP),R1        ;;GET THE INPUT NUMBER
006242  000261                                  SEC                     ;;SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
006244  112737  000060  006306          1$:     MOVB    #'0,$BIN        ;;SET CHARACTER TO AN ASCII "0".
006252  006101                                  ROL     R1              ;;GET THIS BIT
006254  001406                                  BEQ     2$              ;;DONE?
006256  105537  006306                          ADCB    $BIN            ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
006262  104401  006306                          TYPE    ,$BIN           ;;GO TYPE THIS BIT
006266  000241                                  CLC                     ;;CLEAR "C" SO CAN KEEP TRACK OF BITS
006270  000765                                  BR      1$              ;;GO DO THE NEXT BIT
006272  012601                          2$:     MOV     (SP)+,R1        ;;POP THE STACK INTO R1
006274  016666  000002  000004                  MOV     2(SP),4(SP)     ;;ADJUST THE STACK
006302  012616                                  MOV     (SP)+,(SP)
006304  000002                                  RTI                     ;;RETURN TO USER
006306     000     000          $BIN:   .BYTE   0,0             ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
```

```
1659                                    .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
                                        ;;*****************************************************************
                                        ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                        ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                        ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                        ;*      .BYTE   N               ;;N 1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                        ;*      .BYTE   M               ;;M=1 OR 0
                                        ;*                                      ;;1=TYPE LEADING ZEROS
                                        ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                        ;*
                                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                        ;*$TYPOS OR $TYPOC
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                                        ;*
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT
006310  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
006314  116637  000001  006533          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
006322  112637  006535                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
006326  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
006332  000406                          BR      $TYPON
006334  112737  000001  006533  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
006342  112737  000006  006535          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
006350  112737  000005  006532  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
006356  010346                          MOV     R3,-(SP)        ;;SAVE R3
006360  010446                          MOV     R4,-(SP)        ;;SAVE R4
006362  010546                          MOV     R5,-(SP)        ;;SAVE R5
006364  113704  006535                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
006370  005404                          NEG     R4
006372  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
006376  110437  006534                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
006402  113704  006533                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
006406  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
006412  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
006414  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
006416  000404                          BR      3$              ;;GO DO MSB
006420  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
006422  006105                          ROL     R5
006424  006105                          ROL     R5
006426  010503                          MOV     R5,R3
006430  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
006432  105337  006534                  DECB    $OMODE          ;;TYPE THIS DIGIT?
006436  100016                          BPL     7$              ;;BR IF NO
006440  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
006444  001002                          BNE     4$              ;;TEST FOR 0
006446  005704                          TST     R4              ;;SUPPRESS THIS 0?
006450  001403                          BEQ     5$              ;;BR IF YES
006452  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
006454  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
006460  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
```

BINARY TO OCTAL (ASCII) AND TYPE

```
006464  110337  006530                  MOVB    R3,8$           ;;SAVE FOR TYPING
006470  104401  006530                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
006474  105337  006532          7$:     DECB    $OCNT           ;;COUNT BY 1
006500  003347                          BGT     2$              ;;BR IF MORE TO DO
006502  002402                          BLT     6$              ;;BR IF DONE
006504  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
006506  000744                          BR      2$              ;;GO DO THE LAST DIGIT
006510  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
006512  012604                          MOV     (SP)+,R4        ;;RESTORE R4
006514  012603                          MOV     (SP)+,R3        ;;RESTORE R3
006516  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
006524  012616                          MOV     (SP)+,(SP)
006526  000002                          RTI                     ;;RETURN
006530     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
006531     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
006532     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
006533     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
006534  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
1660                                    .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
006536                                  $TYPDS:
006536  010046                                 MOV     R0,-(SP)        ;;PUSH R0 ON STACK
006540  010146                                 MOV     R1,-(SP)        ;;PUSH R1 ON STACK
006542  010246                                 MOV     R2,-(SP)        ;;PUSH R2 ON STACK
006544  010346                                 MOV     R3,-(SP)        ;;PUSH R3 ON STACK
006546  010546                                 MOV     R5,-(SP)        ;;PUSH R5 ON STACK
006550  012746  020200                        MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
006554  016605  000020                        MOV     20(SP),R5       ;;GET THE INPUT NUMBER
006560  100004                                BPL     1$              ;;BR IF INPUT IS POS.
006562  005405                                NEG     R5              ;;MAKE THE BINARY NUMBER POS.
006564  112766  000055  000001                MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
006572  005000                          1$:   CLR     R0              ;;ZERO THE CONSTANTS INDEX
006574  012703  006752                        MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
006600  112723  000040                        MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
006604  005002                          2$:   CLR     R2              ;;CLEAR THE BCD NUMBER
006606  016001  006742                        MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
006612  160105                          3$:   SUB     R1,R5           ;;FORM THIS BCD DIGIT
006614  002402                                BLT     4$              ;;BR IF DONE
006616  005202                                INC     R2              ;;INCREASE THE BCD DIGIT BY 1
006620  000774                                BR      3$
006622  060105                          4$:   ADD     R1,R5           ;;ADD BACK THE CONSTANT
006624  005702                                TST     R2              ;;CHECK IF BCD DIGIT=0
006626  001002                                BNE     5$              ;;FALL THROUGH IF 0
006630  105716                                TSTB    (SP)            ;;STILL DOING LEADING 0'S?
006632  100407                                BMI     7$              ;;BR IF YES
006634  106316                          5$:   ASLB    (SP)            ;;MSD?
006636  103003                                BCC     6$              ;;BR IF NO
006640  116663  000001  177777                MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
006646  052702  000060                  6$:   BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
006652  052702  000040                  7$:   BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
006656  110223                                MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
006660  005720                                TST     (R0)+           ;;JUST INCREMENTING
006662  020027  000010                        CMP     R0,#10          ;;CHECK THE TABLE INDEX
006666  002746                                BLT     2$              ;;GO DO THE NEXT DIGIT
006670  003002                                BGT     8$              ;;GO TO EXIT
006672  010502                                MOV     R5,R2           ;;GET THE LSD
006674  000764                                BR      6$              ;;GO CHANGE TO ASCII
006676  105726                          8$:   TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
006700  100003                                BPL     9$              ;;BR IF NO
006702  116663  177777  177776                MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
006710  105013                          9$:   CLRB    (R3)            ;;SET THE TERMINATOR
006712  012605                                MOV     (SP)+,R5        ;;POP STACK INTO R5
006714  012603                                MOV     (SP)+,R3        ;;POP STACK INTO R3
006716  012602                                MOV     (SP)+,R2        ;;POP STACK INTO R2
006720  012601                                MOV     (SP)+,R1        ;;POP STACK INTO R1
006722  012600                                MOV     (SP)+,R0        ;;POP STACK INTO R0
006724  104401  006752                        TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
```

```
006730  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
006736  012616                          MOV     (SP)+,(SP)
006740  000002                          RTI                     ;;RETURN TO USER
006742  023420                  $DTBL:  10000.
006744  001750                          1000.
006746  000144                          100.
006750  000012                          10.
006752                          $DBLK:  .BLKW   4
```

SAVE AND RESTORE R0-R5 ROUTINES

```
                        .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
                        ;;*****************************************************************
                        ;*SAVE R0-R5
                        ;*CALL:
                        ;*      SAVREG
                        ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
                        ;*
                        ;*TOP---(+16)
                        ;* +2---(+18)
                        ;* +4---R5
                        ;* +6---R4
                        ;* +8---R3
                        ;*+10---R2
                        ;*+12---R1
                        ;*+14---R0
006762                  $SAVREG:
006762  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
006764  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
006766  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
006770  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
006772  010446                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
006774  010546                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
006776  016646  000022          MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
007002  016646  000022          MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
007006  016646  000022          MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
007012  016646  000022          MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
007016  000002                  RTI
                        ;*RESTORE R0-R5
                        ;*CALL:
                        ;*      RESREG
007020                  $RESREG:
007020  012666  000022          MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
007024  012666  000022          MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
007030  012666  000022          MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
007034  012666  000022          MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
007040  012605                  MOV     (SP)+,R5        ;;POP STACK INTO R5
007042  012604                  MOV     (SP)+,R4        ;;POP STACK INTO R4
007044  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
007046  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
007050  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
007052  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
007054  000002                  RTI
```

```
1662                                    .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
                                        ;;*****************************************************************
                                        ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
                                        ;*UNSIGNED OCTAL ASCIZ NUMBER.
                                        ;*CALL
                                        ;*      MOV     #PNTR,-(SP)     ;;POINTER TO LOW WORD OF BINARY NUMBER
                                        ;*      JSR     PC,@#$DB20      ;;CALL THE ROUTINE
                                        ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK
007056  104413                  $DB20:  SAVREG                  ;;SAVE ALL REGISTERS
007060  016601  000002                  MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
007064  012705  007175                  MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
007070  012704  000014                  MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
007074  012703  177770                  MOV     #^C7,R3         ;;MASK
007100  012100                          MOV     (R1)+,R0        ;;LOWER WORD
007102  012101                          MOV     (R1)+,R1        ;;HIGH WORD
007104  005002                          CLR     R2              ;;TERMINATOR
007106  110245                  1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
007110  010002                          MOV     R0,R2           ;;GET THIS DIGIT
007112  005304                          DEC     R4              ;;COUNT THIS CHARACTER
007114  003007                          BGT     3$              ;;BR IF NOT THE LAST DIGIT
007116  001405                          BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
007120  005205                          INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
007122  010566  000002                  MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
007126  104414                          RESREG                  ;;RESTORE ALL REGISTERS
007130  000207                          RTS     PC              ;;RETURN TO USER
007132  006203                  2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
007134  006001                  3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
007136  006000                          ROR     R0              ;;      THE NEXT OCTAL DIGIT
007140  006001                          ROR     R1
007142  006000                          ROR     R0
007144  006001                          ROR     R1
007146  006000                          ROR     R0
007150  040302                          BIC     R3,R2           ;;MASK OUT ALL JUNK
007152  062702  000060                  ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
007156  000753                          BR      1$              ;;GO PUT IT IN THE DATA TABLE
007160                          $OCTVL: .BLKB   14.             ;;RESERVE DATA TABLE
```

```
1663                                    .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.
007176  010046                          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
007200  016600  000002                          MOV     2(SP),R0        ;;GET TRAP ADDRESS
007204  005740                          TST     -(R0)           ;;BACKUP BY 2
007206  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
007210  006300                          ASL     R0              ;;POSITION FOR INDEXING
007212  016000  007232                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
007216  000200                          RTS     R0              ;;GO TO ROUTINE
                                        ;;THIS IS USE TO HANDLE THE ''GETPRI'' MACRO
007220  011646                          $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
007222  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
007230  000002                          RTI                     ;;RESTORE THE PSW
                                        .SBTTL  TRAP TABLE
                                        ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                        ;*BY THE ''TRAP'' INSTRUCTION.
                                        ;       ROUTINE
                                        ;       -------
007232  007220                          $TRPAD: .WORD   $TRAP2
007234  005422                                  $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
007236  006334                                  $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
007240  006310                                  $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
007242  006350                                  $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
007244  006536                                  $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
007246  006234                                  $TYPBN  ;;CALL=TYPBN    TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER
007250  004346                                  $GTSWR  ;;CALL=GTSWR    TRAP+7(104407)  GET SOFT-SWR SETTING
007252  004276                                  $CKSWR  ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
007254  004620                                  $RDCHR  ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
007256  004750                                  $RDLIN  ;;CALL=RDLIN    TRAP+12(104412) TTY TYPEIN STRING ROUTINE
007260  006762                                  $SAVREG ;;CALL SAVREG   TRAP+13(104413) SAVE R0-R5 ROUTINE
007262  007020                                  $RESREG ;;CALL-RESREG   TRAP+14(104414) RESTORE R0-R5 ROUTINE
```

```
1664                                    .SBTTL  POWER DOWN AND UP ROUTINES
                                        ;;*****************************************************************
                                        ;POWER DOWN ROUTINE
007264  012737  007442  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
007272  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
007300  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
007302  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
007304  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
007306  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
007310  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
007312  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
007314  017746  171620                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
007320  010637  007446                  MOV     SP,$SAVR6       ;;SAVE SP
007324  012737  007336  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
007332  000000                          HALT
007334  000776                          BR      .-2             ;;HANG UP
                                        ;;*****************************************************************
                                        ;POWER UP ROUTINE
007336  012737  007442  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
007344  013706  007446                  MOV     $SAVR6,SP       ;;GET SP
007350  005037  007446                  CLR     $SAVR6          ;;WAIT LOUP FOR THE TTY
007354  005237  007446          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
007360  001375                          BNE     1$              ;;OF  WORD
007362  012677  171552                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
007366  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
007370  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
007372  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
007374  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
007376  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
007400  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
007402  012737  007264  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
007410  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
007416  104401                          TYPE                    ;;REPORT THE POWER FAILURE
007420  007450          $PWRMG: .WORD   PWRMSG                  ;;POWER FAIL MESSAGE POINTER
007422  012716                  MOV     (PC)+,(SP)              ;;RESTART AT START
007424  020000          $PWRAD: .WORD   START                   ;;RESTART ADDRESS
007426  042766  000020  000002          BIC     #20,2(SP)       ;;CLEAR "T" BIT
007434  005037  001310                  CLR     $TBIT           ;;CLEAR THE "T" BIT FLAG
007440  000002                          RTI
007442  000000          $ILLUP: HALT                            ;;THE POWER UP SEQUENCE WAS STARTED
007444  000776                  BR      .-2                     ;; BEFORE THE POWER DOWN WAS COMPLETE
007446  000000          $SAVR6: 0                               ;;PUT THE SP HERE
1665 007450     012     015     040  PWRMSG: .ASCIZ  <12><15>? POWER FAILURE - RESTARTING ?<12><15>
     007453     120     117     127
     007456     105     122     040
     007461     106     101     111
     007464     114     125     122
     007467     105     040     055
     007472     040     122     105
     007475     123     124     101
     007500     122     124     111
     007503     116     107     040
     007506     012     015     000
1666                            .EVEN
```

```
1668                                    .SBTTL  ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS
1669                                            .NLIST  BEX
1670 007512    125     116     105  EM1:      .ASCIZ  /UNEXPECTED CPU TRAP TO LOC. 004/
1671 007552    125     116     105  EM2:      .ASCIZ  /UNEXPECTED MEM. MGMT. TRAP TO LOC. 250/
1672 007621    115     105     115  EM10:     .ASCIZ  /MEMORY MGMT. ACCESS ABORT DID NOT OCCUR/
1673 007671    101     103     103  EM11:     .ASCIZ  /ACCESS ERROR DID NOT ABORT INSTRUCTION/
1674 007740    123     122     060  EM12:     .ASCIZ  /SR0 DID NOT REPORT ACCESS ERROR CORRECTLY/
1675 010012    123     122     062  EM13:     .ASCIZ  /SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR./
1676 010063    120     101     107  EM14:     .ASCIZ  /PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE/
1677 010144    120     101     107  EM15:     .ASCIZ  /PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE/
1678 010227    123     122     060  EM16:     .ASCIZ  /SR0 DID NOT REPORT PAGE LGTH. ABORT CORRECTLY/
1679 010305    123     122     060  EM21:     .ASCIZ  /SR0 OR SR2 CHANGED BY A SECOND ABORT/
1680 010352    123     122     060  EM22:     .ASCIZ  /SR0 OR SR2 WERE NOT 'RESET'' BY A RESET/
1681 010421    123     122     062  EM23:     .ASCIZ  /SR2 NOT TRACKING CORRECTLY/
1682 010454    104     111     104  EM24:     .ASCIZ  /DID NOT TRAP THRU KERNEL SPACE/
1683 010513    113     124     040  EM25:     .ASCIZ  /KT ERROR SERVICED ON ODD ADDR. ERROR/
1684 010560    123     122     060  EM26:     .ASCIZ  /SR0 OR SR2 CHANGED BY ODD ADDR. ERROR/
1685 010626    105     122     122  EM27:     .ASCIZ  /ERROR DURING 'DOUBLE ERROR'' (KT & ODD ADDR.)/
1686 010703    115     106     120  EM30:     .ASCIZ  /MFPI INSTRUCTION PUSHED WRONG DATA/
1687 010746    115     124     120  EM31:     .ASCIZ  /MTPI INSTRUCTION LOADED WRONG DATA/
1688 011011    123     124     101  EM32:     .ASCIZ  /STACK NOT PUSHED BY MFPI-MTPI/
1689 011047    113     105     122  EM33:     .ASCIZ  /KERNEL PAGE ACCESS INSTEAD OF USER: MFPI-MTPI/
1690 011125    115     056     115  EM34:     .ASCIZ  /M.M. ABORT IN KERNAL D-SPACE HAD WRONG CONDITION/
1691 011206    111     114     114  EM35:     .ASCIZ  /ILLEGAL MODE 10 NOT ABORTED/
1692 011242    123     122     060  EM36:     .ASCIZ  /SR0 DID NOT REPORT ILLEGAL MODE 10 CORRECTLY/
1693 011317    120     123     127  EM37:     .ASCIZ  /PSW CHANGED BY AN RTI IN USER MODE/
1694 011362    101     102     117  EM40:     .ASCIZ  /ABORT I KERNAL D-SPACE PICKED UP VECTOR FROM I-SPACE/
1695 011447    104     040     123  EM41:     .ASCIZ  /D SPACE ENABLE CIRCUITRY HAS FAILED/
1696 011513    111     116     103  EM42:     .ASCIZ  /INCORRECT STORE BY MTP INSTRUCTION/
1697 011556    124     122     111  EM43:     .ASCIZ  /TRIED TO REFERENCE NON-RESIDENT PAGE/
1698 011623    127     122     117  EM44:     .ASCIZ  /WRONG DATA FETCHED BY MFP INSTRUCTION/
1699 011671    111     114     114  EM45:     .ASCIZ  /ILLEGAL CSM DID NOT TRAP TO 10/
1700 011730    103     123     115  EM46:     .ASCIZ  /CSM DID NOT ENTER SUPERVISOR MODE/
1701 011772    103     123     115  EM47:     .ASCIZ  /CSM SET UP WRONG PREVIOUS MODE/
1702 012031    103     123     115  EM50:     .ASCIZ  /CSM SET UP STACK WRONG/
1703 012060    103     123     115  EM51:     .ASCIZ  /CSM PUSHED INCORRECT ARGUMENT/
1704 012116    103     123     115  EM52:     .ASCIZ  /CSM PUSHED WRONG PC/
1705 012142    103     123     115  EM53:     .ASCIZ  /CSM DID NOT CLEAR OLD PSW BITS <3:0>/
1706 012207    103     123     115  EM54:     .ASCIZ  /CSM ACCESSED WRONG SUPERVISOR SPACE/
1707 012253    103     123     115  EM55:     .ASCIZ  /CSM ABORTED WHEN IT SHOULD NOT HAVE/
1708 012317    103     123     115  EM56:     .ASCIZ  ?CSM FAILED TO INCREMENT/DECREMENT REGISTER PROPERLY?
1709 012403    103     123     115  EM57:     .ASCIZ  /CSM FAILED TO PUT PROPER ARGUMENT ON STACK/
1710                                            .EVEN
```

DATA HEADERS

```
1712                                                    .SBTTL  DATA HEADERS
1713 012456  117  114  104  DH1:    .ASCIZ  /OLD PC  OLD PSW R6 WAS  CPUERR  TESTNO  ERRORPC/
1714 012536  117  114  104  DH2:    .ASCIZ  /OLD PC  OLD PSW R6 WAS  SR0     SR2     TESTNO  ERRORPC/
1715 012626  120  104  122  DH10:   .ASCIZ  /PDR 4   PSW      TESTNO  ERRORPC/
1716 012666  123  122  060  DH12:   .ASCIZ  /SR0 WAS EXPECTD PDR 4   PSW     TESTNO  ERRORPC/
1717 012746  123  122  062  DH13:   .ASCIZ  /SR2 WAS EXPECTD PDR 4   PSW     TESTNO  ERRORPC/
1718 013026  126  056  102  DH14:   .ASCIZ  /V.B.A.  KIPDR4  SR0 WAS SR2 WAS TESTNO  ERRORPC/
1719 013106  126  056  102  DH15:   .ASCIZ  /V.B.A.  KIPDR4  TESTNO  ERRORPC/
1720 013146  126  056  102  DH16:   .ASCIZ  /V.B.A.  KIPDR4  SR0 WAS EXPECTD TESTNO  ERRORPC/
1721 013226  126  056  102  DH17:   .ASCIZ  /V.B.A.  KIPDR4  SR2 WAS EXPECTD TESTNO  ERRORPC/
1722 013306  123  122  062  DH20:   .ASCIZ  /SR2 WAS EXPECTD TESTNO  ERRORPC/
1723 013346  106  111  122  DH21:   .ASCII  /FIRST ABORT     SECOND ABORT/<CRLF>
1724 013403  123  122  060          .ASCIZ  /SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO  ERRORPC/
1725 013463  123  122  060  DH22:   .ASCIZ  /SR0 WAS SR2 WAS TESTNO  ERRORPC/
1726 013523  120  123  127  DH24:   .ASCIZ  /PSW WAS R6 WAS  TESTNO  ERRORPC/
1727 013563  105  130  120  DH26:   .ASCII  /EXPECTED          RECEIVED/<CRLF>
1728 013615  123  122  060          .ASCIZ  /SR0     SR2     SR0 WAS SR2 WAS TESTNO  ERRORPC/
1729 013675  105  130  120  DH27:   .ASCII  /EXPECTED:/<CRLF>
1730 013707  120  123  127          .ASCII  /PSW     PC      SR0     SR2/<CRLF>
1731 013743  061  067  060          .ASCII  /170017  (3$+4)  020147  (3$)/<CRLF>
1732 014000  122  105  103          .ASCII  /RECEIVED:/<CRLF>
1733 014012  120  123  127          .ASCIZ  /PSW     PC      SR0     SR2     TESTNO  ERRORPC/
1734 014072  104  101  124  DH30:   .ASCII  /DATA    DATA/<CRLF>
1735 014107  105  130  120          .ASCIZ  /EXPECTD RECEIVD TESTNO  ERR PC/
1736 014146  124  105  123  DH32:   .ASCIZ  /TESTNO  ERR PC/
1737 014165  123  122  060  DH33:   .ASCIZ  /SR0 WAS SR2 WAS TESTNO  ERRORPC/
1738 014225  050  115  115  DH34:   .ASCIZ  /(MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC FXPECTING 020031/
1739 014316  123  122  060  DH36:   .ASCIZ  /SR0 WAS EXPECTD TESTNO  ERRORPC/
1740 014356  120  123  127  DH37:   .ASCIZ  /PSW WAS EXPECTD TESTNO  ERRORPC/
1741 014416  050  120  123  DH40:   .ASCIZ  /(PSW)   TESTNO  ERRORPC EXPECTING XXX340/
1742 014467  105  122  122  DH41:   .ASCII  /ERROR   AUTOI-D VIRTUAL/<CRLF>
1743 014517  122  105  107          .ASCIZ  /REGISTR REGISTR ADDRESS TESTNO  PC AT ABORT/
1744 014573  107  104  104  DH42:   .ASCIZ  /GDDATA  STORED  TESTNO  ERRORPC/
1745 014633  050  115  115  DH43:   .ASCIZ  /(MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC/
1746 014703  105  130  120  DH44:   .ASCIZ  /EXPECTD (PSW)   TESTNO  ERRORPC/
1747 014743  117  114  104  DH45:   .ASCIZ  /OLDPSW  TESTNO  ERRORPC/
1748 014773  124  105  123  DH55:   .ASCIZ  /TESTNO  ERR PC  R0 EXP  R0 RCV/
1749 015032  124  105  123  DH57:   .ASCIZ  /TESTNO  ERR PC  ARGEXP  ARGRCV/
1750                                         .EVEN
```

```
1752                                                    .SBTTL  DATA TABLES
1753 015072  001260  001262  001256  DT1:      .WORD   TRAPPC,TRAPPS,WASR6,CPUERR,TESTNO,$ERRPC,0
1754 015110  001260  001262  001256  DT2:      .WORD   TRAPPC,TRAPPS,WASR6,WASSR0,WASSR2,TESTNO,$ERRPC,0
1755 015130  001166  001176  001254  DT10:     .WORD   $REG2,$TMP0,TESTNO,$ERRPC,0
1756 015142  001264  001170  001166  DT12:     .WORD   WASSR0,$REG3,$REG2,$TMP0,TESTNO,$ERRPC,0
1757 015160  001270  001172  001166  DT13:     .WORD   WASSR2,$REG4,$REG2,$TMP0,TESTNO,$ERRPC,0
1758 015176  001162  001172  001264  DT14:     .WORD   $REG0,$REG4,WASSR0,WASSR2,TESTNO,$ERRPC,0
1759 015214  001162  001172  001254  DT15:     .WORD   $REG0,$REG4,TESTNO,$ERRPC,0
1760 015226  001162  001172  001264  DT16:     .WORD   $REG0,$REG4,WASSR0,$REG2,TESTNO,$ERRPC.0
1761 015244  001162  001172  001270  DT17:     .WORD   $REG0,$REG4,WASSR2,$REG3,TESTNO,$ERRPC,0
1762 015262  001270  001164  001254  DT20:     .WORD   WASSR2,$REG1,TESTNO,$ERRPC,0
1763 015274  001176  001202  001264  DT21:     .WORD   $TMP0,$TMP2,WASSR0,WASSR2,TESTNO,$ERRPC,0
1764 015312  001264  001270  001254  DT22:     .WORD   WASSR0,WASSR2,TESTNO,$ERRPC,0
1765 015324  001164  001166  001254  DT24:     .WORD   $REG1,$REG2,TESTNO,$ERRPC,0
1766 015336  001162  001164  001264  DT26:     .WORD   $REG0,$REG1,WASSR0,WASSR2,TESTNO,$ERRPC,0
1767 015354  001164  001170  001264  DT27:     .WORD   $REG1,$REG3,WASSR0,WASSR2,TESTNO,$ERRPC,0
1768 015372  035230  001176  001254  DT30:     .WORD   ACSMSP,$TMP0,TESTNO,$ERRPC,0
1769 015404  001254  001116  000000  DT32:     .WORD   TESTNO,$ERRPC,0
1770 015412  001164  001166  001170  DT34:     .WORD   $REG1,$REG2,$REG3,TESTNO,$ERRPC,0
1771 015426  001264  001164  001254  DT36:     .WORD   WASSR0,$REG1,TESTNO,$ERRPC,0
1772 015440  001162  001254  001116  DT40:     .WORD   $REG0,TESTNO,$ERRPC,0
1773 015450  001264  001266  001270  DT41:     .WORD   WASSR0,WASSR1,WASSR2,TESTNO,BADPC,0
1774 015464  001162  001164  001254  DT42:     .WORD   $REG0,$REG1,TESTNO,$ERRPC,0
1775 015476  001162  001164  001166  DT43:     .WORD   $REG0,$REG1,$REG2,TESTNO,$ERRPC,0
1776 015512  001264  001266  001270  DT45:     .WORD   WASSR0,WASSR1,WASSR2,TESTNO,$ERRPC,0
1777 015526  001170  035226  001254  DT46:     .WORD   $REG3,ACSMPS,TESTNO,$ERRPC,0
1778 015540  001162  035220  001254  DT47:     .WORD   $REG0,CSM1ST,TESTNO,$ERRPC,0
1779 015552  001176  035222  001254  DT50:     .WORD   $TMP0,CSM2ND,TESTNO,$ERRPC,0
1780 015564  035224  001254  001116  DT52:     .WORD   CSM3RD,TESTNO,$ERRPC,0
1781 015574  001254  001116  001176  DT55:     .WORD   TESTNO,$ERRPC,$TMP0,$REG0,0
1782 015606  001254  001116  001176  DT57:     .WORD   TESTNO,$ERRPC,$TMP0,$TMP1,0
```

```
                                                .SBTTL  DATA FORMAT BYTE STRINGS
1784
1785 015620     000     000     000    DF1:     .BYTE   0,0,0,0,0
1786 015625     000     000     000    DF2:     .BYTE   0,0,0,0,0,0,0
1787 015634     000     000     000    DF3:     .BYTE   0,0,0,0
1788 015640     000     000     000    DF5:     .BYTE   0,0,0
1789 015643     000     000     000    DF12:    .BYTE   0,0,0,0,0,0
1790 015651     000     000             DF32:    .BYTE   0,0
1791                                             .EVEN
```

```
*****   TRAP HANDLING ROUTINES  *****

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

                                        .SBTTL  CPU TRAP HANDLER ROUTINE
                                        ;;*****************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS THRU
                                        ;*      "ERRVEC" (LOC. 004).  IF THIS SUBROUTINE IS ENTERED BY A
                                        ;*      SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A HALT IS
                                        ;*      EXECUTED.
                                        ;*
                                        ;;*****************************************************************
015654  005227                  TIMERR: INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME THRU
015656  177777                  TIMFLG: .WORD   -1              ;NEGATIVE ONE FOR "HAVE ENTERED" FLAG
015660  001401                          BEQ     1$              ;BRANCH IF FIRST TIME IN
015662  000000                          HALT                    ;STOP! - I'VE ENTERED THIS ROUTINE
                                                                ;A SECOND TIME BEFORE I FINISHED
                                                                ;REPORTING THE FIRST ERROR.  THE
                                                                ;SECOND ENTRY ADDRESS SHOULD BE ON
                                                                ;THE KERNEL STACK.
015664  012637  001260          1$:     MOV     (KSP)+,TRAPPC   ;SAVE PC+2 AT TIME OF ABORT
015670  012637  001262                  MOV     (KSP)+,TRAPS    ;SAVE PS AT TIME OF ABORT
015674  010637  001256                  MOV     KSP,WASR6       ;SAVE STACK POINTER VALUE
015700  104001                          ERROR   +1              ;UNEXPECTED TRAP OR ABORT TO LOC. 4
015702  012737  177777  015656          MOV     #-1,TIMFLG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
015710  005037  177766                  CLR     CPUERR          ;CLEAR THE CPU ERROR REGISTER
015714  013746  001262                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
015720  013746  001260                  MOV     TRAPPC,-(KSP)
015724  000006                          RTT                     ;RETURN FROM INTERRUPT OR ABORT
```

```
                                        .SBTTL  MEMORY MANAGEMENT TRAP HANDLER ROUTINE
                                        ;;*****************************************************************
                                        ;*
                                        ;*      THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED MEMORY MANAGEMENT
                                        ;*      TRAPS AND ABORTS THRU 'MMVEC'' (LOC. 250).  IF THIS SUBROUTINE IS
                                        ;*      ENTERED BY A SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A
                                        ;*      HALT IS EXECUTED.
                                        ;*
                                        ;*****************************************************************
015726  005227                  MGMERR: INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME THRU
015730  177777                  MGMFLG: .WORD   -1              ;NEGATIVE ONE FOR 'HAVE ENTERED'' FLAG
015732  001401                          BEQ     1$              ;BRANCH IF FIRST TIME IN
015734  000000                          HALT                    ;STOP! - I'VE ENTERED THIS ROUTINE
                                                                ;A SECOND TIME BEFORE   FINISHED
                                                                ;REPORTING THE FIRST ERROR.  THE
                                                                ;SECOND ENTRY ADDRESS SHOULD BE ON
                                                                ;THE KERNEL STACK.
015736  012637  001260          1$:     MOV     (KSP)+,TRAPPC   ;SAVE PC+2 AT TIME OF ABORT
015742  012637  001262                  MOV     (KSP)+,TRAPPS   ;SAVE PS AT TIME OF ABORT
015746  010637  001256                  MOV     KSP,WASR6       ;SAVE STACK POINTER VALUE
015752  013737  177572  001264        MOV     SR0,WASSR0      ;SAVE CONTENTS OF KT STATUS REG. 0
015760  013737  177576  001270        MOV     SR2,WASSR2      ;SAVE CONTENTS OF KT STATUS REG. 2
015766  042737  160000  177572        BIC     #160000,SR0     ;CLEAR ERROR BITS IN STATUS REG 0
015774  104002                          ERROR   +2              ;UNEXPECTED TRAP OR ABORT TO LOC. 250
015776  012737  177777  015730        MOV     #-1,MGMFLG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
016004  013746  001262                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
016010  013746  001260                  MOV     TRAPPC,-(KSP)
016014  000006                          RTT                     ;RETURN FROM INTERRUPT OR ABORT.

                                        ;THE NEXT LOCATIONS ARE RESERVED FOR ANY PERMANENT PATCHES:
016016  000000  000000  000000  PATCH:   .WORD   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
016112  000000  000000  000000          .WORD   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

MEMORY MANAGEMENT TRAP HANDLER ROUTINE

```
1852                                        ;*      *****   STARTING POINT OF TEST  *****
1853                                        ;*      *****   STARTING ADDRESS OF 200 *****
1854          020000                        .=20000
1855
1856 020000                           START:
                                      .SBTTL  INITIALIZE THE COMMON TAGS
                                      ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
      020000  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
      020004  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
      020006  022706  001140                  CMP     #SWR,R6 ;;DONE?
      020012  001374                          BNE     .-6             ;;LOOP BACK IF NO
      020014  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                      ;;INITIALIZE A FEW VECTORS
      020020  012737  003032  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
      020026  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
      020034  012737  003260  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
      020042  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
      020050  012737  007176  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
      020056  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
      020064  012737  007264  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
      020072  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
      020100  013737  035746  035740          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
      020106  005037  001212                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
      020112  112737  000001  001115          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
                                      ;;INITIALIZE THE 'T-BIT'' TRAP VECTOR. THEN LOAD LOCATION ''$RTRN'', IN
                                      ;;THE ''END-OF-PASS'' ($EOP) ROUTINE, WITH A ''RTI'' OR ''RTT''.
      020120  012737  036164  000014          MOV     #$RTRN,@#TBITVEC ;;SET ''T'' BIT VECTOR TO $RTRN
      020126  012737  000340  000016          MOV     #340,@#TBITVEC+2 ;;LEVEL 7
      020134  012737  000002  036164          MOV     #RTI,$RTRN      ;;SET $RTRN TO A RTI
      020142  012737  020170  000010          MOV     #65$,@#RESVEC   ;;TRY TO DO A RTT
      020150  005046                          CLR     -(SP)           ;;DUMMY PS
      020152  012746  020160                  MOV     #64$,-(SP)      ;;AND PC
      020156  000006                          RTT                     ;;TRY THE RTT
      020160  012737  000006  036164  64$:    MOV     #RTT,$RTRN      ;;RTT IS LEGAL--SET $RTRN TO A RTT
      020166  000402                          BR      66$
      020170  062706  000010          65$:    ADD     #10,SP          ;;RTT ILLEGAL--CLEAN OFF THE STACK
      020174  012737  000012  000010  66$:    MOV     #RESVEC+2,@#RESVEC ;;RESTORE TRAP CATCHER
      020202  005037  001310                  CLR     $TBIT           ;;CLEAR ''T'' BIT SWITCH
      020206  012737  020206  001106          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
      020214  012737  020214  001110          MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                      ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                      ;;EQUAL TO A ''-1'', SETUP FOR A SOFTWARE SWITCH REGISTER.
      020222  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
      020226  012737  020262  000004          MOV     #67$,@#ERRVEC   ;;SET UP ERROR VECTOR
      020234  012737  177570  001140          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
      020242  012737  177570  001142          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
      020250  022777  177777  160662          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
      020256  001012                          BNE     69$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                      ;;AND  THE HARDWARE SWR IS NOT = -1
      020260  000403                          BR      68$             ;;BRANCH IF NO TIMEOUT
      020262  012716  020270          67$:    MOV     #68$,(SP)       ;;SET UP FOR TRAP RETURN
      020266  000002                          RTI
      020270  012737  000176  001140  68$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
      020276  012737  000174  001142          MOV     #DISPREG,DISPLAY
      020304  012637  000004          69$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
      020310  005037  001232                  CLR     $PASS           ;;CLEAR PASS COUNT
      020314  132737  000200  001245          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
```

INITIALIZE THE COMMON TAGS

```
020322  001403                          BEQ     70$             ;;YES,USE NON-APT SWITCH
020324  012737  001246  001140          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
020332                          70$:
1857                            .SBTTL  TYPE PROGRAM NAME
                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
020332  005227  177777                  INC     #-1             ;;FIRST TIME?
020336  001047                          BNE     71$             ;;BRANCH IF NO
020340  022737  036120  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
020346  001443                          BEQ     71$             ;;BRANCH IF YES
020350  104401  020416                  TYPE    ,72$            ;;TYPE ASCIZ STRING
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
020354  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
020360  001012                          BNE     73$             ;;BRANCH IF YES
020362  123727  001244  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
020370  001406                          BEQ     73$             ;;BRANCH IF YES
020372  023727  001140  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
020400  001005                          BNE     74$             ;;BRANCH IF NO
020402  104407                          GTSWR                   ;;GET SOFT-SWR SETTINGS
020404  000403                          BR      74$
020406  112737  000001  001134  73$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
020414                          74$:
020414  000420                          BR      71$             ;;GET OVER THE ASCIZ
                                ;;72$:  .ASCIZ  <CRLF>#CKKTBC0 11/44 MEM MGMT PRT B#<CRLF>
020456                          71$:
1858
1859 020456                     LOOP:
1860 020456  012706  001100            MOV     #STACK,KSP      ;INITIALIZE THE STACK POINTER
1861 020462  012737  040000  177776      MOV     #40000,PSW      ;TURN ON SUPERVISOR MODE
1862 020470  012706  000700            MOV     #SUPSTK,SSP     ;INITIALIZE SUPER. STACK POINTER
1863 020474  012737  140000  177776      MOV     #140000,PSW     ;TURN ON USER MODE
1864 020502  012706  000600            MOV     #USESTK,USP     ;INITIALIZE USER STACK POINTER
1865 020506  005037  177776            CLR     PSW             ;RETURN TO KERNAL MODE
1866 020512  012737  015654  000004      MOV     #TIMERR,ERRVEC  ;LOAD CPU SERVICE ROUTINE INTO TRAP VECTOR
1867 020520  012737  000340  000006      MOV     #340,ERRVEC+2   ;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
1868 020526  012737  015726  000250      MOV     #MGMERR,MMVEC   ;LOAD MEMORY MANAGENT ROUTINE INTO VECTOR
1869 020534  012737  000340  000252      MOV     #340,MMVEC+2    ;SET NEW PS TO FRIORITY LEVEL 7-KERNEL
1870 020542  012700  177777            MOV     #-1,R0          ;PUT -1 INTO R0 TO INITIALIZE FLAGS
1871 020546  010037  015656            MOV     R0,TIMFLG       ;INITIALIZE CPU ERROR FLAG
1872 020552  010037  015730            MOV     R0,MGMFLG       ;INITIALIZE MEMORY MANAGEMENT ERROR FLAG
1873 020556  012737  000340  001274      MOV     #340,TBITPS     ;INITIALIZE LOG THAT HOLDS T-BIT PSW
1874 020564  005037  177572      INIT:   CLR     MMR0            ;BE SURE MEM. MGMT IS OFF TO START WITH,
1875 020570  012737  000020  172516      MOV     #BIT4,MMR3      ;BUT TURN ON 22-BIT ADDRESSING MODE
1876 020576  004737  002050            JSR     PC,APRINIT      ;JUMP TO PAR/PDR INIT ROUTINE
```

```
1878                            .SBTTL  GROUP 1   ACCESS CONTROL FIELD/SPECIAL ABORTS TESTS
1879                            ;/.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\
1880                            ;;*****************************************************************
1881                            ;*
1882                            ;*      THE NEXT THREE (3) TESTS CAUSE MEMORY MANAGEMENT ERRORS
1883                            ;*      TO CHECK THE ABILITY OF STATUS REGISTER 0 TO RECORD KT
1884                            ;*      ERRORS AND THE ABILITY OF STATUS REGISTER 2 TO LOCK UP THE
1885                            ;*      VIRTUAL ADDR. OF THE INSTRUCTION THAT CAUSED THE ERROR.
1886                            ;*      THE BITS OF SR2 ARE CHECKED AND BITS <15:13>, <6:5>, AND <3:0>
1887                            ;*      ARE CHECKED IN SR0.  SO THE SR0 AND SR2 LOGIC AND THE
1888                            ;*      KT ERROR LOGIC ARE CHECKED.
1889                            ;*
1890                            ;;*****************************************************************
1891                            ;/.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\
```

```
1900                                    .SBTTL  TEST # 1 - NON-RESIDENT ABORT TEST (ACF=0&4)
                                        ;*****************************************************************
                                        ;*TEST 1        NON-RESIDENT ABORT TEST (ACF=0&4)
                                        ;*
                                        ;*      THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
                                        ;*      LOGIC BY CAUSING NON-RESIDENT ABORTS IN KERNEL, SUPERVISOR
                                        ;*      AND USER MODES.  PDR 4 IS LOADED WITH ACF'S = 0&4 AND
                                        ;*      THEN PHYSICAL ADDR. 60000 IS ACCESSED TO CAUSE THE ABORT.
                                        ;*
                                        ;*****************************************************************
020602  000004                          TST1:   SCOPE
1901 020604  012700  000600            1$:     MOV     #600,R0         ;LOAD DATA FOR PAR'S INTO R0
1902 020610  010037  172346                    MOV     R0,KIPAR3       ;MAP KERNEL PAR'S 3&4 TO 12-16K
1903 020614  010037  172350                    MOV     R0,KIPAR4
1904 020620  010037  172246                    MOV     R0,SIPAR3       ;MAP SUPERVISOR PAR'S 3&4 TO 12-16K
1905 020624  010037  172250                    MOV     R0,SIPAR4
1906 020630  010037  177646                    MOV     R0,UIPAR3       ;MAP USER PAR'S 3&4 TO 12-16K
1907 020634  010037  177650                    MOV     R0,UIPAR4
1908 020640  012700  060000                    MOV     #60000,R0       ;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
1909 020644  012701  100000                    MOV     #100000,R1      ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
1910 020650  012703  100011                    MOV     #100011,R3      ;LOAD R3 WITH WHAT SR0 SHOULD READ - N.R., KERNEL, PG.4
1911 020654  012702  077400                    MOV     #77400,R2       ;LOAD ACF=0 (NON-RESIDENT) PDR VALUE IN R2
1912 020660  012737  020734  000250    2$:     MOV     #5$,@#MMVEC     ;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
1913 020666  010237  172310                    MOV     R2,KIPDR4       ;LOAD ACF TEST VALUE INTO KIPDR4
1914 020672  010237  172210                    MOV     R2,SIPDR4       ;LOAD ACF TEST VALUE INTO SIPDR4
1915 020676  010237  177610                    MOV     R2,UIPDR4       ;LOAD ACF TEST VALUE INTO UIPDR4
1916 020702  012737  020716  001110          MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
1917 020710  012737  000001  177572          MOV     #1,@#MMR0       ;TURN ON MEMORY MANAGEMENT
1918 020716  005010                    3$:     CLR     (R0)            ;CLEAR PHYS. LOC. 60000 USING PDR3
1919 020720  013737  177776  001176          MOV     PSW,$TMP0       ;SAVE PSW IN CASE OF ERROR
1920 020726  005211                    4$:     INC     (R1)            ;TRY TO REF. IT USING PDR4 - SHOULD TRAP TO 5$
1921 020730  104003                            ERROR   +3              ;MEM. MGMT. ABORT DID NOT OCCUR
1922                                                                    ;FOR TIGHTER SCOPE LOOP
1923                                                                    ;REPLACE ERROR CALL WITH
1924                                                                    ;'BR 3$'' = 000772
1925 020732  000425                            BR      8$              ;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
1926 020734  062706  000004            5$:     ADD     #4,SP           ;RESTORE STACK POINTER
1927 020740  005710                            TST     (R0)            ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
1928 020742  001401                            BEQ     6$              ;BRANCH IF YES
1929 020744  104004                            ERROR   +4              ;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
1930                                                                    ;FOR TIGHTER SCOPE LOOP
1931                                                                    ;REPLACE ERROR CALL WITH
1932                                                                    ;'BR 3$'' = 000764
1933 020746  013737  177572  001264    6$:     MOV     SR0,WASSR0      ;READ STATUS REGISTER 0
1934 020754  013737  177576  001270            MOV     SR2,WASSR2      ;READ STATUS REGISTER 2
1935 020762  020337  001264                    CMP     R3,WASSR0       ;DID SR0 REPORT NON-RESIDENT ERROR CORRECTLY?
1936 020766  001401                            BEQ     7$              ;BRANCH IF YES
1937 020770  104005                            ERROR   +5              ;SR0 DID NOT REPORT NON-RES. ERROR CORRECTLY
1938                                                                    ;FOR TIGHTER SCOPE LOOP
1939                                                                    ;REPLACE ERROR CALL WITH
1940                                                                    ;'BR 3$'' = 000752
1941 020772  012704  020726            7$:     MOV     #4$,R4          ;LOAD R4 WITH WHAT SR2 SHOULD READ
1942 020776  020437  001270                    CMP     R4,WASSR2       ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4$)?
1943 021002  001401                            BEQ     8$              ;BRANCH IF YES
1944 021004  104006                            ERROR   +6              ;SR2 DID NOT LOCK VIRTUAL ADDR. OF NON-RES. ERROR
1945                                                                    ;FOR  IGHTER SCOPE LOOP
1946                                                                    ;REPLACE ERROR CALL WITH
```

```
1947
1948 021006  005037  177572          8$:     CLR     MMR0            ;'BR 3$'' = 000744
1949 021012  032737  140000  001176          BIT     #140000,$TMP0   ;TURN OFF MEMORY MANAGEMENT
1950 021020  001006                          BNE     11$             ;HAS ACF=0&4 BEEN TESTED IN USER YET
1951 021022  012703  100151                  MOV     #100151,R3      ;BRANCH IF YES
1952 021026  012737  140000  177776          MOV     #140000,PSW     ;LOAD R3 WITH WHAT SR0 SHOULD READ - N.R., USER, PG.4
1953 021034  000711                          BR      2$              ;GO TO USER MODE
1954 021036  022737  040000  001176  11$:     CMP     #40000,$TMP0    ;REPEAT TEST IN USER MODE
1955 021044  001006                          BNE     9$              ;HAS ACF=0&4 BEEN TESTED IN SUPERVISOR YET
1956 021046  012703  100051                  MOV     #100051,R3      ;BRANCH IF YES
1957 021052  012737  040000  177776          MOV     #40000,PSW      ;LOAD R3 WITH WHAT SR0 SHOULD READ - N.R., SUPERVISOR, PG.4
1958 021060  000677                          BR      2$              ;GO TO SUPERVISOR MODE
1959 021062  022702  077404          9$:     CMP     #77404,R2       ;REPEAT TEST IN SUPERVISOR MODE
1960 021066  001407                          BEQ     10$             ;HAS ACF=4 BEEN TESTED YET?
1961 021070  012702  077404                  MOV     #77404,R2       ;BRANCH IF YES
1962 021074  012703  100011                  MOV     #100011,R3      ;THEN LOAD ACF=4 (NON-RES) PDR VALUE IN R2
1963 021100  005037  177776                  CLR     PSW             ;LOAD R3 WITH WHAT SR0 SHOULD READ-N.R.,KERNEL,PG. 4
1964 021104  000665                          BR      2$              ;GO BACK TO KERNEL MODE
1965 021106  005037  177776          10$:    CLR     PSW             ;GO BACK & TEST ACF=4 IN SAME MODE
1966 021112  012737  015726  000250          MOV     #MGMERR,MMVEC   ;GO BACK TO KERNEL MODE BEFORE LEAVING
1967                                                                 ;RESTORE ADDRESS OF NORMAL MEMORY
                                                                     ;MANAGEMENT ERROR ROUTINE TO MMVEC
```

```
                        .SBTTL  TEST # 2 - READ-ONLY ABORT TEST (ACF 2)
                        ;*****************************************************************
                        ;*TEST 2        READ-ONLY ABORT TEST (ACF-2)
                        ;*
                        ;*      THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
                        ;*      LOGIC BY CAUSING READ-ONLY ABORTS IN KERNEL, SUPERVISOR AND
                        ;*      USER MODES.  PDR 4 IS LOAD WITH ACF=2 AND THEN
                        ;*      PHYSICAL ADDR. 60000 IS WRITTEN TO CAUSE THE ABORT.
                        ;*
                        ;*****************************************************************
 021120  000004         TST2:   SCOPE
1977 021122             1$:                             ;KERNEL, SUPERVISOR AND USER PAR'S 3 & 4,
1978                                                    ;AND PDR 3 ARE SETUP FROM LAST TEST
1979 021122  012700  060000            MOV     #60000,R0       ;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
1980 021126  012701  100000            MOV     #100000,R1      ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
1981 021132  012703  020011            MOV     #20011,R3       ;LOAD R3 WITH WHAT SR0 SHOULD READ - R/O, KERNEL, PG.4
1982 021136  012702  077402            MOV     #77402,R2       ;LOAD ACF=2 (READ-ONLY) PDR VALUE IN R2
1983 021142  012737  021214  000250 2$:  MOV     #5$,MMVEC       ;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
1984 021150  010237  172310            MOV     R2,KIPDR4       ;LOAD ACF=2 INTO KIPDR4
1985 021154  010237  172210            MOV     R2,SIPDR4       ;LOAD ACF=2 INTO SIPDR4
1986 021160  010237  177610            MOV     R2,UIPDR4       ;LOAD ACF=2 INTO UIPDR4
1987 021164  012737  021176  001110    MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
1988 021172  005237  177572            INC     MMR0            ;TURN ON MEMORY MANAGEMENT
1989 021176  005010            3$:     CLR     (R0)            ;CLEAR PHYS. LOC. 60000 USING PDR3
1990 021200  013737  177776  001176    MOV     PSW,$TMP0       ;SAVE PSW IN CASE OF ERROR
1991 021206  005211            4$:     INC     (R1)            ;TRY TO WRITE USING PDR4 - SHOULD TRAP TO 5$
1992 021210  104003                    ERROR   +3              ;MEM. MGMT. ABORT DID NOT OCCUR
1993                                                    ;FOR TIGHTER SCOPE LOOP
1994                                                    ;REPLACE ERROR CALL WITH
1995                                                    ;'BR 3$'' = 000772
1996 021212  000425                    BR      8$              ;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
1997 021214  062706  000004    5$:     ADD     #4,SP           ;RESTORE STACK POINTER
1998 021220  005710                    TST     (R0)            ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
1999 021222  001401                    BEQ     6$              ;BRANCH IF YES
2000 021224  104004                    ERROR   +4              ;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
2001                                                    ;FOR TIGHTER SCOPE LOOP
2002                                                    ;REPLACE ERROR CALL WITH
2003                                                    ;'BR 3$'' = 000764
2004 021226  013737  177572  001264 6$:     MOV     SR0,WASSR0      ;READ STATUS REG. 0
2005 021234  013737  177576  001270            MOV     SR2,WASSR2      ;READ STATUS REG. 2
2006 021242  020337  001264            CMP     R3,WASSR0       ;DID SR0 REPORT READ-ONLY ERROR CORRECTLY?
2007 021246  001401                    BEQ     7$              ;BRANCH IF YES
2008 021250  104005                    ERROR   +5              ;SR0 DID NOT REPORT R/O ERROR CORRECTLY
2009                                                    ;FOR TIGHTER SCOPE LOOP
2010                                                    ;REPLACE ERROR CALL WITH
2011                                                    ;'BR 3$'' = 000752
2012 021252  012704  021206    7$:     MOV     #4$,R4          ;LOAD R4 WITH WHAT SR2 SHOULD READ
2013 021256  020437  001270            CMP     R4,WASSR2       ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (-4$)?
2014 021262  001401                    BEQ     8$              ;BRANCH IF YES
2015 021264  104006                    ERROR   +6              ;SR2 DID NOT LOCKUP VIRTUAL ADDR. OF R/O ERROR
2016                                                    ;FOR TIGHTER SCOPE LOOP
2017                                                    ;REPLACE ERROR CALL WITH
2018                                                    ;'BR 3$'' = 000744
2019 021266  005037  177572    8$:     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2020 021272  032737  140000  001176    BIT     #140000,$TMP0   ;HAS ACF-2 BEEN TESTED IN USER MODE?
2021 021300  001006                    BNE     11$             ;BRANCH IF YES
2022 021302  012703  020151            MOV     #20151,R3       ;LOAD R3 WITH WHAT SR0 SHOULD READ-R/O, USER, PG.4
```

```
2023 021306  012737  140000  177776          MOV     #140000,PSW     ;GO TO USER MODE
2024 021314  000712                          BR      2$              ;REPEAT TEST IN USER MODE
2025 021316  022737  040000  001176  11$:    CMP     #40000,$TMP0    ;HAS ACF=2 BEEN TESTED IN SUPERVISOR MODE?
2026 021324  001006                          BNE     9$              ;BRANCH IF YES
2027 021326  012703  020051                  MOV     #20051,R3       ;LOAD R3 WITH WHAT SR0 SHOULD READ-R/O, SUPERVISOR, PG.4
2028 021332  012737  040000  177776          MOV     #40000,PSW      ;GO TO SUPERVISOR MODE
2029 021340  000700                          BR      2$              ;REPEAT TEST IN SUPERVISOR MODE
2030 021342  005037  177776          9$:     CLR     PSW             ;GO BACK TO KERNEL MODE BEFORE LEAVING
2031 021346  012737  015726  000250          MOV     #MGMERR,MMVEC   ;RESTORE ADDRESS OF NORMAL MEMORY
2032                                                                 ;MANAGEMENT ERROR ROUTINE TO MMVEC.
```

```
2043                                    .SBTTL  TEST # 3 - TEST ILLEGAL MODE "10"
                                        ;*****************************************************************
                                        ;*TEST 3        TEST ILLEGAL MODE "10"
                                        ;*
                                        ;*      THIS TEST CHECKS TO SEE THAT A 10 IN THE CURRENT MODE BITS OF THE
                                        ;*      PSW WHILE MEMORY MANAGEMENT IS ON IS ILLEGAL.  A
                                        ;*      MEMORY MANAGEMENT ABORT SHOULD OCCUR AND STATUS REGISTER 0
                                        ;*      SHOULD REPORT NON-RESIDENT ABORT, MODE = 10, PAGE = 1 (100103).  STATUS
                                        ;*      REGISTER 2 SHOULD LOCKUP THE ADDRESS OF THE INSTRUCTION
                                        ;*      THAT LOADED THE PSW.
                                        ;*
                                        ;*****************************************************************
     021354  000004                     TST3:   SCOPE
2044 021356  012737  021400  001110     1$:     MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
2045 021364  012737  021420  000250             MOV     #3$,MMVEC       ;LOAD MEM. MGMT. TRAP VECTOR WITH 3$
2046 021372  012737  000001  177572             MOV     #1,MMR0         ;TURN ON MEMORY MANAGEMENT
2047 021400  012737  100000  177776     10$:    MOV     #100000,PSW     ;SET 10 IN PSW CURRENT MODE BITS
2048 021406  104030                     2$:     ERROR   +30             ;ILLEGAL MODE 10 NOT ABORTED
2049                                             ;FOR A TIGHTER SCOPE LOOP, REPLACE ERROR CALL WITH 'BR '0$''   000774
2050 021410  012737  015726  000250             MOV     #MGMERR,MMVEC   ;RESTORE MEM. MGMT. ABORT VECTOR
2051 021416  000437                             BR      5$              ;BRANCH AROUND SR0 & SR2 CHECKS
2052 021420  013737  177576  001270     3$:     MOV     SR2,WASSR2      ;READ CONTENTS OF SR2
2053 021426  012737  015726  000250             MOV     #MGMERR,MMVEC   ;RESTORE MEM. MGMT. ABORT VECTOR
2054 021434  012706  001100                     MOV     #KERSTK,SP      ;RESTORE STACK POINTER
2055 021440  022737  100103  177572             CMP     #100103,SR0     ;DID SR0 REPORT ILLEGAL MODE CORRECTLY?
2056 021446  001406                             BEQ     4$              ;BRANCH IF YES
2057 021450  012701  100103                     MOV     #100103,R1      ;LOAD EXPECTED CONTENTS OR SR0 INTO R1
2058 021454  013737  177572  001264             MOV     SR0,WASSR0      ;READ CONTENTS OF SR0
2059 021462  104031                             ERROR   +31             ;SR0 DID NOT REPORT NR ABORT, PG=1, MODE=10
2060                                             ;FOR TIGHTER SCOPE LOOP, REPLACE ERROR CALL WITH 'BR 10$'' = 000746
2061 021464  032737  100000  172516     4$:     BIT     #BIT15,SR3      ;SEE IF ECO #8 HAS BEEN INSTALLED       ;DPM001
2062 021472  001003                             BNE     45$             ;BRANCH TO ECO #8 COMPARE IF SO         ;DPM001
2063 021474  012701  021400                     MOV     #10$,R1         ;LOAD EXPECTED CONTENTS OF SR2 INTO R1
2064 021500  000402                             BR      46$             ;BRANCH TO DO THE COMPARE               ;DPM001
2065 021502  012701  021406             45$:    MOV     #2$,R1          ;LOAD EXPECTED CONTENTS OF SR2 INTO R1  ;DPM001
2066 021506  020137  001270             46$:    CMP     R1,WASSR2       ;DID SR2 LOCKUP VIRT. ADDR OF ABORTED INST.
2067 021512  001401                             BEQ     5$              ;BRANCH IF YES
2068 021514  104013                             ERROR   +13             ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ILL. MODE INST.
2069                                             ;FOR TIGHTER SCOPE LOOP, REPLACE ERROR CALL WITH 'BR 10$'' = 000731
2070 021516  042737  160000  177572     5$:     BIC     #160000,SR0     ;CLEAR POSSIBLE ERROR BITS IN SR0
2071                                    ;*****************************************************************
2072                                    ;*
2073                                    ;*      THE NEXT TWO (2) TESTS WILL BE CHECKING THE PAGE LENGTH
2074                                    ;*      COMPARATORS AND SOME MORE OF THE KT ERROR DETECTION
2075                                    ;*      AND STATUS LOGIC.  THE PAGE LENGTH FIELD (PLF) IN KERNEL
2076                                    ;*      PDR 4 IS VARIED AND FOR EVERY PLF, THREE (3) VIRTUAL
2077                                    ;*      ADDRESSES ARE READ.  WHILE USING BOTH UPWARD & DOWNWARD PAGE
2078                                    ;*      EXPANSION, ONE OF THOSE THREE VIRTUAL ADDRESSES WILL CAUSE A
2079                                    ;*      "PAGE LENGTH ABORT" WHILE THE OTHER TWO WON'T.
2080                                    ;*
2081                                    ;*      STATUS REGISTER 0 & 2 ARE CHECKED WHEN THE PAGE LENGTH
2082                                    ;*      ABORT DOES OCCUR TO SEE THAT THE ABORT IS REPORTED AND THAT
2083                                    ;*      THE VIRTUAL ADDRESS OF THE INSTRUCTION THAT CAUSED THE ABORT
2084                                    ;*      IS LOCKED UP.
2085                                    ;*
2086                                    ;*****************************************************************
```

```
2098                                    .SBTTL  TEST # 4 - PAGE LENGTH FAULTS-UPWARD EXPANSION
                                        ;*****************************************************************
                                        ;*TEST 4        PAGE LENGTH FAULTS-UPWARD EXPANSION
                                        ;*
                                        ;*      THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR 4
                                        ;*      FROM 1 TO 177 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
                                        ;*      ARE ACCESSED.  WHEN VBA <12:6> IS LESS THAN OR EQUAL TO PDR <14:8>
                                        ;*      NO ABORT SHOULD OCCUR.  WHEN VBA <12:6> IS GREATER THAN PDR <14:8>,
                                        ;*      A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED BY SR0 & SR2.
                                        ;*      THE PAGE EXPANSION DIRECTION IN THIS TEST IS UPWARD, (THE ED BIT
                                        ;*      (BIT 3) OF PDR 4 = 0).
                                        ;*
                                        ;*****************************************************************
     021524  000004                     TST4:   SCOPE
2099 021526  012737  077406  172306     1$:     MOV     #77406,KIPDR3   ;MAKE SURE PDR3 IS DESCRIBED AS R/W
2100 021534  012737  077406  172312             MOV     #77406,KIPDR5   ;MAKE SURE PDR5 IS DESCRIBED AS R/W
2101 021542  012700  100000                     MOV     #100000,R0      ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
2102 021546  012704  000406                     MOV     #406,R4         ;LOAD FIRST PDR VALUE IN R4 (PLF=1, ACF=6)
2103 021552  012737  021744  000250     2$:     MOV     #9$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
2104 021560  010437  172310                     MOV     R4,KIPDR4       ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
2105 021564  012737  021572  001110             MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2106 021572  012706  001100             3$:     MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
2107 021576  011001                             MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA < PLF - NO ABORT)
2108 021600  062700  000100                     ADD     #100,R0         ;FORM NEXT VIRTUAL ADDRESS IN R0
2109 021604  012737  021612  001110             MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
2110 021612  012706  001100             4$:     MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
2111 021616  011001                             MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
2112 021620  062700  000100                     ADD     #100,R0         ;FORM NEXT VIRTUAL ADDR IN R0
2113 021624  020027  117700                     CMP     R0,#117700      ;HAVE ALL PLF'S BEEN TESTED YET?
2114 021630  001470                             BEQ     10$             ;BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED
2115 021632  012737  021646  001110             MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
2116 021640  012737  021654  000250             MOV     #6$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
2117 021646  011001                     5$:     MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA > PLF - ABORT TO 6$)
2118 021650  104010                             ERROR   +10             ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
2119                                                                     ;FOR TIGHTER SCOPE LOOP
2120                                                                     ;REPLACE ERROR CALL WITH
2121                                                                     ;'BR 5$'' = 000776
2122 021652  000424                             BR      8$              ;BRANCH AROUND ABORT CHECKS
2123 021654  012706  001100             6$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER FOLLOWING ABORT
2124 021660  013737  177572  001264             MOV     SR0,WASSR0      ;READ M.M. STATUS REG. 0
2125 021666  013737  177576  001270             MOV     SR2,WASSR2      ;READ M.M. STATUS REG. 2
2126 021674  012702  040011                     MOV     #40011,R2       ;PUT EXPECTED SR0 CONTENTS IN R2
2127 021700  020237  001264                     CMP     R2,WASSR0       ;DID SR0 REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
2128 021704  001401                             BEQ     7$              ;BRANCH IF YES
2129 021706  104011                             ERROR   +11             ;SR0 DID NOT REPORT PG. LENGTH ABORT CORRECTLY
2130                                                                     ;FOR TIGHTER SCOPE LOOP
2131                                                                     ;REPLACE ERROR CALL WITH
2132                                                                     ;'BR 5$'' = 000757
2133 021710  012703  021646             7$:     MOV     #5$,R3          ;PUT EXPECTED SR2 CONTENTS IN R3
2134 021714  020337  001270                     CMP     R3,WASSR2       ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
2135 021720  001401                             BEQ     8$              ;BRANCH IF YES
2136 021722  104012                             ERROR   +12             ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
2137                                                                     ;FOR TIGHTER SCOPE LOOP
2138                                                                     ;REPLACE ERROR CALL WITH
2139                                                                     ;'BR 5$'' = 000751
2140 021724  042737  160000  177572     8$:     BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
2141 021732  062704  000400                     ADD     #400,R4         ;FORM NEXT PLF VALUE FOR KIPDR4
```

```
2142 021736  162700  000100                    SUB     #100,R0         ;FORM FIRST VIRT. ADDR FOR THAT PLF VALUE
2143 021742  000703                            BR      2$              ;BRANCH BACK AND ACCESS 3 VBA'S FOR
2144                                                                   ;THE PLF VALUE JUST FORMED
2145 021744  012637  001260          9$:       MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
2146 021750  012637  001262                    MOV     (KSP)+,TRAPPS
2147 021754  013737  177572  001264            MOV     SR0,WASSR0      ;SAVE CONTENTS OF SR0 FOR ERROR
2148 021762  013737  177576  001270            MOV     SR2,WASSR2      ;SAVE CONTENTS OF SR2 FOR ERROR
2149 021770  042737  160000  177572            BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
2150 021776  104007                            ERROR   +7              ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
2151                                                                   ;FOR TIGHTER SCOPE LOOP
2152                                                                   ;REPLACE ERROR CALL WITH
2153                                                                   ;A 'NOP' = 000240
2154 022000  013746  001262                    MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
2155 022004  013746  001260                    MOV     TRAPPC,-(KSP)
2156 022010  000002                            RTI                     ;RETURN FROM UNEXPECTED ABORT
2157
2158 022012  012737  015726  000250  10$:      MOV     #MGMERR,MMVEC   ;RESTORE NORMAL M.M. TRAP HANDLER
2159                                                                   ;ADDRESS TO M.M. TRAP VECTOR
```

```
2160
                .SBTTL TEST #5 - PAGE LENGTH FAULTS-DOWNWARD EXPANSION
```

```
2172                                    .SBTTL  TEST # 5 - PAGE LENGTH FAULTS-DOWNWARD EXPANSION
                                        ;*****************************************************************
                                        ;*TEST 5        PAGE LENGTH FAULTS-DOWNWARD EXPANSION
                                        ;*
                                        ;*      THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR4
                                        ;*      FROM 176 TO 0 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
                                        ;*      ARE ACCESSED.  WHEN VBA <12:6> IS GREATER THAN OR EQUAL TO PDR <14:8>
                                        ;*      NO PAGE ABORT SHOULD OCCUR.  WHEN VBA <12:6> IS LESS THAN PDR <14:8>
                                        ;*      A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTFD BY SR0 & SR2.
                                        ;*      THE PAGE EXPANSION DIRECTION IN THIS TEST IS DOWNWARD, (THE ED BIT
                                        ;*      (BIT 3) OF PDR4=1).
                                        ;*
                                        ;*****************************************************************
     022020  000004                     TST5:   SCOPE
2173 022022  012700  117700             1$:     MOV     #117700,R0      ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
2174 022026  012704  077016                     MOV     #77016,R4       ;LOAD FIRST PDR VALUE IN R4 (PLF=176,ACF=6)
2175 022032  012737  022224  000250     2$:     MOV     #9$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
2176 022040  010437  172310                     MOV     R4,KIPDR4       ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
2177 022044  012737  022052  001110             MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2178 022052  012706  001100             3$:     MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
2179 022056  011001                             MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA > PLF - NO ABORT)
2180 022060  162700  000100                     SUB     #100,R0         ;FORM NEXT VIRTUAL ADDRESS IN R0
2181 022064  012737  022072  001110             MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
2182 022072  012706  001100             4$:     MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
2183 022076  011001                             MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
2184 022100  162700  000100                     SUB     #100,R0         ;FORM NEXT VIRTUAL ADDR. IN R0
2185 022104  020027  100000                     CMP     R0,#100000      ;HAVE ALL PLF'S BEEN TESTED YET?
2186 022110  001470                             BEQ     10$             ;BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED
2187 022112  012737  022126  001110             MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
2188 022120  012737  022134  000250             MOV     #6$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
2189 022126  011001                     5$:     MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA < PLF - ABORT TO 6$)
2190 022130  104010                             ERROR   +10             ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
2191                                            ;FOR TIGHTER SCOPE LOOP
2192                                            ;REPLACE ERROR CALL WITH
2193                                            ;'BR 5$'' = 000776
2194 022132  000424                             BR      8$              ;BRANCH AROUND ABORT CHECKS
2195 022134  012706  001100             6$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER FOLLOWING ABORT
2196 022140  013737  177572  001264             MOV     SR0,WASSR0      ;READ M.M. STATUS REG. 0
2197 022146  013737  177576  001270             MOV     SR2,WASSR2      ;READ M.M. STATUS REG. 2
2198 022154  012702  040011                     MOV     #40011,R2       ;PUT EXPECTED SR0 CONTENTS IN R2
2199 022160  020237  001264                     CMP     R2,WASSR0       ;DID SR0 REPORT PG. LENGTH ABORT, PG. 4, KERNEL?
2200 022164  001401                             BEQ     7$              ;BRANCH IF YES
2201 022166  104011                             ERROR   +11             ;SR0 DID NOT REPORT PG. LENGTH ABORT CORRECTLY
2202                                            ;FOR TIGHTER SCOPE LOOP
2203                                            ;REPLACE ERROR CALL WITH
2204                                            ;'BR 5$'' = 000757
2205 022170  012703  022126             7$:     MOV     #5$,R3          ;PUT EXPECTED SR2 CONTENTS IN R3
2206 022174  020337  001270                     CMP     R3,WASSR2       ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
2207 022200  001401                             BEQ     8$              ;BRANCH IF YES
2208 022202  104012                             ERROR   +12             ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
2209                                            ;FOR TIGHTER SCOPE LOOP
2210                                            ;REPLACE ERROR CALL WITH
2211                                            ;'BR 5$'' = 000751
2212 022204  042737  160000  177572     8$:     BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
2213 022212  162704  000400                     SUB     #400,R4         ;FORM NEXT PLF VALUE FOR KIPDR4
2214 022216  062700  000100                     ADD     #100,R0         ;FORM FIRST VIRT. ADDR. FOR THAT PLF VALUE
2215 022222  000703                             BR      2$              ;BRANCH BACK AND ACCESS 3 VBA'S FOR
```

```
2216                                                                    ;THE PLF VALUE JUST FORMED
2217 022224  012637  001260          9$:    MOV     (KSP)+,TRAPPC       ;SAVE PC & PS OF TRAP
2218 022230  012637  001262                 MOV     (KSP)+,TRAPPS
2219 022234  013737  177572  001264        MOV     SR0,WASSR0          ;SAVE CONTENTS OF SR0 FOR ERROR
2220 022242  013737  177576  001270        MOV     SR2,WASSR2          ;SAVE CONTENTS OF SR2 FOR ERROR
2221 022250  042737  160000  177572        BIC     #160000,SR0         ;CLEAR ERROR BITS IN SR0
2222 022256  104007                        ERROR   +7                  ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
2223                                                                    ;FOR TIGHTER SCOPE LOOP
2224                                                                    ;REPLACE ERROR CALL WITH
2225                                                                    ;A 'NOP' = 000240
2226 022260  013746  001262                 MOV     TRAPPS,-(KSP)       ;PUT PC & PS OF TRAP ON STACK
2227 022264  013746  001260                 MOV     TRAPPC,-(KSP)
2228 022270  000002                        RTI                         ;RETURN FROM UNEXPECTED ABORT
2229
2230 022272  012737  015726  000250  10$:    MOV     #MGMERR,MMVEC       ;RESTORE NORMAL M.M. TRAP HANDLER
2231                                                                    ;ADDRESS TO M.M. TRAP VECTOR
```

```
2245                                    .SBTTL  TEST # 6 - SR2 BIT TEST
                                        ;*****************************************************************
                                        ;*TEST 6        SR2 BIT TEST
                                        ;*
                                        ;*      THIS TEST CHECKS THE BITS IN MEMORY MANAGEMENT REGISTER 2 BY
                                        ;*      CAUSING 'READ-ONLY ABORTS'' AT VIRTUAL ADDRESSES BETWEEN 100000
                                        ;*      TO 111000 (PHYSICAL ADDRESSES 060000-071000).  KIPDR4 IS USED TO EXECUTE
                                        ;*      THE FOLLOWING FOUR WORDS OF CODE WHICH ARE MOVED THRU MEMORY:
                                        ;*      010727  MOV     PC,(PC)+  ;THIS INSTRUCTION SHOULD CAUSE A R/O ABORT
                                        ;*      000000                    ;ITS VIRTUAL ADDR. SHOULD BE LOCKED UP IN SR2
                                        ;*      000137  JMP     @#3$      ;THIS INSTRUCTION IS ALSO MOVED THRU MEMORY
                                        ;*      (ADDR. OF 3$)             ;IN CASE A R/O ABORT DOES NOT OCCUR,
                                        ;*                                ;IN WHICH CASE SR2 WILL NOT CONTAIN CORRECT ADDR.
                                        ;*
                                        ;*****************************************************************
     022300  000004                    TST6:   SCOPE
2246 022302  012737  000600  172346    1$:     MOV     #600,KIPAR3       ;BE SURE PAR3 IS MAPPED TO 12-16K
2247 022310  012737  000600  172350            MOV     #600,KIPAR4       ;BE SURE PAR4 IS MAPPED TO 12-16K
2248 022316  012737  077406  172306            MOV     #77406,KIPDR3     ;MAP PAGE 3  128 BLOCKS, R/W
2249 022324  012737  077402  172310            MOV     #77402,KIPDR4     ;MAP PAGE 4  128 BLOCKS, READ-ONLY
2250 022332  012700  060000                    MOV     #60000,R0         ;LOAD R0 WITH VIRTUAL ADDR. WHICH USES PDR3
2251 022336  012701  100000                    MOV     #100000,R1        ;LOAD R1 WITH VIRTUAL ADDR. WHICH USES PDR4
2252 022342  012737  022376  000250            MOV     #3$,MMVEC         ;SET M.M. TRAP VECTOR TO 3$
2253 022350  012737  022356  001110            MOV     #2$,$LPERR        ;SET LOOP ON ERROR POINTER TO 2$
2254 022356  012720  010727            2$:     MOV     #010727,(R0)+     ;LOAD 'MOV PC,(PC)+'' INSTRUCTION AT ADDR.
2255 022362  005020                            CLR     (R0)+             ;       REACHED THRU PDR/PAR 4.
2256 022364  012720  000137                    MOV     #000137,(R0)+     ;LOAD ''JMP @#3$'' INSTRUCTION AT VIRT. ADDR.
2257 022370  012710  022376                    MOV     #3$,(R0)          ;       IN CASE R/O VIOL. DOES NOT ABORT
2258 022374  010107                            MOV     R1,PC             ;TRANSFER PROGRAM EXECUTION TO 'PAGE 4 INSTRUCTIONS''
2259 022376  012706  001100            3$:     MOV     #KERSTK,KSP       ;RESTORE STACK POINTER
2260 022402  013737  177576  001270            MOV     SR2,WASSR2        ;READ CONTENTS OF STATUS REG 2
2261 022410  020137  001270                    CMP     R1,WASSR2         ;WAS ADDR. OF 'RELOCATED - R/O ABORT'' LOCKED UP?
2262 022414  001401                            BEQ     4$                ;BRANCH IF YES
2263 022416  104013                            ERROR   +13               ;SR2 DID NOT LOCK UP VIRTUAL ADDR. OF R/O VIOL.
2264                                                                      ;FOR TIGHTER SCOPE LOOP
2265                                                                      ;REPLACE ERROR CALL WITH
2266                                                                      ;'BR 2$'' = 000757
2267 022420  042737  160000  177572  4$:     BIC     #160000,SR0       ;CLEAR THE ERROR BITS IN SR0
2268 022426  162700  000004                    SUB     #4,R0             ;RESET R0 TO POINT TO NEXT VIRT. ADDR. TO LOAD
2269 022432  062701  000002                    ADD     #2,R1             ;FORM VIRTUAL ADDR. THAT SHOULD BE LOCKED UP NEXT
2270 022436  020127  111002                    CMP     R1,#111002        ;HAVE ALL VBA'S 100000-111000 BEEN TESTED?
2271 022442  103745                            BLO     2$                ;BRANCH IF NO
2272
2273 022444  012737  077406  172310  5$:     MOV     #77406,KIPDR4     ;RESTORE PDR4 TO R/W ACCESS
2274 022452  012737  015726  000250            MOV     #MGMERR,MMVEC     ;RESTORE ADDRESS OF NORMAL M.M.
2275                                                                      ;TRAP HANDLER TO M.M. VECTOR
```

```
2289                                    .SBTTL  TEST # 7 - MORE CHECKS OF SR0 & SR2
                                        ;*****************************************************************
                                        ;*TEST 7        MORE CHECKS OF SR0 & SR2
                                        ;*
                                        ;*      THIS TEST PERFORMS SOME ADDITIONAL CHECKS OF THE SR0 & SR2 LOGIC.
                                        ;*      FIRST IT CHECKS THAT SR2 'TRACKS' ALONG ACTING AS A VIRTUAL ADDRESS
                                        ;*      PROGRAM COUNTER.  ALSO SR0 & SR2 ARE LOCKED UP BY A PAGE LENGTH
                                        ;*      ABORT, THEN WITHOUT CLEARING SR0'S ERROR BITS, A R/O ABORT IS CAUSED.
                                        ;*      SR0 & SR2 SHOULD NOT BE CHANGED BY THE SECOND ABORT AND THE
                                        ;*      INFORMATION ABOUT THE PAGE LENGTH ABORT SOULD STILL BE LOCKED UP.
                                        ;*      IN ADDITION A 'RESET' IS EXECUTED TO VERIFY THAT SR0 IS CLEARED
                                        ;*      AND SR2 IS UNLOCKED BY A RESET.  AFTER MEMORY MANAGEMENT IS TURNED BACK ON,
                                        ;*      SR2 IS CHECKED TO SEE THAT IT IS TRACKING AGAIN.
                                        ;*
                                        ;*****************************************************************
     022460  000004                     TST7:   SCOPE
2290 022462  012737  000600  172352     1$:     MOV     #600,KIPAR5     ;MAP KERNEL PAGE 5 TO 12-16K
2291 022470  012737  000406  172310             MOV     #406,KIPDR4     ;SETUP PDR4 FOR PAGE LENGTH ABORT
2292 022476  012737  077402  172312             MOV     #77402,KIPDR5   ;SETUP PDR5 FOR R/O ABORT
2293 022504  012737  022512  001110             MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2294 022512  013737  177576  001270     2$:     MOV     SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING
2295 022520  012701  022512                     MOV     #2$,R1          ;PUT EXPECTED VIRTUAL PC IN R1
2296 022524  020137  001270                     CMP     R1,WASSR2       ;DID SR2 CONTAIN VIRTUAL PC AT 2$?
2297 022530  001401                             BEQ     3$              ;BRANCH IF YES
2298 022532  104016                             ERROR   +16             ;SR2 NOT TRACKING CORRECTLY
2299                                                                     ;FOR TIGHTER SCOPE LOOP
2300                                                                     ;REPLACE ERROR CALL WITH
2301                                                                     ;'BR 2$' = 000767
2302 022534  012737  022542  001110     3$:     MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
2303 022542  013737  177576  001270     4$:     MOV     SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING
2304 022550  012701  022542                     MOV     #4$,R1          ;PUT EXPECTED VIRTUAL PC IN R1
2305 022554  020137  001270                     CMP     R1,WASSR2       ;DID SR2 CONTAIN VIRTUAL PC AT 4$
2306 022560  001401                             BEQ     5$              ;BRANCH IF YES
2307 022562  104016                             ERROR   +16             ;SR2 NOT TRACKING CORRECTLY
2308                                                                     ;FOR TIGHTER SCOPE LOOP
2309                                                                     ;REPLACE ERROR CALL WITH
2310                                                                     ;'BR 4$' = 000767
2311 022564  012737  022572  001110     5$:     MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
2312 022572  012737  022610  000250     6$:     MOV     #7$,MMVEC       ;PUT ADDRESS OF 7$ IN M.M. TRAP VECTOR
2313 022600  005037  001200                     CLR     $TMP1           ;CLEAR ERROR INDICATOR
2314 022604  005237  100500                     INC     @#100500        ;CAUSE PAGE LENGTH ABORT - TRAP TO 7$
2315 022610  012706  001100             7$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER AFTER ABORT
2316 022614  013737  177572  001176             MOV     SR0,$TMP0       ;SAVE SR0'S INFORMATION ON PG. LGTH. ABORT
2317 022622  013737  177576  001202             MOV     SR2,$TMP2       ;SAVE SR2'S INFORMATION ON PG. LGTH. ABORT
2318 022630  012737  022642  000250             MOV     #8$,MMVEC       ;PUT ADDRESS OF 8$ IN M.M. TRAP VECTOR
2319 022636  005237  120000                     INC     @#120000        ;CAUSE R/O ABORT - TRAP TO 8$
2320 022642  012706  001100             8$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER AFTER ABORT
2321 022646  013737  177572  001264             MOV     SR0,WASSR0      ;READ SR0 FOLLWOING SECOND KT ABORT
2322 022654  013737  177576  001270             MOV     SR2,WASSR2      ;READ SR2 FOLLOWING SECOND KT ABORT
2323 022662  023737  001176  001264             CMP     $TMP0,WASSR0    ;IS SR0 STILL HOLDING INFO ON FIRST ABORT?
2324 022670  001402                             BEQ     9$              ;BRANCH IF YES
2325 022672  005237  001200                     INC     $TMP1           ;SET ERROR INDICATOR
2326 022676  023737  001202  001270     9$:     CMP     $TMP2,WASSR2    ;DOES SR2 STILL HOLD PC OF FIRST ABORT?
2327 022704  001402                             BEQ     10$             ;BRANCH IF YES
2328 022706  005237  001200                     INC     $TMP1           ;SET ERROR INDICATOR
2329 022712  005737  001200             10$:    TST     $TMP1           ;WERE SR0 OR SR2 CHANGED BY A SECOND ABORT?
2330 022716  001401                             BEQ     11$             ;BRANCH IF NO
```

```
2331 022720  104014                          ERROR   +14             ;ONE OF STATUS REGS. CHANGED BY SECOND ABORT
2332                                                                 ;FOR TIGHTER SCOPE LOOP
2333                                                                 ;REPLACE ERROR CALL WITH
2334                                                                 ;'BR 6$'' = 000726
2335 022722  005037  001200          11$:    CLR     $TMP1           ;CLEAR ERROR INDICATOR
2336 022726  000005                          RESET                   ;EXECUTE A RESET, APPLYING AN ''INIT''
2337 022730  013737  177572  001264          MOV     SR0,WASSR0      ;READ SR0
2338 022736  005737  001264                  TST     WASSR0          ;WAS SR0 CLEARED BY THE RESET?
2339 022742  001402                          BEQ     12$             ;BRANCH IF YES
2340 022744  005237  001200                  INC     $TMP1           ;SR0 NOT CLEARED BY A RESET
2341 022750  013737  177576  001270  12$:    MOV     SR2,WASSR2      ;READ SR2
2342 022756  022737  022750  001270          CMP     #12$,WASSR2     ;WAS SR2 UNLOCKED BY A RESET?
2343 022764  001402                          BEQ     13$             ;BRANCH IF YES
2344 022766  005237  001200                  INC     $TMP1           ;SR2 NOT UNLOCKED BY A RESET
2345 022772  005737  001200          13$:    TST     $TMP1           ;WERE SR0 & SR2 BOTH 'RESET'' BY A RESET?
2346 022776  001401                          BEQ     14$             ;BRANCH IF YES
2347 023000  104015                          ERROR   +15             ;SR0 OR SR2 NOT 'RESET'' BY A RESET
2348                                                                 ;FOR TIGHTER SCOPE LOOP
2349                                                                 ;REPLACE ERROR CALL WITH
2350                                                                 ;'BR 6$'' = 000676
2351 023002  005237  177572          14$:    INC     SR0             ;TURN MEMORY MANAGEMENT BACK ON
2352 023006  013737  177576  001270  15$:    MOV     SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING AGAIN
2353 023014  012701  023006                  MOV     #15$,R1         ;PUT EXPECTED VIRTUAL PC IN R1
2354 023020  020137  001270                  CMP     R1,WASSR2       ;DID SR2 CONTAIN VIRTUAL PC AT 15$
2355 023024  001401                          BEQ     16$             ;BRANCH IF YES
2356 023026  104016                          ERROR   +16             ;SR2 NOT TRACKING CORRECTLY
2357                                                                 ;FOR TIGHTER SCOPE LOOP
2358                                                                 ;REPLACE ERROR CALL WITH
2359                                                                 ;'BR 6$'' = 000663
2360 023030  012737  077406  172310  16$:    MOV     #77406,KIPDR4   ;RESET PDR4 TO 128 BLKS, R/W
2361 023036  012737  077406  172312          MOV     #77406,KIPDR5   ;RESET PDR5 TO 128 BLKS, R/W
2362 023044  012737  015726  000250          MOV     #MGMERR,MMVEC   ;RESTORE ADDRESS OF NORMAL MEMORY
2363                                                                 ;MANAGEMENT TRAP ROUTINE TO M.M. VECTOR
```

TEST # 10 - SUPER/USER ABORT PICKS UP KERNEL VECTOR

```
2377                                    .SBTTL  TEST # 10 - SUPER/USER ABORT PICKS UP KERNEL VECTOR
                                        ;*****************************************************************
                                        ;*TEST 10       SUPER/USER ABORT PICKS UP KERNEL VECTOR
                                        ;*
                                        ;*      THIS TEST CHECKS TO BE SURE THAT WHEN AN ABORT OCCURS WHILE
                                        ;*      IN SUPERVISOR OR USER MODE, THE TRAP VECTOR INFORMATION
                                        ;*      FETCHED IS TAKEN FROM KERNEL SPACE.  USER PAGE 0 IS MAPPED
                                        ;*      TO 12K (60000-77776) SO THAT IF USER SPACE IS USED INSTEAD
                                        ;*      OF KERNEL, THE NEW PC THAT WAS LOADED AT LOC. 060004 IS USED
                                        ;*      INSTEAD OF THE NEW PC THAT SHOULD BE PICKED UP FROM LOC. 000004.
                                        ;*      THE SUPERVISOR PAGE 0 IS THEN MAPPED TO 12K, AND THE TEST
                                        ;*      IS REPEATED FOR SUPERVISOR MODE.  AN ODD ADDRFSS ERROR IS
                                        ;*      USED TO CAUSE A TRAP TO "4".
                                        ;*
                                        ;*****************************************************************
     023052  000004                     TST10:  SCOPE
2378 023054  004737  002346             1$:     JSR     PC,TOFF         ;TURN OFF T-BIT TRAPPING FOR THIS TEST
2379 023060  012737  023066  001110             MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2380 023066  005037  177776             2$:     CLR     PSW             ;GO TO KERNEL MODE
2381 023072  012706  001100                     MOV     #KERSTK,KSP     ;SETUP KERNEL STACK PTR.
2382                                    ;*
2383                                    ;*      TEST USER MODE ABORT
2384                                    ;*
2385 023076  012737  000600  177640             MOV     #600,UIPAR0     ;MAP USER PAGE 0 TO 12K
2386 023104  012737  023166  000004             MOV     #4$,@#4         ;LOAD KERNEL VECTOR 4 (LOC.4) WITH 4$
2387 023112  012737  000340  000006             MOV     #340,@#6        ;LOAD VECTOR+2 WITH NEW PSW
2388 023120  012737  140000  177776             MOV     #140000,PSW     ;GO TO USER MODE
2389 023126  012706  000600                     MOV     #USESTK,USP     ;SETUP USER STACK PTR.
2390 023132  012737  023152  000004             MOV     #3$,@#4         ;LOAD USER VECTOR 4 (LOC. 60004) WITH 3$
2391 023140  012737  000340  000006             MOV     #340,@#6        ;LOAD VECTOR+2 WITH NEW PSW
2392 023146  005737  023153                     TST     3$+1            ;CAUSE ODD ADDR. ERROR TRAP TO "4"
2393                                                                    ;SHOULD PICK UP NEW PC=4$ FROM KERNEL
2394                                                                    ;LOC. 4, NOT PC=3$ FROM USER LOC. 4 (=60004)
2395 023152  013701  177776             3$:     MOV     PSW,R1          ;SAVE PSW FOR ERROR
2396 023156  010602                             MOV     SP,R2           ;SAVE VALUE OF STACK POINTER FOR ERROR
2397 023160  005037  177776                     CLR     PSW             ;BE SURE BACK IN KERNEL MODE
2398 023164  104017                             ERROR   +17             ;DID NOT TRAP THRU KERNEL SPACE
2399                                                                    ;FOR TIGHTER SCOPE LOOP
2400                                                                    ;REPLACE ERROR CALL WITH
2401                                                                    ;"BR 2$" = 000740
2402 023166  005037  177776             4$:     CLR     PSW             ;BE SURE BACK IN KERNEL MODE
2403 023172  012706  001100                     MOV     #KERSTK,KSP     ;RESTORE KERNEL S.P. IN CASE IT CHANGED
2404 023176  005037  177640                     CLR     UIPAR0          ;REMAP USER PAGE 0 TO 0-4K
2405 023202  012737  140000  177776             MOV     #140000,PSW     ;GO TO USER MODE
2406 023210  012706  000600                     MOV     #USESTK,USP     ;RESTORE USER STACK POINTER
2407 023214  005037  177776                     CLR     PSW             ;GO BACK TO KERNAL MODE
2408                                    ;*
2409                                    ;*      NOW TEST THE SUPERVISOR MODE ABORT
2410                                    ;*
2411 023220  012737  000600  172240             MOV     #600,SIPAR0     ;MAP SUPERVISOR PAGE 0 TO 12K
2412 023226  012737  023310  000004             MOV     #6$,@#4         ;LOAD KERNAL VECTOR 4 WITH 6$
2413 023234  012737  000340  000006             MOV     #340,@#6        ;LOAD VECTOR+2 WITH NEW PSW
2414 023242  012737  040000  177776             MOV     #40000,PSW      ;GO TO SUPERVISOR MODE
2415 023250  012706  000700                     MOV     #SUPSTK,SSP     ;SETUP SUPERVISOR STACK PTR.
2416 023254  012737  023274  000004             MOV     #5$,@#4         ;LOAD SUPERVISOR VECTOR 4 (LOC. 60004) WITH 5$
2417 023262  012737  000340  000006             MOV     #340,@#6        ;LOAD VECTOR+2 WITH NEW PSW
2418 023270  005737  023275                     TST     5$+1            ;CAUSE ODD ADDR. ERROR TRAP TO "4"
```

```
                                                                ;SHOULD PICK UP NEW PC 6$ FROM KERNEL
                                                                ;LOC. 4, NOT PC=5$ FROM SUPERVISOR LOC. 4 (-60004)
023274  013701  177776          5$:   MOV     PSW,R1            ;SAVE PSW FOR ERROR
023300  010602                      MOV     SP,R2             ;SAVE VALUE OF STACK POINTER FOR ERROR
023302  005037  177776              CLR     PSW               ;BE SURE BACK IN KERNEL MODE
023306  104017                      ERROR   +17               ;DID NOT TRAP THRU KERNEL SPACE
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;'BR 2$'' = 000740
023310  005037  177776          6$:   CLR     PS#               ;BE SURE BACK IN KERNEL MODE
023314  012706  001100              MOV     #KERSTK,KSP       ;RESTORE KERNEL S.P. IN CASE IT CHANGED
023320  005037  172240              CLR     SIPAR0            ;REMAP SUPERVISOR PAGE 0 TO 0-4K
023324  012737  040000  177776      MOV     #40000,PSW        ;GO TO SUPERVISOR MODE
023332  012706  000700              MOV     #SUPSTK,SSP       ;RESTORE SUPERVISOR STACK POINTER
023336  005037  177776              CLR     PSW               ;GO BACK TO KERNEL MODE
023342  012737  015654  000004      MOV     #TIMERR,@#4       ;RESTORE ADDR. OF NORMAL CPU TRAP HANDLER TO 4
023350  004737  002402              JSR     PC,TON            ;TURN T-BIT TRAPPING BACK ON
```

```
2442                                    .SBTTL  TEST # 11 - RTI IN SUPER/USER MODE DOES NOT CHANGE PSW
                                        ;;*****************************************************************
                                        ;*TEST 11       RTI IN SUPER/USER MODE DOES NOT CHANGE PSW
                                        ;*
                                        ;*      THIS TEST CHECKS TO SEE THAT WHEN AN RTI IS EXECUTED IN SUPERVISOR
                                        ;*      OR USER MODE, THE MODE OR PRIORITY BITS OF THE PSW ARE NOT CHANGED.
                                        ;*
                                        ;;*****************************************************************
     023354  000004                     TST11:  SCOPE
2443
2444 023356  012737  023370  001110    1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2445 023364  012702  170000             MOV     #170000,R2      ;LOAD 'PRESENT & EXPECTED'' PSW VALUE INTO R2
2446 023370  010237  177776             2$:     MOV     R2,PSW          ;GO TO PRESENT MODE-PRIORITY 0
2447 023374  012746  000340                     MOV     #340,-(SP)      ;PUT A NEW PSW (PRIORITY=7) ON STACK
2448 023400  012746  023406                     MOV     #3$,-(SP)       ;PUT NEW PC ON THE STACK
2449 023404  000002                             RTI                     ;DO AN RTI FROM PRESENT MODE
2450 023406  013701  177776             3$:     MOV     PSW,R1          ;READ NEW PSW INTO R1
2451 023412  042701  007437                     BIC     #7437,R1        ;MASK OFF COND. CODE, T-BIT, AND UNUSED BITS
2452 023416  005037  177776                     CLR     PSW             ;GO BACK TO KERNEL MODE
2453 023422  020201                             CMP     R2,R1           ;DID PSW STAY IN PRESENT MODE, PRIORITY-0?
2454 023424  001401                             BEQ     4$              ;BRANCH IF YES
2455 023426  104032                             ERROR   +32             ;PSW CHANGED BY AN RTI FROM USER
2456                                                                     ;FOR A TIGHTER SCOPE LOOP
2457                                                                     ;REPLACE ERROR CALL WITH
2458                                                                     ;'BR=2$'' = 000760
2459 023430  022702  050000             4$:     CMP     #50000,R2       ;IF SUPERVISOR MODE HAS BEEN CHECKED,
2460 023434  001403                             BEQ     TST12           ;;GO TO NEXT TEST
2461 023436  012702  050000                     MOV     #50000,R2       ;ELSE,SET SUPERVISOR MODE,
2462 023442  000752                             BR      2$              ;AND BRANCH BACK TO TEST IT.
```

```
2474                                    .SBTTL  TEST # 12 - KT ERROR NOT SERVICED IF ODD ADDR. ERROR
                                        ;*****************************************************************
                                        ;*TEST 12       KT ERROR NOT SERVICED IF ODD ADDR. ERROR
                                        ;*
                                        ;*      THIS TEST CHECKS TO SEE THAT IF A CERTAIN VIRTUAL ADDRESS THAT
                                        ;*      WOULD CAUSE A MEMORY MANAGEMENT ERROR CAUSES AN ODD ADDRESS
                                        ;*      ERROR FIRST, THE ODD ADDRESS ERROR IS SERVICED BUT THE MEMORY
                                        ;*      MANAGEMENT ERROR ISN'T.  THIS MEANS THAT SR0 AND SR2
                                        ;*      SHOULD NOT REPORT THE ERROR OR LOCK UP ITS VIRTUAL ADDRESS.
                                        ;*      A READ-ONLY VIOLATION IS USED AS THE POTENTIAL MEMORY MANAGEMENT
                                        ;*      ERROR
                                        ;*
                                        ;*****************************************************************
     023444  000004                     TST12:  SCOPE
2475 023446  012737  000600  172350     1$:     MOV     #600,KIPAR4     ;MAP KERNEL PAGE 4 TO 12-16K
2476 023454  012705  077402             1$:     MOV     #77402,R5       ;LOAD PDR4 DATA INTO R5
2477 023460  010537  172310             1$:     MOV     R5,KIPDR4       ;MAP PAGE 4 READ-ONLY
2478 023464  012737  023514  000004     1$:     MOV     #4$,@#4         ;SET CPU TRAP VECTOR TO ADDRESS OF 4$
2479 023472  012737  023512  000250     1$:     MOV     #3$,@#250       ;SET M.M. TRAP VECTOR TO ADDRESS OF 3$
2480 023500  012737  023506  001110     1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2481 023506  005237  060001             2$:     INC     60001           ;CAUSE ODD ADDR. ERROR & POTENTIAL R/O ABORT
2482 023512  104020                     3$:     ERROR   +20             ;TRAPPED THRU M.M. VECTOR BU' SHOULDN'T HAVE
2483                                                                     ;FOR TIGHTER SCOPE LOOP
2484                                                                     ;REPLACE ERROR CALL WITH
2485                                                                     ;'BR 2$'' = 000776
2486 023514  012706  001100             4$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER AFTER TRAPPING
2487 023520  005037  001200                     CLR     $TMP1           ;CLEAR ERROR INDICATOR
2488 023524  013737  177572  001264             MOV     SR0,WASSR0      ;READ STATUS REG. 0
2489 023532  013737  177576  001270     5$:     MOV     SR2,WASSR2      ;READ STATUS REG. 2
2490 023540  012700  000017                     MOV     #17,R0          ;LOAD EXPECTED SR0 CONTENTS INTO R0
2491 023544  020037  001264                     CMP     R0,WASSR0       ;SR0  ERROR BITS LEFT CLEAR BY TRAPPING?
2492 023550  001402                             BEQ     6$              ;BRANCH IF YES
2493 023552  005237  001200                     INC     $TMP1           ;SR0 ERROR BITS SET WHEN ODD ADDR. SERVICED
2494 023556  012701  023532             6$:     MOV     #5$,R1          ;LOAD EXPECTED SR2 CONTENTS INTO R1
2495 023562  020137  001270                     CMP     R1,WASSR2       ;WAS SR2 LEFT UNLOCKED BY TRAPPING?
2496 023566  001402                             BEQ     7$              ;BRANCH IF YES
2497 023570  005237  001200                     INC     $TMP1           ;SR2 LOCKED UP BY ODD ADDR. ERROR
2498 023574  005737  001200             7$:     TST     $TMP1           ;WHERE SR0 OR SR2 EFFECTED?
2499 023600  001404                             BEQ     8$              ;BRANCH IF NO
2500 023602  104021                             ERROR   +21             ;SR0 OR SR2 CHANGED BY ODD ADDR. ERROR
2501                                                                     ;FOR TIGHTER SCOPE LOOP
2502                                                                     ;REPLACE ERROR CALL WITH
2503                                                                     ;'BR 2$'' = 000741
2504 023604  042737  160000  177572             BIC     #160000,SR0     ;CLEAR ERROR BITS THAT MAY BE SET IN SR0
2505 023612  012737  015654  000004     8$:     MOV     #TIMERR,@#4     ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
2506 023620  012737  015726  000250             MOV     #MGMERR,@#250   ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
2507 023626  012737  077406  172310             MOV     #77406,KIPDR4   ;REMAP PAGE 4 TO READ/WRITE
```

```
2522                                    .SBTTL  TEST # 13 - PC & PSW SAVED FOR KT ERROR ON ODD ADDR.
                                        ;*****************************************************************
                                        ;*TEST 13       PC & PSW SAVED FOR KT ERROR ON ODD ADDR.
                                        ;*
                                        ;*      THIS TEST CHECKS THE PC AND PROCESSOR STATUS WORD SAVED WHEN
                                        ;*      A KT ERROR OCCURS DURING THE SECOND PUSH ON THE STACK DURING
                                        ;*      SERVICING OF AN ODD ADDR. ERROR.  DURING A 'DOUBLE ERROR''
                                        ;*      SEQUENCE SUCH AS THIS, THE PSW SAVED WILL BE THE ONE PICKED UP
                                        ;*      FROM VECTOR+2 (LOC. 6 IN THIS CASE) AFTER THE FIRST TRAP,
                                        ;*      NOT THE PSW PRESENT BEFORE THE FIRST TRAP.  SR0 AND SR2
                                        ;*      SHOULD RECORD THE KT ERROR (A R/O VIOLATION BY THE USER STACK PTR.)
                                        ;*
                                        ;*      NOTE THAT THE PREVIOUS MODE BITS <13:12> OF THE PSW
                                        ;*      WILL BE SET IN THE PSW THAT IS SAVED.
                                        ;*
                                        ;*****************************************************************
     023634  000004
2523 023636  004737  002346            TST13:  SCOPE
                                        1$:     JSR     PC,TOFF         ;TURN T-BIT TRAPPING OFF FOR THIS TEST
2524 023642  012737  000600  177646            MOV     #600,UIPAR3     ;MAP USER PAGE 3 TO 12-16K
2525 023650  012737  000600  177650            MOV     #600,UIPAR4     ;MAP USER PAGE 4 TO 12-16K
2526 023656  012737  077402  177606            MOV     #77402,UIPDR3   ;MAP USER PAGE 3 READ-ONLY
2527 023664  012737  077406  177610            MOV     #77406,UIPDR4   ;MAP USER PAGE 4 READ/WRITE
2528 023672  012737  023746  000004            MOV     #4$,@#4         ;LOAD ADDRESS OF 4$ IN CPU (ODD ADDR.) VECTOR
2529 023700  012737  140017  000006            MOV     #140017,@#6     ;LOAD PSW THAT SHOULD BE PUT ON STACK IN VECTOR+2
2530 023706  012737  023746  000250            MOV     #4$,@#250       ;LOAD ADDRESS OF 4$ IN M.M. TRAP VECTOR
2531 023714  012737  000340  000252            MOV     #340,@#252      ;LOAD A KERNEL PSW IN MMVEC+2
2532 023722  012737  023730  001110            MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2533 023730  012737  140000  177776  2$:       MOV     #140000,PSW     ;GO TO USER MODE
2534 023736  012706  100002                    MOV     #100002,USP     ;SET USER STACK PTR. SO SECOND PUSH IS IN PG. 3
2535 023742  005237  100005            3$:     INC     100005          ;CAUSE ODD ADDRESS ERROR THAT WILL CAUSE
2536                                                                    ;R/O ERROR WHEN TRY TO SAVE OLD PC
2537 023746  016601  000002            4$:     MOV     2(KSP),R1       ;PUT PSW SAVED ON KERNEL STACK INTO R1
2538 023752  011603                            MOV     (KSP),R3        ;PUT PC SAVED ON KERNEL STACK INTO R3
2539 023754  013737  177572  001264            MOV     SR0,WASSR0      ;READ THE CONTENTS OF M.M. STATUS REG. 0
2540 023762  013737  177576  001270            MOV     SR2,WASSR2      ;READ THE CONTENTS OF M.M. STATUS REG. 2
2541 023770  042737  160000  177572            BIC     #160000,SR0     ;CLEAR THE ERROR BITS IN SR0
2542 023776  005037  177776                    CLR     PSW             ;BE SURE IN KERNEL MODE
2543 024002  012706  001100                    MOV     #KERSTK,KSP     ;RESTORE KERNEL STACK POINTER
2544 024006  012737  140000  177776            MOV     #140000,PSW     ;GO TO USER MODE
2545 024014  012706  000600                    MOV     #USESTK,USP     ;RESTORE USER STACK POINTER
2546 024020  005037  177776                    CLR     PSW             ;GO BACK TO KERNEL MODE
2547 024024  005037  001176                    CLR     $TMP0           ;CLEAR ERROR INDICATOR
2548 024030  020127  170017                    CMP     R1,#170017      ;WAS THE PSW SAVED THE ONE PICKED UP BY THE
2549                                                                    ;ODD ADDR. TRAP FROM ERRVEC+2?
2550                                                                    ;VALUE 170017 = PSW FROM LOC. 6 WITH
2551                                                                    ;PREVIOUS MODE BITS = USER
2552 024034  001402                            BEQ     5$              ;BRANCH IF YES
2553 024036  005237  001176                    INC     $TMP0           ;WRONG PSW SAVED DURING 'DOUBLE ERROR'' SEQUENCE
2554 024042  020327  023746            5$:     CMP     R3,#3$+4        ;WAS THE PC AT THE TIME OF THE ODD ADDR. ERROR
2555                                                                    ;SAVED ON THE STACK?
2556 024046  001402                            BEQ     6$              ;BRANCH IF YES
2557 024050  005237  001176                    INC     $TMP0           ;WRONG PC SAVED DURING TRAP SEQUENCE
2558 024054  023727  001264  020147  6$:       CMP     WASSR0,#20147   ;DID SR0 REPORT - USER, PAGE 3, R/O ABORT?
2559 024062  001402                            BEQ     7$              ;BRANCH IF YES
2560 024064  005237  001176                    INC     $TMP0           ;SR0 DID NOT REPORT R/O ABORT
2561 024070  023727  001270  023742  7$:       CMP     WASSR2,#3$      ;DID SR2 LOCK UP VIRTUAL ADDR. OF LAST
2562                                                                    ;INSTRUCTION SUCCESSFULLY FETCHED?
```

TEST # 13 - PC & PSW SAVED FOR KT ERROR ON ODD ADDR.

```
2563 024076  001402                        BEQ     8$              ;BRANCH IF YES
2564 024100  005237  001176                INC     $TMP0           ;SR2 DID NOT LOCK UP ADDR. OF ODD ADDR. INST.
2565 024104  005737  001176          8$:   TST     $TMP0           ;ANY 'ERRORS' DURING TRAP SEQUENCE?
2566 024110  001401                        BEQ     9$              ;BRANCH IF NO
2567 024112  104022                        ERROR   +22             ;THE WRONG PC OR PSW WERE SAVED
2568                                                                ;OR SR0 OR SR2 DID NOT REPORT R/O
2569                                                                ;ERROR DURING ODD ADDR. - KT TRAP
2570                                                                ;SEQUENCE
2571                                                                ;FOR TIGHTER SCOPE LOOP
2572                                                                ;REPLACE ERROR CALL WITH
2573                                                                ;'BR 2$' = 000710
2574 024114  012737  015654  000004  9$:   MOV     #TIMERR,@#4     ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
2575 024122  012737  000340  000006        MOV     #340,@#6        ;RELOAD ERRVEC+2 WITH KERNEL PSW
2576 024130  012737  015726  000250        MOV     #MGMERR,@#250   ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
2577 024136  012737  077406  177606        MOV     #77406,UIPDR3   ;REMAP USER PAGE 3 READ/WRITE
2578 024144  004737  002402                JSR     PC,TON          ;TURN T-BIT TRAPPING BACK ON
```

```
2579                                    .SBTTL  GROUP 2          D-SPACE TESTS
2580                                    ;/.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\
2581                                    ;/.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\
```

TEST # 14 - ENABLE D-SPACE AND SEE THAT I-SPACE IS FORCED

```
2597                                    .SBTTL  TEST # 14 - ENABLE D-SPACE AND SEE THAT I-SPACE IS FORCED
                                        ;*****************************************************************
                                        ;*TEST 14       ENABLE D-SPACE AND SEE THAT I-SPACE IS FORCED
                                        ;*
                                        ;*      THIS TEST SHOWS THAT I-SPACE IS FORCED DURING INSTRUCTION FETCHES,
                                        ;*      AND ADDRESS, INDEX AND OPERAND FETCHES IF THE REGISTER FIELD IS 7.
                                        ;*
                                        ;*      ALL ERRORS FOUND IN THIS TEST ARE REPORTED WHEN THE CPU
                                        ;*      ABORTS THRU 'MMVEC'' TO SUBROUTINE 'NODSPAC'. THIS SUB-
                                        ;*      ROUTINE WILL REPORT THAT D-SPACE WAS NOT ENABLED PROPERLY.
                                        ;*
                                        ;*      NOTE - WHENEVER A DSTM=3,6,7 IS SHOWN, AND THE OPERAND IS
                                        ;*             '100000', I-SPACE IS FORCED IN THE OPERAND FETCH,
                                        ;*             BUT D-SPACE WILL BE FORCED ON THE ACCESS OF THE
                                        ;*             LOCATION;  CORRECT OPERATION IS CHECKED.
                                        ;*
                                        ;*****************************************************************
     024150  000004                     TST14:  SCOPE
2598                                            ;INITIALIZE KERNAL I/D SPACE PAR'S AND PDR'S
2599 024152  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2600 024156  012700  077400                     MOV     #77400,R0       ;MAKE KERNAL D-SPACE PGS. 1,2&3 NON-RESIDENT
2601 024162  010037  172310                     MOV     R0,KIPDR4       ;AND KERNAL I-SPACE PDR4 NON-RESIDENT
2602 024166  010037  172322                     MOV     R0,KDPDR1
2603 024172  010037  172324                     MOV     R0,KDPDR2
2604 024176  010037  172326                     MOV     R0,KDPDR3
2605 024202  012737  000600  172370           MOV     #600,KDPAR4     ;MAP KDPAR4 TO 12->16K
2606 024210  012737  002222  000250           MOV     #NODSPAC,MMVEC  ;SET M.M. VECTOR TO D-SPACE SERVICE ROUTINE
2607 024216  012737  024236  001110           MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
2608 024224  005237  177572                     INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2609 024230  012737  000024  172516           MOV     #24,MMR3        ;ENABLE 22-BIT,KERNAL D-SPACE MAPPING
2610                                    ;*      TEST THAT INSTRUCTION FETCHES FORCE I-SPACE
2611                                    ;*
2612 024236  000400                     10$:    BR      4$              ;BRANCH
2613 024240  000244                     4$:     CLZ                     ;CLEAR ZERO BIT IN PSW
2614 024242  001776                             BEQ     4$              ;NO BRANCH
2615 024244  000264                     5$:     SEZ                     ;SET ZERO BIT IN PSW
2616 024246  001376                             BNE     5$              ;NO BRANCH
2617 024250  000270                     6$:     SEN                     ;SET NEGATIVE BIT IN PSW
2618 024252  100376                             BPL     6$              ;NO BRANCH
2619                                    ;*      TRY SOME SOP INSTRUCTIONS WITH SRCM=DSTM=0
2620                                    ;*      THESE SHOULD NEVER INVOKE D-SPACE
2621                                    ;*
2622 024254  005700                             TST     R0
2623 024256  005200                             INC     R0
2624 024260  005300                             DEC     R0
2625 024262  006000                             ROR     R0
2626 024264  006100                             ROL     R0
2627 024266  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2628                                    ;*      TEST SOB NON-MOD WITH DSTM=2,3,6,7; DSTF-7
2629                                    ;*
2630 024272  012737  100002  060000           MOV     #^100002,@#60000 ;SET UP TEST LOCATION
2631 024300  012737  024312  001110           MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
2632 024306  005237  177572                     INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2633 024312  005727  060000             11$:    TST     #60000          ;DSTM=2 SOP NON-MOD
2634 024316  005737  100000                     TST     @#100000        ;DSTM=3 SOP NON-MOD
2635 024322  005737  100000                     TST     100000          ;DSTM=6 SOP NON-MOD
2636 024326  005777  053446                     TST     @100000         ;DSTM=7 SOP NON-MOD
```

```
TEST # 14 - ENABLE D-SPACE AND SEE THAT I-SPACE IS FORCED

                                        CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
                                ;*      TEST SOB MOD WITH DSTM=2,3,6,7; DSTF=7
                                ;*
                                        MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
                                        MOV     #100000,@#60002 ;SET UP TEST VALUES
                                        INC     MMR0            ;TURN ON MEMORY MANAGEMENT
                                12$:    CLR     #60000          ;DSTM=2 SOP MOD
                                        CLR     @#100000        ;DSTM=3 SOP MOD
                                        CLR     100000          ;DSTM=6 SOP MOD
                                        CLR     @100002         ;DSTM-7 SOP MOD
                                        CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
                                ;*
                                ;*      THE NEXT THREE TESTS ARE CONCERNED WITH TESTING
                                ;*      DOP AND NOT(SRCM-DSTM=0)
                                ;*
                                ;*      TEST DOP DEST NON-MOD SRCM=DSTM=2,3,6,7; SRCF=DSTF=7
                                ;*
                                        MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
                                        MOV     #100000,@#60002 ;SET UP TEST VALUE
                                        INC     MMR0            ;TURN ON MEMORY MANAGEMENT
                                13$:    CMP     #60000,#60000   ;SRCM=2 DSTM 2 DOP NON-MOD
                                        CMP     @#100000,@#100000 ;SRCM=3 DSTM=3 DOP NON-MOD
                                        CMP     100000,100000   ;SRCM=6 DSTM=6 DOP NON-MOD
                                        CMP     @100002,@100002 ;SRCM=7 DSTM=7 DOP NON-MOD
                                        CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
                                ;*      TEST MOV DEST AND NOT(SRCM=DSTM=0)
                                ;*      SRCM DSTM-2,3,6,7; SRCF=DSTF=7
                                ;*
                                        MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
                                        MOV     #100000,@#60002 ;SET UP TEST VALUE
                                        INC     MMR0            ;TURN ON MEMORY MANAGEMENT
                                14$:    MOV     #60002,#60002   ;SRCM=2 DSTM=2 MOV
                                        MOV     #60000,@#100000 ;SRCM=2 DSTM=3 MOV
                                        MOV     #60000,100000   ;SRCM=2 DSTM=6 MOV
                                        MOV     #60000,@100002  ;SRCM=2 DSTM=7 MOV
                                        CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
                                ;*      TEST DOP DEST MOD AND NOT SUB
                                ;*      SRCM=DSTM=2,3,6,7; SRCF=DSTF=7
                                ;*
                                        MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
                                        INC     MMR0            ;TURN ON MEMORY MANAGEMENT
                                15$:    BIS     #60000,#60000   ;SRCM=2 DSTM=2 DOP MOD
                                        BIS     #00000,@#100000 ;SRCM=2 DSTM=3 DOP MOD
                                        BIS     #00000,100000   ;SRCM=2 DSTM=6 DOP MOD
                                        BIS     #00000,@100002  ;SRCM=2 DSTM=7 DOP MOD
                                        CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
                                ;*      TEST SWAB WITH DSTM=2,3,6,7; DSTF=7
                                ;*
                                        MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
                                        MOV     #100002,@#60000 ;SET UP TEST VALUES
                                        INC     MMR0            ;TURN ON MEMORY MANAGEMENT
                                16$:    SWAB    #60000          ;DSTM=2 SWAB
                                        SWAB    @#100002        ;DSTM=3 SWAB
                                        SWAB    100002          ;DSTM=6 SWAB
                                        SWAB    @100000         ;DSTM=7 SWAB
                                        CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
                                ;*      TEST ROT/SHFT WITH DSTM=2,3,6,7; DSTF=7
```

```
TEST # 14 - ENABLE D-SPACE AND SEE THAT I-SPACE IS FORCED

2694                                        ;*
2695 024644  012737  024664  001110        MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
2696 024652  012737  100000  060002        MOV     #100000,@#60002 ;SET UP TEST VALUES
2697 024660  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2698 024664  006127  060000        17$:    ROL     #60000          ;DSTM=2 ROT/SHFT
2699 024670  006137  100000                ROL     @#100000        ;DSTM=3 ROT/SHFT
2700 024674  006137  100000                ROL     100000          ;DSTM=6 ROT/SHFT
2701 024700  006177  053076                ROL     @100002         ;DSTM=7 ROT/SHFT
2702 024704  005037  177572                CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2703                                        ;*      TEST ASH/ASHC WITH DSTM=2,3,6,7; DSTF=7
2704                                        ;*
2705 024710  012737  024736  001110        MOV     #18$,$LPERR     ;SET LOOP ON ERROR POINTER TO 18$
2706 024716  012737  000001  060000        MOV     #1,@#60000      ;SET UP TEST VALUES
2707 024724  012737  100000  060002        MOV     #100000,@#60002
2708 024732  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2709 024736  072027  000001        18$:    ASH     #1,R0           ;DSTM=2 ASH/ASHC
2710 024742  072037  100000                ASH     @#100000,R0     ;DSTM=3 ASH/ASHC
2711 024746  072037  100000                ASH     100000,R0       ;DSTM=6 ASH/ASHC
2712 024752  072077  053024                ASH     @100002,R0      ;DSTM=7 ASH/ASHC
2713 024756  005037  177572                CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2714                                        ;*      TEST MUL/DIV WITH DSTM=2,3,6,7; DSTF=7
2715                                        ;*
2716 024762  012737  024774  001110        MOV     #19$,$LPERR     ;SET LOOP ON ERROR POINTER TO 19$
2717 024770  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2718 024774  070027  000002        19$:    MUL     #2,R0           ;DSTM=2 MUL/DIV
2719 025000  070037  100000                MUL     @#100000,R0     ;DSTM=3 MUL/DIV
2720 025004  070037  100000                MUL     100000,R0       ;DSTM=6 MUL/DIV
2721 025010  070077  052766                MUL     @100002,R0      ;DSTM=7 MUL/DIV
2722 025014  005037  177572                CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2723                                        ;*      TEST JMP WITH DSTM=3,6,7; DSTF=7
2724                                        ;*
2725 025020  012737  025040  001110        MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
2726 025026  012737  025054  060000        MOV     #23$,@#60000    ;SET UP TEST VALUES
2727 025034  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2728 025040  000137  025044        20$:    JMP     @#21$           ;DSTM=3 JMP
2729 025044  000137  025050        21$:    JMP     22$             ;DSTM=6 JMP
2730 025050  000177  052724        22$:    JMP     @100000         ;DSTM=7 JMP
2731 025054  005037  177572        23$:    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2732                                        ;*      TEST SUB WITH DSTM=2,3,6,7; DSTF=7
2733                                        ;*
2734 025060  012737  025102  001110        MOV     #28$,$LPERR     ;SET LOOP ON ERROR POINTER TO 28$
2735 025066  005000                        CLR     R0              ;SET UP TEST VALUES
2736 025070  012737  100000  060002        MOV     #100000,@#60002
2737 025076  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2738 025102  160027  060002        28$:    SUB     R0,#60002       ;DSTM=2 SUB
2739 025106  160037  100000                SUB     R0,@#100000     ;DSTM=3 SUB
2740 025112  160037  100000                SUB     R0,100000       ;DSTM=6 SUB
2741 025116  160077  052660                SUB     R0,@100002      ;DSTM=7 SUB
2742 025122  005037  177572                CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
```

```
                                .SBTTL  TEST # 15 - ENABLE D-SPACE AND SEE I-SPACE IS NOT FORCED
                                ;*****************************************************************
                                ;*TEST 15       ENABLE D-SPACE AND SEE I-SPACE IS NOT FORCED
                                ;*
                                ;*      THIS TEST SHOWS THAT I-SPACE IS NOT FORCED IF THE REGISTER FIELD
                                ;*      IS NOT 7, BUT THE OTHER CONDITIONS ARE MET.
                                ;*
                                ;*      ALL ERRORS FOUND IN THIS TEST ARE REPORTED WHEN THE CPU ABORTS
                                ;*      THROUGH 'MMVEC' TO SUBROUTINE 'NODSPAC'.  THIS SUBROUTINE WILL
                                ;*      REPORT THAT D-SPACE WAS NOT ENABLED PROPERLY.
                                ;*
                                ;*****************************************************************
 025126  000004                 TST15:  SCOPE
2754 025130  012700  077406             MOV     #77406,R0
2755 025134  010037  172310             MOV     R0,KIPDR4       ;MAKE KIPDR4 R/W,4K,200 BLOCKS
2756 025140  010037  172322             MOV     R0,KDPDR1       ;MAKE KDPDR1 R/W,4K,200 BLOCKS
2757 025144  010037  172324             MOV     R0,KDPDR2       ;MAKE KDPDR2 R/W,4K,200 BLOCKS
2758 025150  010037  172326             MOV     R0,KDPDR3       ;MAKE KDPDR3 R/W,4K,200 BLOCKS
2759 025154  105037  172306             CLRB    KIPDR3          ;MAKE KIPDR3 NON-RESIDENT
2760                            ;*      TEST SOP NON-MOD; DSTM=1,2,3,5,7
2761                            ;*
2762 025160  012700  060000     20$:    MOV     #60000,R0       ;SET UP CONSTANTS FOR TEST
2763 025164  010037  060000             MOV     R0,@#60000
2764 025170  012737  060002  060002     MOV     #60002,@#60002
2765 025176  012737  060004  060004     MOV     #60004,@#60004
2766 025204  012737  060006  060006     MOV     #60006,@#60006
2767 025212  012737  025254  001110     MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
2768 025220  012737  060000  060000     MOV     #60000,@#60000
2769 025226  012737  060002  060002     MOV     #60002,@#60002
2770 025234  012737  060004  060004     MOV     #60004,@#60004
2771 025242  012737  060006  060006     MOV     #60006,@#60006
2772 025250  005237  177572             INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2773 025254  005710             1$:    TST     (R0)            ;DSTM=1 SOP NON-MOD
2774 025256  005720                     TST     (R0)+           ;DSTM=2 SOP NON-MOD
2775 025260  005730                     TST     @(R0)+          ;DSTM=3 SOP NON-MOD
2776 025262  005750                     TST     @-(R0)          ;DSTM=5 SOP NON-MOD
2777 025264  005770  000000             TST     @0(R0)          ;DSTM=7 SOP NON-MOD
2778 025270  005037  177572             CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2779                            ;*      TEST SOP MOD; DSTM=1,2,3,5,7
2780                            ;*
2781 025274  012737  025306  001110     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2782 025302  005237  177572             INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2783 025306  005010             2$:    CLR     (R0)            ;DSTM=1 SOP MOD
2784 025310  005020                     CLR     (R0)+           ;DSTM=2 SOP MOD
2785 025312  005030                     CLR     @(R0)+          ;DSTM=3 SOP MOD
2786 025314  005050                     CLR     @-(R0)          ;DSTM=5 SOP MOD
2787 025316  005070  000000             CLR     @0(R0)          ;DSTM=7 SOP MOD
2788 025322  005037  177572             CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2789                            ;*      TEST DOP DEST NON-MOD WITH SRCM=1,2,3,5,7 AND DSTM=1,2,3,5,7
2790                            ;*      ALL SOURCE MODES TO BE TESTED ARE TESTED HERE
2791                            ;*
2792 025326  012737  025370  001110     MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2793 025334  012702  000032             MOV     #32,R2          ;SET UP ADDRESSES 60000-60064 FOR TEST
2794 025340  012700  060000             MOV     #60000,R0
2795 025344  012701  060000             MOV     #60000,R1
2796 025350  010021             21$:    MOV     R0,(R1)+
2797 025352  062700  000002             ADD     #2,R0
```

```
2798 025356  077204                          SOB     R2,21$
2799 025360  012700  060000                  MOV     #60000,R0
2800 025364  005237  177572                  INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2801 025370  021010                  3$:     CMP     (R0),(R0)       ;SRCM=1 DSTM=1 DOP DEST NON-MOD
2802 025372  021020                          CMP     (R0),(R0)+      ;SRCM=1 DSTM=2 DOP DEST NON-MOD
2803 025374  021030                          CMP     (R0),@(R0)+     ;SRCM=1 DSTM=3 DOP DEST NON-MOD
2804 025376  021050                          CMP     (R0),@-(R0)     ;SRCM=1 DSTM=5 DOP DEST NON-MOD
2805 025400  021070  000000                  CMP     (R0),@0(R0)     ;SRCM=1 DSTM=7 DOP DEST NON-MOD
2806 025404  022010                          CMP     (R0)+,(R0)      ;SRCM=2 DSTM=1 DOP DEST NON-MOD
2807 025406  022020                          CMP     (R0)+,(R0)+     ;SRCM=2 DSTM=2 DOP DEST NON-MOD
2808 025410  022030                          CMP     (R0)+,@(R0)+    ;SRCM=2 DSTM=3 DOP DEST NON-MOD
2809 025412  022050                          CMP     (R0)+,@-(R0)    ;SRCM=2 DSTM-5 DOP DEST NON-MOD
2810 025414  022070  000000                  CMP     (R0)+,@0(R0)    ;SRCM=2 DSTM=7 DOP DEST NON-MOD
2811 025420  023010                          CMP     @(R0)+,(R0)     ;SRCM=3 DSTM=1 DOP DEST NON-MOD
2812 025422  023020                          CMP     @(R0)+,(R0)+    ;SRCM=3 DSTM=2 DOP DEST NON-MOD
2813 025424  023030                          CMP     @(R0)+,@(R0)+   ;SRCM=3 DSTM=3 DOP DEST NON-MOD
2814 025426  023050                          CMP     @(R0)+,@-(R0)   ;SRCM=3 DSTM=5 DOP DEST NON-MOD
2815 025430  023070  000000                  CMP     @(R0)+,@0(R0)   ;SRCM=3 DSTM=7 DOP DEST NON-MOD
2816 025434  025010                          CMP     @-(R0),(R0)     ;SRCM=5 DSTM=1 DOP DEST NON-MOD
2817 025436  025020                          CMP     @-(R0),(R0)+    ;SRCM=5 DSTM=2 DOP DEST NON-MOD
2818 025440  025030                          CMP     @-(R0),@(R0)+   ;SRCM=5 DSTM=3 DOP DEST NON-MOD
2819 025442  025050                          CMP     @-(R0),@-(R0)   ;SRCM=5 DSTM=5 DOP DEST NON-MOD
2820 025444  025070  000000                  CMP     @-(R0),@0(R0)   ;SRCM=5 DSTM=7 DOP DEST NON-MOD
2821 025450  027010  000000                  CMP     @0(R0),(R0)     ;SRCM=7 DSTM=1 DOP DEST NON-MOD
2822 025454  027020  000000                  CMP     @0(R0),(R0)+    ;SRCM=7 DSTM=2 DOP DEST NON-MOD
2823 025460  027030  000000                  CMP     @0(R0),@(R0)+   ;SRCM=7 DSTM=3 DOP DEST NON-MOD
2824 025464  027050  000000                  CMP     @0(R0),@-(R0)   ;SRCM=7 DSTM=5 DOP DEST NON-MOD
2825 025470  027070  000000  000000          CMP     @0(R0),@0(R0)   ;SRCM=7 DSTM=7 DOP DEST NON-MOD
2826 025476  005037  177572                  CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2827                                 ;*      TEST DOP DEST MOD AND NOT SUB; DSTM=1,2,3,5,7
2828                                 ;*
2829 025502  005000                          CLR     R0              ;SET UP CONSTANTS FOR TEST
2830 025504  012701  060000                  MOV     #60000,R1
2831 025510  012737  025522  001110          MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
2832 025516  005237  177572                  INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2833 025522  050011                  4$:     BIS     R0,(R1)         ;DSTM=1 DOP DEST MOD
2834 025524  050021                          BIS     R0,(R1)+        ;DSTM=2 DOP DEST MOD
2835 025526  050031                          BIS     R0,@(R1)+       ;DSTM=3 DOP DEST MOD
2836 025530  050051                          BIS     R0,@-(R1)       ;DSTM=5 DOP DEST MOD
2837 025532  050071  000000                  BIS     R0,@0(R1)       ;DSTM=7 DOP DEST MOD
2838 025536  005037  177572                  CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2839                                 ;*      TEST MOV DEST AND NOT(SM0 AND DM0); DSTM=3,5,7
2840                                 ;*
2841 025542  012701  060000                  MOV     #60000,R1       ;SET UP CONSTANTS FOR TEST
2842 025546  012737  060002  060000          MOV     #60002,@#60000
2843 025554  012737  025566  001110          MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
2844 025562  005237  177572                  INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2845 025566  010031                  5$:     MOV     R0,@(R1)+       ;DSTM=3 MOV DEST
2846 025570  010051                          MOV     R0,@-(R1)       ;DSTM=5 MOV DEST
2847 025572  010071  000000                  MOV     R0,@0(R1)       ;DSTM=7 MOV DEST
2848 025576  005037  177572                  CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2849                                 ;*      TEST SWAB; DSTM=1,2,3,5,7
2850                                 ;*
2851 025602  012701  060000                  MOV     #60000,R1       ;SET UP CONSTANTS FOR TEST
2852 025606  012737  025620  001110          MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
2853 025614  005237  177572                  INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2854 025620  000311                  6$:     SWAB    (R1)            ;DSTM=1 SWAB
```

```
TEST # 15 - ENABLE D-SPACE AND SEE I-SPACE IS NOT FORCED

2855 025622  000321                        SWAB    (R1)+           ;DSTM=2 SWAB
2856 025624  000331                        SWAB    @(R1)+          ;DSTM=3 SWAB
2857 025626  000351                        SWAB    @-(R1)          ;DSTM=5 SWAB
2858 025630  000371  000000                SWAB    @0(R1)          ;DSTM=7 SWAB
2859 025634  005037  177572                CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2860                               ;*      TEST ROT/SHFT; DSTM=1,2,3,5,7
2861                               ;*
2862 025640  012701  060000                MOV     #60000,R1       ;SET UP CONSTANTS FOR TEST
2863 025644  012702  060006                MOV     #60006,R2
2864 025650  010203                        MOV     R2,R3
2865 025652  012737  060000  060000        MOV     #60000,@#60000
2866 025660  012737  060002  060002        MOV     #60002,@#60002
2867 025666  012737  060004  060004        MOV     #60004,@#60004
2868 025674  012737  060006  060006        MOV     #60006,@#60006
2869 025702  012737  025714  001110        MOV     #7$,$LPERR      ;SET LOOP ON ERROR POINTER TO 7$
2870 025710  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2871 025714  006111                7$:     ROL     (R1)            ;DSTM=1 ROT/SHFT
2872 025716  006121                        ROL     (R1)+           ;DSTM=2 ROT/SHFT
2873 025720  006131                        ROL     @(R1)+          ;DSTM=3 ROT/SHFT
2874 025722  006152                        ROL     @-(R2)          ;DSTM=5 ROT/SHFT
2875 025724  006173  000000                ROL     @0(R3)          ;DSTM=7 ROT/SHFT
2876 025730  005037  177572                CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2877                               ;*      TEST MUL/DIV; DSTM=1,2,3,5,7
2878                               ;*
2879 025734  012737  025750  001110        MOV     #8$,$LPERR      ;SET LOOP ON ERROR POINTER TO 8$
2880 025742  005003                        CLR     R3              ;SET UP CONSTANT FOR TEST
2881 025744  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2882 025750  070310                8$:     MUL     (R0),R3         ;DSTM=1 MUL/DIV
2883 025752  070320                        MUL     (R0)+,R3        ;DSTM=2 MUL/DIV
2884 025754  070330                        MUL     @(R0)+,R3       ;DSTM=3 MUL/DIV
2885 025756  070350                        MUL     @-(R0),R3       ;DSTM=5 MUL/DIV
2886 025760  070370  000000                MUL     @0(R0),R3       ;DSTM=7 MUL/DIV
2887 025764  005037  177572                CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2888                               ;*      TEST ASH/ASHC; DSTM=1,2,3,5,7
2889                               ;*
2890 025770  012737  000001  060000        MOV     #1,@#60000      ;SET UP CONSTANTS FOR THE TEST
2891 025776  012737  060000  060002        MOV     #60000,@#60002
2892 026004  012700  060000                MOV     #60000,R0
2893 026010  012737  026022  001110        MOV     #9$,$LPERR      ;SET LOOP ON ERROR POINTER TO 9$
2894 026016  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2895 026022  072310                9$:     ASH     (R0),R3         ;DSTM=1 ASH/ASHC
2896 026024  072320                        ASH     (R0)+,R3        ;DSTM=2 ASH/ASHC
2897 026026  072330                        ASH     @(R0)+,R3       ;DSTM=3 ASH/ASHC
2898 026030  072350                        ASH     @-(R0),R3       ;DSTM=5 ASH/ASHC
2899 026032  072370  000000                ASH     @0(R0),R3       ;DSTM=7 ASH/ASHC
2900 026036  005037  177572                CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2901                               ;*      TEST JMP; DSTM=3,7
2902                               ;*
2903 026042  012737  026074  001110        MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
2904 026050  012737  026076  060000        MOV     #11$,@#60000    ;SET UP CONSTANTS FOR THE TEST
2905 026056  012737  026102  060002        MOV     #12$,@#60002
2906 026064  012701  060000                MOV     #60000,R1
2907 026070  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2908 026074  000131                10$:    JMP     @(R1)+          ;DSTM=3 JMP
2909 026076  000171  000000        11$:    JMP     @0(R1)          ;DSTM=7 JMP
2910 026102  005037  177572        12$:    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2911                               ;*      TEST JSR; DSTM=3,7
```

```
2912                                            ;*
2913 026106  012737  026140  001110             MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
2914 026114  012737  026142  060000             MOV     #14$,@#60000    ;SET UP CONSTANTS FOR THE TEST
2915 026122  012737  026146  060002             MOV     #15$,@#60002
2916 026130  012701  060000                      MOV     #60000,R1
2917 026134  005237  177572  - - - -            INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2918 026140  004731                    13$:     JSR     PC,@(R1)+       ;DSTM=3 JSR
2919 026142  004771  000000            14$:     JSR     PC,@0(R1)       ;DSTM=7 JSR
2920 026146  005037  177572            15$:     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2921 026152  112737  000006  172306             MOVB    #6,KIPDR3       ;MAKE KIPDR3 RESIDENT
2922 026160  012706  001100                      MOV     #KERSTK,KSP     ;RESET STACK POINTER
```

```
                                        .SBTTL  TEST # 16 - PROPER ENABLING OF SUPER. D-SPACE
                                        ;;*****************************************************************
                                        ;*TEST 16       PROPER ENABLING OF SUPER. D-SPACE
                                        ;*
                                        ;*      THIS TEST CHECKS FOR PROPER ENABLING OF THE SUPERVISOR D-SPACE.
                                        ;*
                                        ;*      ANY ERRORS ENCOUNTERED WILL BE REPORTED THROUGH 'MMVEC' TO
                                        ;*      SUBROUTINE 'NODSPAC'.
                                        ;*
                                        ;*****************************************************************
 026164  000004                        TST16:  SCOPE
2932 026166  105037  172222            20$:    CLRB    SDPDR1          ;MAKE SDPDR1 NON-RESIDENT
2933 026172  012737  026220  001110            MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
2934  26200  012737  000022  172516            MOV     #22,MMR3        ;ENABLE 22-BIT SUPERVISOR D-SPACE
2935 026206  052737  040000  177776            BIS     #40000,PSW      ;ENABLE SUPERVISOR MODE
2936 026214  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2937                                    ;*      THE NEXT INSTRUCTIONS SHOULD NEVER INVOKE D-SPACE
2938                                    ;*
2939 026220  000400                    1$:     BR      2$
2940 026222  005700                    2$:     TST     R0
2941 026224  005200                            INC     R0
2942 026226  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2943 026232  112737  000006  172222            MOVB    #6,SDPDR1       ;MAKE SDPDR1 RESIDENT
2944 026240  105037  172206                    CLRB    SIPDR3          ;MAKE SIPDR3 NON-RESIDENT
2945                                    ;*      TEST SOP INSTRUCTIONS
2946                                    ;*
2947 026244  012737  026256  001110            MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2948 026252  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2949 026256  005737  060000            3$:     TST     @#60000         ;DSTM=3 DSTF=7 SOP NON-MOD
2950 026262  005037  060000                    CLR     @#60000         ;DSTM=3 DSTF=7 SOP MOD
2951 026266  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2952                                    ;*      TEST DOP INSTUCTIONS
2953                                    ;*
2954 026272  012737  026310  001110            MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
2955 026300  012700  060000                    MOV     #60000,R0       ;SET UP CONSTANT FOR TEST
2956 026304  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2957 026310  023710  060000            4$:     CMP     @#60000,(R0)    ;SRCM=3 DSTM=1 DOP DEST NON-MOD
2958 026314  052730  000000                    BIS     #0,@(R0)+       ;SRCM=2 DSTM-3 DOP DEST MOD
2959 026320  013737  060002  060002            MOV     @#60002,@#60002 ;SRCM=3 DSTM=3 MOV DEST
2960 026326  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2961                                    ;*      TEST ROT/SHFT AND ASH/ASHC INSTRUCTIONS
2962                                    ;*
2963 026332  012737  026356  001110            MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
2964 026340  012737  000001  060000            MOV     #1,@#60000      ;SET CONSTANT FOR TEST
2965 026346  012700  000001                    MOV     #1,R0           ;SET UP CONSTANT FOR TEST
2966 026352  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2967 026356  006137  060000            5$:     ROL     @#60000         ;DSTM=3 ROT/SHFT
2968 026362  072337  060000                    ASH     @#60000,R3      ;DSTM=3 ASH/ASHC
2969 026366  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2970                                    ;*      TEST MUL/DIV AND SWAB INSTRUCTIONS
2971                                    ;*
2972 026372  012737  026404  001110            MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
2973 026400  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2974 026404  070037  060000            6$:     MUL     @#60000,R0      ;DSTM=3 MUL/DIV
2975 026410  000337  060004                    SWAB    @#60004         ;DSTM=3 SWAB
2976 026414  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2977 026420  042737  000002  172516            BIC     #2,MMR3         ;DISABLE SUPERVISOR D-SPACE
```

TEST # 16 - PROPER ENABLING OF SUPER. D-SPACE

```
2978 026426  112737  000006  172206          MOVB    #6,SIPDR3       ;MAKE SIPDR3 RESIDENT
```

```
                                                .SBTTL  TEST # 17 - PROPER ENABLING OF USER D-SPACE
                                                ;*****************************************************************
                                                ;*TEST 17       PROPER ENABLING OF USER D-SPACE
                                                ;*
                                                ;*      THIS TEST CHECKS FOR PROPER ENABLING OF THE USER D-SPACE.
                                                ;*
                                                ;*      ANY ERRORS ENCOUNTERED WILL BE REPORTED THROUGH 'MMVEC' TO
                                                ;*      SUBROUTINE 'NODSPAC'.
                                                ;*
                                                ;*****************************************************************
 026434  000004                                TST17:  SCOPE
2988 026436  105037  177622                    20$:    CLRB    UDPDR1          ;MAKE UDPDR1 NON-RESIDENT
2989 026442  012737  026470  001110                    MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
2990 026450  012737  000021  172516                    MOV     #21,MMR3        ;ENABLE 22-BIT USER D-SPACE
2991 026456  052737  140000  177776                    BIS     #140000,PSW     ;ENABLE USER MODE
2992 026464  005237  177572                            INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2993                                            ;*      THE NEXT INSTRUCTIONS SHOULD NEVER INVOKE D-SPACE
2994                                            ;*
2995 026470  000400                            1$:     BR      2$
2996 026472  005700                            2$:     TST     R0
2997 026474  005200                                    INC     R0
2998 026476  005037  177572                            CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2999 026502  112737  000006  177622                    MOVB    #6,UDPDR1       ;MAKE UDPDR1 RESIDENT
3000 026510  105037  177606                            CLRB    UIPDR3          ;MAKE UIPDR3 NON-RESIDENT
3001                                            ;*      TEST SOP INSTRUCTIONS
3002                                            ;*
3003 026514  012737  026526  001110                    MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
3004 026522  005237  177572                            INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3005 026526  005737  060000                    3$:     TST     @#60000         ;DSTM=3 DSTF=7 SOP NON-MOD
3006 026532  005037  060000                            CLR     @#60000         ;DSTM=3 DSTF=7 SOP MOD
3007 026536  005037  177572                            CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3008                                            ;*      TEST DOP INSTUCTIONS
3009                                            ;*
3010 026542  012737  026560  001110                    MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
3011 026550  012700  060000                            MOV     #60000,R0       ;SET UP CONSTANT FOR TEST
3012 026554  005237  177572                            INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3013 026560  023710  060000                    4$:     CMP     @#60000,(R0)    ;SRCM=3 DSTM=1 DOP DEST NON-MOD
3014 026564  052730  000000                            BIS     #0,@(R0)+       ;SRCM=2 DSTM=3 DOP DEST MOD
3015 026570  013737  060002  060002                    MOV     @#60002,@#60002 ;SRCM=3 DSTM=3 MOV DEST
3016 026576  005037  177572                            CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3017                                            ;*      TEST ROT/SHFT AND ASH/ASHC INSTRUCTIONS
3018                                            ;*
3019 026602  012737  026622  001110                    MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
3020 026610  012737  000001  060000                    MOV     #1,@#60000      ;SET CONSTANT FOR TEST
3021 026616  005237  177572                            INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3022 026622  006137  060000                    5$:     ROL     @#60000         ;DSTM=3 ROT/SHFT
3023 026626  072337  060000                            ASH     @#60000,R3      ;DSTM=3 ASH/ASHC
3024 026632  005037  177572                            CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3025                                            ;*      TEST MUL/DIV AND SWAB INSTRUCTIONS
3026                                            ;*
3027 026636  012737  026654  001110                    MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
3028 026644  012700  000001                            MOV     #1,R0           ;SET UP CONSTANT FOR TEST
3029 026650  005237  177572                            INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3030 026654  070037  060000                    6$:     MUL     @#60000,R0      ;DSTM=3 MUL/DIV
3031 026660  000337  060004                            SWAB    @#60004         ;DSTM=3 SWAB
3032 026664  005037  177572                            CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3033 026670  042737  000001  172516                    BIC     #1,MMR3         ;DISABLE USER D-SPACE
```

TEST # 17 - PROPER ENABLING OF USER D-SPACE

```
3034 026676  112737  000006  177606          MOVB    #6,UIPDR3       ;MAKE UIPDR3 RESIDENT
3035 026704  005037  177776                  CLR     PSW             ;RESET TO KERNAL SPACE
```

```
3044                                    .SBTTL  TEST # 20 - TRAPPING IN D-SPACE KERNAL MODE
                                        ;*****************************************************************
                                        ;*TEST 20       TRAPPING IN D-SPACE KERNAL MODE
                                        ;*
                                        ;*      THIS TEST VERIFIES THAT THE ABORT VECTOR IS TAKEN FROM
                                        ;*      D-SPACE AND NOT I-SPACE. THE I-SPACE VECTOR POINTS TO
                                        ;*      10$ AND THE D-SPACE VECTOR POINTS TO 15$. EACH PSW IN
                                        ;*      VIRTUAL 252 IS DIFFERENT SO THE PROGRAM CAN TELL WHICH
                                        ;*      AREA IT IS PICKED UP FROM.
                                        ;*****************************************************************
     026710  000004                     TST20:  SCOPE
3045 026712  004737  002346                     JSR     PC,TOFF         ;TURN OF T-BIT FOR THIS TEST
3046 026716  012737  026752  001110     20$:    MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
3047 026724  012737  027020  000250             MOV     #10$,@#MMVEC    ;SET M.M. VEC. TO HOLD BAD VECTOR
3048 026732  005037  000252                     CLR     @#MMVEC+2       ;PSW IN 252 HAS PRIORITY OF ZERO
3049 026736  012737  027026  000350             MOV     #15$,@#350      ;SET D-SPACE M.M VECTOR TO 350
3050 026744  012737  000340  000352             MOV     #340,@#352      ;SET PSW PRIORITY TO 7
3051 026752  012706  001000             1$:     MOV     #1000,KSP       ;SET UP KERNAL VECTOR
3052 026756  005237  177572                     INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3053                                    ;NOW SET UP FOR AN ABORT IN KERNAL MODE WITH D-SPACE ENABLED
3054                                    ;
3055 026762  012737  077402  172330             MOV     #77402,KDPDR4   ;KERNAL D-SPACE PAGE 4 IS READ ONLY
3056 026770  012737  000600  172370             MOV     #600,KDPAR4     ;MAP D-SPACE PAGE 4 TO 12K
3057 026776  052737  000004  172516             BIS     #BIT2,MMR3      ;ENABLE KERNAL D-SPACE MAPPING
3058 027004  012737  000001  172360             MOV     #1,KDPAR0       ;MAP KERNAL D PAGE 0 TO 000100
3059 027012  012737  177777  100000             MOV     #-1,@#100000    ;TRY TO WRITE TO PAGE 4
3060 027020  013700  177776             10$:    MOV     PSW,R0          ;SAVE PSW FOR COMPARE
3061 027024  000402                             BR      16$             ;BRANCH TO D-SPACE READ CODE
3062 027026  013700  177776             15$:    MOV     PSW,R0          ;SAVE PSW FOR COMPARE
3063 027032  005037  172360             16$:    CLR     KDPAR0          ;REMAP KERNAL D PAGE 0 TO PHYSICAL 0
3064 027036  012706  001100                     MOV     #KERSTK,KSP     ;RESET STACK POINTER AFTER D-SPACE ABORT
3065 027042  042737  000004  172516             BIC     #BIT2,MMR3      ;TURN OFF KERNAL D-SPACE ENABLE
3066 027050  013701  177572                     MOV     MMR0,R1         ;SAVE MMR0 FOR COMPARE
3067 027054  013702  177574                     MOV     MMR1,R2         ;SAVE MMR1 FOR COMPARE
3068 027060  013703  177576                     MOV     MMR2,R3         ;SAVE MMR2 FOR COMPARE
3069 027064  122700  000340                     CMPB    #340,R0         ;DID YOU PICK CORRECT PSW
3070 027070  001401                             BEQ     2$              ;BRANCH IF PSW IS 340
3071 027072  104033                             ERROR   +33             ;WRONG PSW PICKED IN ABORT SEQUENCE
3072 027074  022701  020031             2$:     CMP     #020031,R1      ;EXPECTING READ ONLY ABORT
3073                                                                     ;KERNAL MODE D-SPACE PAGE 4
3074 027100  001401                             BEQ     3$              ;BRANCH IF CONDITION IS CORRECT
3075 027102  104027                             ERROR   +27             ;WRONG M.M. ABORT CONDITION
3076 027104  005037  177572             3$:     CLR     MMR0            ;CLEAR OFF MMR0 FOR EXIT OF TEST
3077 027110  012737  015726  000250             MOV     #MGMERR,MMVEC   ;RESTORE NORMAL M.M. TRAP VECTOR
3078 027116  012737  000340  000252             MOV     #340,MMVEC+2    ;RESTORE TRAP PSW (PRIORITY 7)
3079
```

```
3081                                    .SBTTL  GROUP 3  MOVE TO/FROM PREVIOUS INSTRUCTIONS TESTS
3082                                    ;/.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\
3083                                    ;;**************************************  **************************
3084                                    ;*
3085                                    ;*      THIS GROUP OF TESTS WILL TEST ALL THE LOGIC ASSOCIATED WITH
3086                                    ;*      THE 'MFPI', 'MTPI' AND 'MFPD' INSTRUCTIONS.
3087                                    ;*
3088                                    ;;*********************************************************************
3089                                    ;/.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\
```

```
3104                                    .SBTTL  TEST # 21 - MOVE FROM PREVIOUS (SUPERVISOR) I-SPACE
                                        ;*****************************************************************
                                        ;*TEST 21       MOVE FROM PREVIOUS (SUPERVISOR) I-SPACE
                                        ;*
                                        ;*      THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE PREVIOUS MODE
                                        ;*      IS CLOCKED CORRECTLY.  THERE IS A DESCRIPTION BEFORE EACH DESTINATION
                                        ;*      MODE TESTED.  THE TEST ITSELF IS CARRIED OUT IN SUBROUTINE MFPITS,
                                        ;*      WHICH USES THE MFPI INSTRUCTION CODE FOLLOWING THE JSR CALL TO EXECUTE
                                        ;*      THE TEST.  *IMPORTANT* - ALL 'NOP'S'' FOLLOWING MFPI'S OF MODES 1,2,
                                        ;*      4 AND 5 ARE TO BE LEFT ALONE.  THE SUBROUTINE LOADS THE TWO WORDS
                                        ;*      AFTER THE JSR CALL, PREPARING FOR MODES 3, 6 AND 7.
                                        ;*
                                        ;*      IF THE CORRECT MODE (SUPERVISOR) IS NOT ENABLED A NON-RESIDENT ABORT
                                        ;*      WILL OCCUR AND TRAP TO MFPIV1, WHERE THE ERRORS ARE REPORTED.
                                        ;*
                                        ;*****************************************************************
     027124  000004
3105 027126  004737  002050            TST21:  SCOPE
                                                JSR     PC,APRINIT      ;INIT ALL PAR/PDR'S
3106 027132  012737  000001  177572            MOV     #1,MMR0         ;TURN ON MEMORY MANAGEMENT
3107 027140  012737  000600  172350    1$:     MOV     #600,KIPAR4     ;MAP KIPAR4 TO 12K
3108 027146  012737  000600  172250            MOV     #600,SIPAR4     ;MAP SIPAR4 TO 12K
3109 027154  012700  036514                    MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
3110 027160  010037  100000                    MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 60000
3111 027164  105037  172310                    CLRB    KIPDR4          ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
3112                                    ;THE FOLLOWING WILL TEST DSTM=0 MFPI
3113 027170  012737  027176  001110            MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3114 027176  012737  010340  177776    2$:     MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
3115 027204  006506                            MFPI    SSP             ;PUT SUPERVISOR STACK POINTER ON KERNEL STACK
3116 027206  022706  001100                    CMP     #KERSTK,KSP     ;WAS SOMETHING PUSHED ON STACK AT 6$
3117 027212  001407                            BEQ     3$              ;BRANCH IF NOTHING WAS PUSHED
3118 027214  012600                            MOV     (KSP)+,R0       ;POP KERNEL STACK INTO R0
3119 027216  012701  000700                    MOV     #SUPSTK,R1      ;EXPECTING TO GET 700 AS SSP
3120 027222  020001                            CMP     R0,R1           ;DID YOU GET THE RIGHT POINTER?
3121 027224  001403                            BEQ     4$              ;BRANCH IF YOU DID
3122 027226  104023                            ERROR   +23             ;WRONG THING WAS PUSHED ON STACK
3123                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 4$'' WITH 'BR 2$'' = 000764
3124 027230  000401                            BR      4$              ;BRANCH TO NEXT TRY
3125 027232  104025                    3$:     ERROR   +25             ;NOTHING PUSHED ON STACK
3126                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 3$'' ABOVE WITH 'BR 2$'' = 000771
3127
3128                                    ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3129 027234  012737  002506  001110    4$:     MOV     #MFPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MFPILP IN SUBROUTINE
3130 027242  012700  036514                    MOV     #36514,R0       ;RELOAD DATA PATTERN IN R0
3131 027246  012737  010340  002510            MOV     #010340,MFPIPS  ;MAKE PREVIOUS MODE SUPERVISOR IN SUBROUTINE
3132 027254  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3133 027260  012737  027460  002612            MOV     #MFPIV1,MFPIVC  ;LOAD ADDRESS OF THIS TEST'S TRAP CATCHER TO MFPIVC
3134 027266  004737  002454                    JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3135 027272  006512                            MFPI    (R2)            ;<HERE - READ FROM PHYSICAL 60000
3136 027274  000240                            NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3137 027276  104023                            ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3138 027300  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3139                                    ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
3140 027302  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3141 027306  004737  002454                    JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3142 027312  006522                            MFPI    (R2)+           ;<HERE - READ FROM PHYSICAL 60000
3143 027314  000240                            NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3144 027316  104023                            ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
```

```
3145 027320  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3146                              ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
3147 027322  004737  002454                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3148 027326  006537  100000                  MFPI    @#100000        ;<HERE - READ FROM PHYSICAL 60000
3149 027332  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3150 027334  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3151                              ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3152 027336  012702  100002                  MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3153 027342  004737  002454                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3154 027346  006542                          MFPI    -(R2)           ;<HERE - READ FROM PHYSICAL 60000
3155 027350  000240                          NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3156 027352  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3157 027354  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3158                              ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3159 027356  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3160 027364  012702  001204                  MOV     #<$TMP2+2>,R2   ;LOAD ADDR. OF $TMP2+2 INTO R2
3161 027370  004737  002454                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3162 027374  006552                          MFPI    @-(R2)          ;<HERE - READ FROM PHYSICAL 60000
3163 027376  000240                          NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3164 027400  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3165 027402  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3166                              ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3167 027404  005002                          CLR     R2              ;MAKE REGISTER 2 A ZERO
3168 027406  004737  002454                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3169 027412  006562  100000                  MFPI    100000(R2)      ;<HERE - READ FROM PHYSICAL 60000
3170 027416  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3171 027420  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3172                              ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3173 027422  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD TEST LOC. V.A. INTO $TMP2
3174 027430  012702  001202                  MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3175 027434  004737  002454                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3176 027440  006572  000000                  MFPI    @0(R2)          ;USE $TMP2 TO FETCH VIRTUAL ADDRESS OF 60000
3177 027444  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3178 027446  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3179 027450  112737  000006  172310          MOVB    #6,KIPDR4       ;MAKE KIPDR4 RESIDENT
3180 027456  000423                          BR      TST22           ;;BRANCH TO NEXT TEST
```

MM TRAP CATCHER FOR ABOVE TEST

```
                                                .SBTTL  MM TRAP CATCHER FOR ABOVE TEST
027460  012637  001260            MFPIV1: MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
027464  012637  001262                    MOV     (KSP)+,TRAPPS
027470  013737  177572  001264            MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
027476  013737  177576  001270            MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
027504  042737  160000  177572            BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0 AND LEAVE
027512  104026                            ERROR   +26             ;TRIED TO READ NON-RESIDENT PAGE
                        ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI' = 000002
027514  013746  001262                    MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
027520  013746  001260                    MOV     TRAPPC,-(KSP)
027524  000002                            RTI
```

```
3207                                    .SBTTL  TEST # 22 - MOVE FROM PREVIOUS (USER) I-SPACE
                                        ;*****************************************************************
                                        ;*TEST 22       MOVE FROM PREVIOUS (USER) I-SPACE
                                        ;*
                                        ;*      THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE PREVIOUS MODE
                                        ;*      IS CLOCKED CORRECTLY.  THERE IS A DESCRIPTION BEFORE EACH DESTINATION
                                        ;*      MODE TESTED.  THE TEST ITSELF IS CARRIED OUT IN SUBROUTINE MFPITS,
                                        ;*      WHICH USES THE MFPI INSTRUCTION CODE FOLLOWING THE JSR CALL TO EXECUTE
                                        ;*      THE TEST.  *IMPORTANT* - ALL 'NOP'S' FOLLOWING MFPI'S OF MODES 1,2,
                                        ;*      4 AND 5 ARE TO BE LEFT ALONE.  THE SUBROUTINE LOADS THE TWO WORDS
                                        ;*      AFTER THE JSR CALL, PREPARING FOR MODES 3, 6 AND 7.
                                        ;*
                                        ;*      IF THE CORRECT MODE (USER) IS NOT ENABLED A NON-RESIDENT ABORT WILL
                                        ;*      OCCUR AND TRAP TO MFPIV2, WHERE THE ERRORS ARE REPORTED.
                                        ;*
                                        ;*****************************************************************
     027526  000004                     TST22:  SCOPE
3208 027530  012700  036514             1$:     MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
3209 027534  012737  000600  177650             MOV     #600,UIPAR4     ;MAP UIPAR4 TO 12K
3210 027542  010037  100000             MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 60000
3211 027546  105037  172310             CLRB    KIPDR4          ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
3212                                    ;THE FOLLOWING WILL TEST DSTM=0 MFPI
3213 027552  012737  027560  001110             MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3214 027560  012737  030340  177776  2$:     MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3215 027566  012737  030056  000250             MOV     #MFPIV2,MMVEC   ;LOAD ADDRESS OF THIS TEST'S TRAP CATCHER TO MMVEC
3216 027574  006506                     MFPI    USP             ;PUT USER STACK POINTER ON KERNEL STACK
3217 027576  012737  015726  000250             MOV     #MGMERR,MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
3218 027604  022706  001076             CMP     #KERSTK-2,KSP   ;WAS SOMETHING PUSHED ON STACK BY THE MFPI?
3219 027610  001007                     BNE     3$              ;BRANCH TO ERROR IF POINTER IS WRONG
3220 027612  012600                     MOV     (KSP)+,R0       ;POP KERNEL STACK INTO R0
3221 027614  012701  000600             MOV     #USESTK,R1      ;EXPECTING TO GET 600 AS USP
3222 027620  020001                     CMP     R0,R1           ;DID YOU GET THE RIGHT POINTER?
3223 027622  001403                     BEQ     4$              ;BRANCH IF YOU DID
3224 027624  104023                     ERROR   +23             ;WRONG THING WAS PUSHED ON STACK
3225                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 4$' WITH 'BR 2$' = 000764
3226 027626  000401                     BR      4$              ;BRANCH TO NEXT TRY
3227 027630  104025                     3$:     ERROR   +25             ;NOTHING PUSHED ON STACK
3228                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BNE 3$' ABOVE WITH 'BR 2$' = 000771
3229                                            ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3230 027632  012737  002506  001110  4$:     MOV     #MFPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MFPILP IN SUBROUTINE
3231 027640  012737  030340  002510             MOV     #030340,MFPIPS  ;MAKE PREVIOUS MODE USER IN SUBROUTINE LOCATION
3232 027646  012700  036514             MOV     #36514,R0       ;RELOAD DATA PATTERN IN R0
3233 027652  012737  030056  002612             MOV     #MFPIV2,MFPIVC  ;LOAD ADDRESS OF THIS TEST'S TRAP CATCHER TO MFPIVC
3234 027660  012702  100000             MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3235 027664  004737  002454             JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3236 027670  006512                     MFPI    (R2)            ;<HERE - READ FROM PHYSICAL 60000
3237 027672  000240                     NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3238 027674  104023                     ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3239 027676  005726                     TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3240                                    ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
3241 027700  012702  100000             MOV     #100000,R2      ;LOAD VIRTUAL ADDRFSS INTO R2
3242 027704  004737  002454             JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3243 027710  006522                     MFPI    (R2)+           ;<HERE - READ FROM PHYSICAL 60000
3244 027712  000240                     NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3245 027714  104023                     ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3246 027716  005726                     TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3247                                    ;THE FOLLOWING WILL TEST DSTM-3 MFPI.
```

```
3248 027720  004737  002454                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3249 027724  006537  100000                  MFPI    @#100000        ;<HERE - READ FROM PHYSICAL 60000
3250 027730  104023                          ERROR   +23             ;RETURN IS EHRE FOR ERROR - WRONG DATA WAS FETCHED
3251 027732  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3252                              ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3253 027734  012702  100002                  MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3254 027740  004737  002454                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3255 027744  006542                          MFPI    -(R2)           ;<HERE - READ FROM PHYSICAL 60000
3256 027746  000240                          NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3257 027750  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3258 027752  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3259                              ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3260 027754  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3261 027762  012702  001204                  MOV     #<$TMP2+2>,R2   ;LOAD ADDR. OF $TMP2+2 INTO R2
3262 027766  004737  002454                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3263 027772  006552                          MFPI    @-(R2)          ;<HERE - READ FROM PHYSICAL 60000
3264 027774  000240                          NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3265 027776  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3266 030000  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3267                              ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3268 030002  005002                          CLR     R2              ;MAKE REGISTER 2 A ZERO
3269 030004  004737  002454                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3270 030010  006562  100000                  MFPI    100000(R2)      ;<HERE - READ FROM PHYSICAL 60000
3271 030014  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3272 030016  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3273                              ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3274 030020  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD TEST LOC. V.A. INTO $TMP2
3275 030026  012702  001202                  MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3276 030032  004737  002454                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3277 030036  006572  000000                  MFPI    @0(R2)          ;USE $TMP2 TO FETCH VIRTUAL ADDRESS OF 60000
3278 030042  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3279 030044  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3280 030046  112737  000006  172310-         MOVB    #6,KIPDR4       ;MAKE KIPDR4 RESIDENT
3281 030054  000423                          BR      TST23           ;;BRANCH TO NEXT TEST
```

```
3283                                                    .SBTTL  MM TRAP CATCHER FOR PREVIOUS TEST
3284 030056  012637  001260          MFPIV2: MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3285 030062  012637  001262                  MOV     (KSP)+,TRAPPS
3286 030066  013737  177572  001264          MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
3287 030074  013737  177576  001270          MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3288 030102  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0 AND LEAVE
3289 030110  104026                          ERROR   +26             ;TRIED TO READ NON-RESIDENT PAGE
3290                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI'' = 000002
3291 030112  013746  001262                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3292 030116  013746  001260                  MOV     TRAPPC,-(KSP)
3293 030122  000002                          RTI
```

```
3308                                    .SBTTL  TEST # 23 - MOVE TO PREVIOUS (SUPERVISOR) I-SPACE
                                        ;*********************************************************************
                                        ;*TEST 23       MOVE TO PREVIOUS (SUPERVISOR) I-SPACE
                                        ;*
                                        ;*      THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE PREVIOUS MODE
                                        ;*      IS CLOCKED CORRECTLY.  THERE IS A DESCRIPTION BEFORE EACH DESTINATION
                                        ;*      MODE TESTED.  THE TEST ITSELF IS CARRIED OUT IN SUBROUTINE MFPITS,
                                        ;*      WHICH USES THE MTPI INSTRUCTION CODE FOLLOWING THE JSR CALL TO EXECUTE
                                        ;*      THE TEST.  *IMPORTANT* - ALL 'NOP'S' FOLLOWING MTPI'S OF MODES 1,2,
                                        ;*      4 AND 5 ARE TO BE LEFT ALONE.  THE SUBROUTINE LOADS THE TWO WORDS
                                        ;*      AFTER THE JSR CALL, PREPARING FOR MODES 3, 6 AND 7.
                                        ;*
                                        ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT WILL OCCUR AND
                                        ;*      TRAP TO MTPIV1, WHERE THE ERRORS ARE REPORTED.
                                        ;*
                                        ;*********************************************************************
030124  000004                        TST23:  SCOPE
3309 030126  012737  077406  172210    1$:     MOV     #77406,SIPDR4   ;SUPERVISOR I-SPACE PAGE 4 READ/WRITE
3310                                    ;THE FOLLOWING WILL TEST DSTM=0 MTPI
3311 030134  012737  010340  177776    2$:     MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
3312 030142  012746  007777                    MOV     #7777,-(KSP)    ;PUSH DATA ON KERNEL STACK
3313 030146  012737  030506  000250            MOV     #MTPIV1,MMVEC   ;LOAD ADDRESS OF THIS TEST'S TRAP CATCHER TO MMVEC
3314 030154  006606                            MTPI    SSP             ;LOAD SUPERVISOR STACK POINTER
3315 030156  006506                            MFPI    SSP             ;READ SUPERVISOR STACK POINTER
3316 030160  012737  015726  000250            MOV     #MGMERR,MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
3317 030166  012601                            MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3318 030170  022701  007777                    CMP     #7777,R1        ;WAS SUPERVISOR STACK POINTER CHANGED
3319 030174  001401                            BEQ     3$              ;BRANCH IF IT WAS
3320 030176  104025                            ERROR   +25             ;SUPERVISOR STACK POINTER NOT CHANGED
3321                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 3$' WITH 'BR 2$' = 000765
3322 030200  012737  010340  177776    3$:     MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
3323 030206  012746  000700                    MOV     #SUPSTK,-(KSP)  ;GET READY TO RESTORE SUPERVISOR S. POINT
3324 030212  012737  030506  000250            MOV     #MTPIV1,MMVEC   ;LOAD ADDRESS OF THIS TEST'S TRAP CATCHER TO MMVEC
3325 030220  006606                            MTPI    SSP             ;RESTORE SUPERVISOR STACK POINTER
3326 030222  012737  015726  000250            MOV     #MGMERR,MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
3327 030230                             4$:     ;THIS WILL TEST DSTM = 1 MTPI.
3328 030230  012737  002644  001110            MOV     #MTPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MTPILP IN SUBROUTINE
3329 030236  012737  010340  002646            MOV     #010340,MTPIPM  ;MAKE PREVIOUS MODE SUPER IN LOCATION IN SUBROUTINE
3330 030244  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3331 030250  012700  125252                    MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3332 030254  004737  002616                    JSR     PC,MTPITS       ;GO DO THE TEST USING THE MTPI INSTRUCTION FOUND
3333 030260  006612                            MTPI    (R2)            ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3334 030262  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3335 030264  000000                            .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3336 030266  104024                            ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3337 030270  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3338                                    ;THE FOLLOWING WILL TEST DSTM=2 MTPI.
3339 030272  012700  125252                    MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3340 030276  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3341 030302  004737  002616                    JSR     PC,MTPITS       ;GO DO THE TEST USING THE MTPI INSTRUCTION FOUND
3342 030306  006622                            MTPI    (R2)+           ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3343 030310  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3344 030312  177776                            .WORD   -2              ;ADD -2 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3345 030314  104024                            ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3346 030316  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3347                                    ;THIS WILL TEST DSTM = 3 MTPI.
3348 030320  012700  052525                    MOV     #52525,R0       ;LOAD TEST DATA INTO R0
```

```
3349 030324  004737  002616                 JSR     PC,MTPITS       ;GO DO THE TEST USING THE MTPI INSTRUCTION FOUND
3350 030330  006637  100000                 MTPI    @#100000        ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3351 030334  000000                         .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3352 030336  104024                         ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3353 030340  005726                         TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3354                                 ;THIS WILL TEST DSTM = 4 MTPI.
3355 030342  012700  125252                 MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3356 030346  012702  100002                 MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3357 030352  004737  002616                 JSR     PC,MTPITS       ;GO DO THE TEST USING THE MTPI INSTRUCTION FOUND
3358 030356  006642                         MTPI    -(R2)           ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3359 030360  000240                         NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3360 030362  000000                         .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3361 030364  104024                         ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3362 030366  005726                         TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3363                                 ;THE FOLLOWING WILL TEST DSTM=5 MTPI.
3364 030370  012700  052525                 MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3365 030374  012702  001204                 MOV     #<$TMP2+2>,R2   ;LOAD ADDR. OF LOC. $TMP2+2 INTO R2
3366 030400  012737  100000  001202         MOV     #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOC. INTO $TMP2
3367 030406  004737  002616                 JSR     PC,MTPITS       ;GO DO THE TEST USING THE MTPI INSTRUCTION FOUND
3368 030412  006652                         MTPI    @-(R2)          ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3369 030414  000240                         NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3370 030416  076576                         .WORD   100000-$TMP2    ;ADD 100000-$TMP2 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3371 030420  104024                         ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3372 030422  005726                         TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3373                                 ;THIS WILL TEST DSTM = 6 MTPI.
3374 030424  012700  052525                 MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3375 030430  005002                         CLR     R2              ;MAKE REGISTER 2 ZERO
3376 030432  004737  002616                 JSR     PC,MTPITS       ;GO DO THE TEST USING THE MTPI INSTRUCTION FOUND
3377 030436  006662  100000                 MTPI    100000(R2)      ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3378 030442  100000                         .WORD   100000          ;ADD 100000 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3379 030444  104024                         ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3380 030446  005726                         TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3381                                 ;THE FOLLOWING WILL TEST DSTM=7 MTPI.
3382 030450  012700  125252                 MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3383 030454  012737  100000  001202         MOV     #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOCATION INTO LOCATION $TMP2
3384 030462  012702  001202                 MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3385 030466  004737  002616                 JSR     PC,MTPITS       ;GO DO THE TEST USING THE MTPI INSTRUCTION FOUND
3386 030472  006672  000000                 MTPI    @0(R2)          ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3387 030476  076576                         .WORD   100000-$TMP2    ;ADD 100000-$TMP2 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3388 030500  104024                         ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3389 030502  005726                         TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3390 030504  000423                         BR      TST24           ;;BRANCH TO NEXT TEST
3391
3392 030506  012637  001260         MTPIV1: MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3393 030512  012637  001262                 MOV     (KSP)+,TRAPPS
3394 030516  013737  177572  001264         MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
3395 030524  013737  177576  001270         MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3396 030532  042737  160000  177572         BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3397 030540  104024                         ERROR   +24             ;TRIED TO LOAD A N.R. PAGE 4
3398                                 ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI''  000002
3399 030542  013746  001262                 MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3400 030546  013746  001260                 MOV     TRAPPC,-(KSP)
3401 030552  000002                         RTI                     ;RETURN TO TEST
```

```
3402                                    .SBTTL  TEST # 24 - MOVE TO PREVIOUS (USER) I-SPACE
                                        ;*****************************************************************
                                        ;*TEST 24       MOVE TO PREVIOUS (USER) I-SPACE
                                        ;*
                                        ;*      THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE PREVIOUS MODE
                                        ;*      IS CLOCKED CORRECTLY.  THERE IS A DESCRIPTION BEFORE EACH DESTINATION
                                        ;*      MODE TESTED.  THE TEST ITSELF IS CARRIED OUT IN SUBROUTINE MFPITS,
                                        ;*      WHICH USES THE MTPI INSTRUCTION CODE FOLLOWING THE JSR CALL TO EXECUTE
                                        ;*      THE TEST.  *IMPORTANT* - ALL 'NOP'S' FOLLOWING MTPI'S OF MODES 1,2,
                                        ;*      4 AND 5 ARE TO BE LEFT ALONE.  THE SUBROUTINE LOADS THE TWO WORDS
                                        ;*      AFTER THE JSR CALL, PREPARING FOR MODES 3, 6 AND 7.
                                        ;*
                                        ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT WILL OCCUR AND
                                        ;*      TRAP TO MTPIV1, WHERE THE ERRORS ARE REPORTED.
                                        ;*
                                        ;*****************************************************************
     030554  000004
3403 030556  012737  077406  177610    TST24:  SCOPE
3404 030564  012737  000600  177650    1$:     MOV     #77406,UIPDR4   ;USER I-SPACE PAGE 4 READ/WRITE
3405                                            MOV     #600,UIPAR4     ;MAP USER I PAGE 4 TO 12K
3406 030572  012737  030340  177776    ;THE FOLLOWING WILL TEST DSTM=0 MTPI
3407 030600  012746  007777    2$:     MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3408 030604  012737  031152  000250            MOV     #7777,-(KSP)    ;PUSH DATA ON KERNEL STACK
3409 030612  006606                            MOV     #MTPIV2,MMVEC   ;SET M.M. VECTOR TO 20$
3410 030614  006506                            MTPI    USP             ;LOAD USER STACK POINTER
3411 030616  012737  015726  000250            MFPI    USP             ;READ USER STACK POINTER
3412 030624  012601                            MOV     #MGMERR,MMVEC   ;RESTORE MM VECTOR TO NORMAL ROUTINE
3413 030626  022701  007777                    MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3414 030632  001401                            CMP     #7777,R1        ;WAS USER STACK POINTER CHANGED
3415 030634  104025                            BEQ     3$              ;BRANCH IF IT WAS
3416                                           ERROR   +25             ;USER STACK POINTER NOT CHANGED
3417 030636  012737  030340  177776    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 3$' WITH 'BR 2$' = 000765
3418 030644  012746  000600    3$:     MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3419 030650  012737  031152  000250            MOV     #USESTK,-(KSP)  ;GET READY TO RESTORE USER S. POINT
3420 030656  006606                            MOV     #MTPIV2,MMVEC   ;SET M.M. VECTOR TO 20$
3421 030660  012737  015726  000250            MTPI    USP             ;RESTORE USER STACK POINTER
3422                                           MOV     #MGMERR,MMVEC   ;RESTORE MM VECTOR TO NORMAL ROUTINE
3423 030666  012737  002644  001110    ;THIS WILL TEST DSTM = 1 MTPI.
3424 030674  012737  030340  002646            MOV     #MTPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MTPILP IN SUBROUTINE
3425 030702  012737  031152  002734            MOV     #030340,MTPIPM  ;SET PREVIOUS MODE = USER IN SUBROUTINE LOCATION
3426 030710  012702  100000                    MOV     #MTPIV2,MTPIVC  ;SET THIS TEST'S MM TRAP HANDLER IN MTPIVC
3427 030714  012700  125252                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3428 030720  004737  002616                    MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3429 030724  006612                            JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION LOCATED
3430 030726  000240                            MTPI    (R2)            ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3431 030730  000000                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3432 030732  104024                            .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3433 030734  005726                            ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3434                                           TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3435 030736  012700  125252    ;THE FOLLOWING WILL TEST DSTM=2 MTPI.
3436 030742  012702  100000            MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3437 030746  004737  002616            MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3438 030752  006622                    JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION LOCATED
3439 030754  000240                    MTPI    (R2)+           ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3440 030756  177776                    NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3441 030760  104024                    .WORD   -2              ;ADD -2 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3442 030762  005726                    ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
                                       TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
```

TEST # 24 - MOVE TO PREVIOUS (USER) I-SPACE

```
3443                                        ;THIS WILL TEST DSTM = 3 MTPI.
3444 030764  012700  052525                        MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3445 030770  004737  002616                        JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION LOCATED
3446 030774  006637  100000                        MTPI    @#100000        ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3447 031000  000000                                .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3448 031002  104024                                ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3449 031004  005726                                TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3450                                        ;THIS WILL TEST DSTM = 4 MTPI.
3451 031006  012700  125252                        MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3452 031012  012702  100002                        MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3453 031016  004737  002616                        JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION LOCATED
3454 031022  006642                                MTPI    -(R2)           ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3455 031024  000240                                NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3456 031026  000000                                .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3457 031030  104024                                ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3458 031032  005726                                TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3459                                        ;THE FOLLOWING WILL TEST DSTM=5 MTPI.
3460 031034  012700  052525                        MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3461 031040  012702  001204                        MOV     #<$TMP2+2>,R2   ;LOAD ADDR. OF LOC. $TMP2+2 INTO R2
3462 031044  012737  100000  001202                MOV     #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOC. INTO $TMP2
3463 031052  004737  002616                        JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION LOCATED
3464 031056  006652                                MTPI    @-(R2)          ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3465 031060  000240                                NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3466 031062  076576                                .WORD   100000-$TMP2    ;ADD 100000-$TMP2 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3467 031064  104024                                ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3468 031066  005726                                TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3469                                        ;THIS WILL TEST DSTM = 6 MTPI.
3470 031070  012700  052525                        MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3471 031074  005002                                CLR     R2              ;MAKE REGISTER 2 ZERO
3472 031076  004737  002616                        JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION LOCATED
3473 031102  006662  100000                        MTPI    100000(R2)      ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3474 031106  100000                                .WORD   100000          ;ADD 100000 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3475 031110  104024                                ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3476 031112  005726                                TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3477                                        ;THE FOLLOWING WILL TEST DSTM=7 MTPI.
3478 031114  012700  125252                        MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3479 031120  012737  100000  001202                MOV     #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOCATION
3480                                                                        ;INTO LOCATION $TMP2
3481 031126  012702  001202                        MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3482 031132  004737  002616                        JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION LOCATED
3483 031136  006672  000000                        MTPI    @0(R2)          ;<HERE LOAD TEST DATA INTO PHYSICAL 60000
3484 031142  076576                                .WORD   100000-$TMP2    ;ADD 100000-$TMP2 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3485 031144  104024                                ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3486 031146  005726                                TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3487 031150  000423                                BR      TST25           ;;BRANCH TO NEXT TEST
3488
3489 031152  012637  001260                MTPIV2: MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3490 031156  012637  001262                        MOV     (KSP)+,TRAPPS
3491 031162  013737  177572  001264                MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
3492 031170  013737  177576  001270                MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3493 031176  042737  160000  177572                BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3494 031204  104024                                ERROR   +24             ;TRIED TO LOAD A N.R. PAGE 4
3495                                        ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI' - 000002
3496 031206  013746  001262                        MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3497 031212  013746  001260                        MOV     TRAPPC,-(KSP)
3498 031216  000002                                RTI                     ;RETURN TO TEST
```

```
3513                                    .SBTTL  TEST # 25 - MFPI (KERNEL) TO SUPERVISOR MODE
                                        ;;*****************************************************************
                                        ;*TEST 25       MFPI (KERNEL) TO SUPERVISOR MODE
                                        ;*
                                        ;*      THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE FETCH IS FROM
                                        ;*      KERNEL SPACE.  THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE
                                        ;*      TESTED.  THE TEST ITSELF IS CARRIED OUT IN SUBROUTINE MFPITS,
                                        ;*      WHICH USES THE MFPI INSTRUCTION CODE FOLLOWING THE JSR CALL TO EXECUTE
                                        ;*      THE TEST.  *IMPORTANT* - ALL 'NOP'S' FOLLOWING MFPI'S OF MODES 1,2,
                                        ;*      4 AND 5 ARE TO BE LEFT ALONE.  THE SUBROUTINE LOADS THE TWO WORDS
                                        ;*      AFTER THE JSR CALL, PREPARING FOR MODES 3, 6 AND 7.
                                        ;*
                                        ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT WILL OCCUR AND
                                        ;*      TRAP TO MFPIV3, WHERE THE ERRORS ARE REPORTED.
                                        ;*
                                        ;*****************************************************************
 031220  000004
3514 031222  012737  031256  001110    TST25:  SCOPE
                                                MOV     #1$,$LPERR      ;SET LOOP ON ERROR TO 1$
3515 031230  012737  040340  177776            MOV     #040340,PSW     ;GO TO SUPERVISOR MODE FOR THIS TEST
3516 031236  012700  036514                    MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
3517 031242  010037  100000                    MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 60000
3518 031246  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3519                                    ;THE FOLLOWING WILL TEST DSTM=0 MFPI
3520 031252  105037  172210                    CLRB    SIPDR4          ;MAKE SUPERVISOR I-SPACE PAGE 4 NON-RESIDENT
3521 031256  012737  040340  177776    1$:     MOV     #040340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT SUPERVISOR
3522 031264  012737  031566  000250            MOV     #MFPIV3,MMVEC   ;SET M.M. VECTOR TO MFPIV3
3523 031272  006506                            MFPI    KSP             ;PUT KERNEL STACK POINTER ON SUPERVISOR STACK
3524 031274  012737  015726  000250            MOV     #MGMERR,MMVEC   ;RESTORE MM VECTOR TO NORMAL ROUTINE
3525 031302  022706  000700                    CMP     #SUPSTK,SSP     ;WAS SOMETHING PUSHED ON STACK AT THE MFPI
3526 031306  001407                            BEQ     2$              ;BRANCH TO ERROR IF NOTHING WAS PUSHED
3527 031310  012600                            MOV     (SSP)+,R0       ;POP SUPERVISOR STACK INTO R0
3528 031312  012701  001100                    MOV     #KERSTK,R1      ;EXPECTING 1100 AS KSP
3529 031316  020001                            CMP     R0,R1           ;DID YOU GET THE RIGHT POINTER?
3530 031320  001403                            BEQ     3$              ;BRANCH IF YOU DID
3531 031322  104023                            ERROR   +23             ;WRONG THING WAS PUSHED ON STACK
3532                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 3$'' WITH 'BR 1$'' = 000756
3533 031324  000401                            BR      3$              ;BRANCH TO NEXT TRY
3534 031326  104025                    2$:     ERROR   +25             ;NOTHING PUSHED ON STACK
3535                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 2$'' ABOVE WITH 'BR 1$'' = 000762
3536                                    ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3537 031330  012737  002506  001110    3$:     MOV     #MFPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MFPILP IN SUBROUTINE
3538 031336  012737  040340  002510            MOV     #040340,MFPIPS  ;MAKE PREVIOUS MODE KERNEL PRESENT SUPERVISOR
3539 031344  012737  031566  002612            MOV     #MFPIV3,MFPIVC  ;MOVE THIS TEST'S MM TRAP VECTOR TO MFPIVC
3540 031352  012700  036514                    MOV     #36514,R0       ;LOAD DATA EXPECTED INTO R0
3541 031356  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3542 031362  004737  002454                    JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3543 031366  006512                            MFPI    (R2)            ;<HERE - READ FROM PHYSICAL 60000
3544 031370  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3545 031372  104023                            ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3546 031374  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3547                                    ;THE FOLLOWING WILL TEST DSM=2 MFPI.
3548 031376  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3549 031402  004737  002454                    JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3550 031406  006522                            MFPI    (R2)+           ;<HERE - READ FROM PHYSICAL 60000
3551 031410  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3552 031412  104023                            ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3553 031414  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
```

```
3554                                        ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
3555 031416  004737  002454                       JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3556 031422  006537  100000                       MFPI    @#100000        ;<HERE - READ FROM PHYSICAL 60000
3557 031426  104023                              ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3558 031430  005726                              TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3559                                        ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3560 031432  012702  100002                       MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3561 031436  004737  002454                       JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3562 031442  006542                              MFPI    -(R2)           ;<HERE - READ FROM PHYSICAL 60000
3563 031444  000240                              NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3564 031446  104023                              ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3565 031450  005726                              TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3566                                        ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3567 031452  012737  100000  001202              MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3568 031460  012702  001204                       MOV     #<$TMP2+2>,R2   ;LOAD ADDRESS OF $TMP2+2 INTO R2
3569 031464  004737  002454                       JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3570 031470  006552                              MFPI    @-(R2)          ;<HERE - READ FROM PHYSICAL 60000
3571 031472  000240                              NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3572 031474  104023                              ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3573 031476  005726                              TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3574                                        ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3575 031500  005002                              CLR     R2              ;MAKE REGISTER 2 A ZERO
3576 031502  004737  002454                       JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3577 031506  006562  100000                       MFPI    100000(R2)      ;<HERE - READ FROM PHYSICAL 60000
3578 031512  104023                              ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3579 031514  005726                              TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3580                                        ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3581 031516  012737  100000  001202              MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR. INTO $TMP2
3582 031524  012702  001202                       MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3583 031530  004737  002454                       JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3584 031534  006572  000000                       MFPI    @0(R2)          ;<HERE - READ FROM PHYSICAL 60000
3585 031540  104023                              ERROR   +23             ;WRONG DATA WAS FETCHED
3586 031542  005726                              TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3587 031544  012737  000340  177776              MOV     #00340,PSW      ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
3588 031552  112737  000006  172210              MOVB    #6,SIPDR4       ;MAKE SIPDR4 RESIDENT
3589 031560  012706  001100                       MOV     #KERSTK,KSP     ;RESET KERNEL STACK POINTER
3590 031564  000423                              BR      TST26           ;;BRANCH TO NEXT TEXT
```

```
3592                                                    .SBTTL  MM TRAP ROUTINE FOR ABOVE TEST
3593 031566  012637  001260          MFPIV3: MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3594 031572  012637  001262                  MOV     (KSP)+,TRAPPS
3595 031576  013737  177572  001264          MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
3596 031604  013737  177576  001270          MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3597 031612  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3598 031620  104026                          ERROR   +26             ;TRIED TO READ NON-RESIDENT PAGE
3599                                  ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCITON WITH AN 'RTI' = 000002
3600 031622  013746  001262                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3601 031626  013746  001260                  MOV     TRAPPC,-(KSP)
3602 031632  000002                          RTI                     ;RETURN TO TEST
```

. . .

```
3603                                    .SBTTL  TEST # 26 - MFPI (KERNEL) TO USER MODE
                                        ;*****************************************************************
                                        ;*TEST 26       MFPI (KERNEL) TO USER MODE
                                        ;*
                                        ;*      THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE FETCH IS FROM
                                        ;*      KERNEL SPACE.  THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE
                                        ;*      TESTED.  THE TEST ITSELF IS CARRIED OUT IN SUBROUTINE MFPITS,
                                        ;*      WHICH USES THE MFPI INSTRUCTION CODE FOLLOWING THE JSR CALL TO EXECUTE
                                        ;*      THE TEST.  *IMPORTANT* - ALL 'NOP'S' FOLLOWING MFPI'S OF MODES 1,2,
                                        ;*      4 AND 5 ARE TO BE LEFT ALONE.  THE SUBROUTINE LOADS THE TWO WORDS
                                        ;*      AFTER THE JSR CALL, PREPARING FOR MODES 3, 6 AND 7.
                                        ;*
                                        ;*      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT WILL OCCUR AND
                                        ;*      TRAP TO MFPIV4, WHERE THE ERRORS ARE REPORTED.
                                        ;*
                                        ;*****************************************************************
     031634  000004
3604 031636  012737  031644  001110    TST26:  SCOPE
                                                MOV     #1$,$LPERR      ;SET LOOP ON ERROR TO 1$
3605 031644  012737  140340  177776    1$:     MOV     #140340,PSW     ;GO TO USER MODE FOR THIS TEST
3606 031652  012700  036514                    MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
3607 031656  010037  100000                    MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 60000
3608 031662  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3609                                    ;THE FOLLOWING WILL TEST DSTM=0 MFPI
3610 031666  012737  032176  000250            MOV     #MFPIV4,MMVEC   ;SET M.M. VECTOR TO MFPIV4
3611 031674  105037  177610                    CLRB    UIPDR4          ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT
3612 031700  012737  140340  177776            MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3613 031706  006506                            MFPI    KSP             ;PUT KERNEL STACK POINTER ON USER STACK
3614 031710  012737  015726  000250            MOV     #MMGMERR,MMVEC  ;RESTORE MM VECTOR TO NORMAL ROUTINE
3615 031716  022706  000600                    CMP     #USESTK,USP     ;WAS SOMETHING PUSHED ON STACK AT THE MFPI
3616 031722  001407                            BEQ     2$              ;BRANCH IF NOTHING WAS PUSHED
3617 031724  012600                            MOV     (USP)+,R0       ;POP USER STACK INTO R0
3618 031726  012701  001100                    MOV     #KERSTK,R1      ;EXPECTING 1100 AS KSP
3619 031732  020001                            CMP     R0,R1           ;DID YOU GET THE RIGHT POINTER?
3620 031734  001403                            BEQ     3$              ;BRANCH IF YOU DID
3621 031736  104023                            ERROR   +23             ;WRONG THING WAS PUSHED ON STACK
3622                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 3$'' WITH 'BR 1$'' = 000763
3623 031740  000401                            BR      3$              ;BRANCH TO NEXT TRY
3624 031742  104025                    2$:     ERROR   +25             ;NOTHING PUSHED ON STACK
3625                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 2$'' WITH 'BR 1$'' = 000763
3626
3627                                    ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3628 031744  012737  002506  001110    3$:     MOV     #MFPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MFPILP IN SUBROUTINE
3629 031752  012737  140340  002510            MOV     #140340,MFPIPS  ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3630 031760  012737  032176  002612            MOV     #MFPIV4,MFPIVC  ;MOVE THIS TEST'S MM TRAP HANDLER TO MFPIVC
3631 031766  012700  036514                    MOV     #36514,R0       ;LOAD DATA EXPECTED INTO R0
3632 031772  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3633 031776  004737  002454                    JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3634 032002  006512                            MFPI    (R2)            ;<HERE - READ FROM PHYSICAL 60000
3635 032004  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3636 032006  104023                            ERROR   +23             ;WRONG DATA WAS FETCHED
3637 032010  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3638                                    ;THE FOLLOWING WILL TEST DSM=2 MFPI.
3639 032012  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3640 032016  004737  002454                    JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3641 032022  006522                            MFPI    (R2)+           ;<HERE - READ FROM PHYSICAL 60000
3642 032024  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3643 032026  104023                            ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
```

```
3644 032030  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3645                              ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
3646 032032  004737  002454                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3647 032036  006537  100000                  MFPI    @#100000        ;<HERE - READ FROM PHYSICAL 60000
3648 032042  104023                          ERROR   +23             ;WRONG DATA WAS FETCHED
3649 032044  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3650                              ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3651 032046  012702  100002                  MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3652 032052  004737  002454                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3653 032056  006542                          MFPI    -(R2)           ;<HERE - READ FROM PHYSICAL 60000
3654 032060  000240                          NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3655 032062  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3656 032064  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3657                              ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3658 032066  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3659 032074  012702  001204                  MOV     #<$TMP2+2>,R2   ;LOAD ADDRESS OF $TMP2+2 INTO R2
3660 032100  004737  002454                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3661 032104  006552                          MFPI    @-(R2)          ;<HERE - READ FROM PHYSICAL 60000
3662 032106  000240                          NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3663 032110  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3664 032112  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3665                              ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3666 032114  005002                          CLR     R2              ;MAKE REGISTER 2 A ZERO
3667 032116  004737  002454                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3668 032122  006562  100000                  MFPI    100000(R2)      ;<HERE - READ FROM PHYSICAL 60000
3669 032126  104023                          ERROR   +23             ;WRONG DATA WAS FETCHED
3670 032130  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3671                              ;THE FOLLOWING WILL TEST DSTM-7 MFPI.
3672 032132  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR. INTO $TMP2
3673 032140  012702  001202                  MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3674 032144  004737  002454                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3675 032150  006572  000000                  MFPI    @0(R2)          ;<HERE - READ FROM PHYSICAL 60000
3676 032154  104023                          ERROR   +23             ;WRONG DATA WAS FETCHED
3677 032156  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3678 032160  012737  000340  177776          MOV     #00340,PSW      ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
3679 032166  112737  000006  177610          MOVB    #6,UIPDR4       ;MAKE UIPDR4 RESIDENT
3680 032174  000423                          BR      TST27           ;;BRANCH TO NEXT TEXT
```

```
                                                .SBTTL  MM TRAP HANDLER FOR ABOVE TEST
032176  012637  001260                  MFPIV4: MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
032202  012637  001262                          MOV     (KSP)+,TRAPPS
032206  013737  177572  001264                  MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
032214  013737  177576  001270                  MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
032222  042737  160000  177572                  BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
032230  104026                                  ERROR   +26             ;TRIED TO READ NON-RESIDENT PAGE
                                        ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI'' = 000002
032232  013746  001262                          MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
032236  013746  001260                          MOV     TRAPPC,-(KSP)
032242  000002                                  RTI                     ;RETURN TO TEST
```

```
3707                                    .SBTTL  TEST # 27 - MFPI (SUPERVISOR) WITH SUPER D-SPACE ENABLED
                                        ;*****************************************************************
                                        ;*TEST 27       MFPI (SUPERVISOR) WITH SUPER D-SPACE ENABLED
                                        ;*
                                        ;*      THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT PREVIOUS MODE IS
                                        ;*      CLOCKED CORRRECTLY, AND THAT D-SPACE IS NOT ENABLED.  THE TEST ITSELF
                                        ;*      IS CARRIED OUT IN SUBROUTINE MFPITS, WHICH USES THE MFPI INSTRUCTION
                                        ;*      CODE FOLLOWING THE JSR CALL TO EXECUTE THE TEST.  *IMPORTANT* - ALL
                                        ;*      'NOP'S'' FOLLOWING MFPI'S OF MODES 1,2, 4 AND 5 ARE TO BE LEFT ALONE.
                                        ;*      THE SUBFOUTINE LOADS THE TWO WORDS AFTER THE JSR CALL, PREPARING FOR
                                        ;*      MODES 3, 6 AND 7.
                                        ;*
                                        ;*      IF THE CORRECT MODE IS NOT ENABLED, A NON-RESIDENT ABORT WILL OCCUR AND
                                        ;*      TRAP TO MFPIV5, WHERE THE ERRORS ARE REPORTED.
                                        ;*
                                        ;*****************************************************************
     032244  000004
3708 032246  012700  077400            TST27:  SCOPE
3709                                            MOV     #77400,R0       ;MAKE PAGE 4 IN ALL BUT SUPERVISOR I
                                                                        ;AND KERNAL I NON-RESIDENT
3710 032252  010037  172330                    MOV     R0,KDPDR4       ;KERNAL D-SPACE PAGE 4
3711 032256  010037  172230                    MOV     R0,SDPDR4       ;SUPERVISOR D-SPACE PAGE 4
3712 032262  010037  177630                    MOV     R0,UDPDR4       ;USER D-SPACE PAGE 4
3713 032266  010037  177610                    MOV     R0,UIPDR4       ;USER I-SPACE PAGE 4
3714 032272  012700  036514                    MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
3715 032276  010037  060000                    MOV     R0,@#60000      ;LOAD DATA PATTERN INTO PHYS. 60000
3716 032302  052737  000002  172516          BIS     #BIT1,MMR3      ;ENABLE SUPERVISOR D-SPACE
3717 032310  105037  172310                    CLRB    KIPDR4          ;MAKE KERNAL I-SPACE PAGE 4 NON-RESIDENT
3718                                    ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3719 032314  012737  002506  001110          MOV     #MFPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MFPILP IN SUBROUTINE
3720 032322  012737  010340  002510          MOV     #010340,MFPIPS  ;MAKE PREVIOUS MODE SUPERVISOR IN SUBRTN LOCATION
3721 032330  012737  032454  002612          MOV     #MFPIV5,MFPIVC  ;MOVE THIS TEST'S MM TRAP HANDLER TO MFPIVC
3722 032336  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3723 032342  004737  002454                    JSR     PC,MFPITS       ;GO DO TEST USING THE MFPI INSTRUCTION FOUND
3724 032346  006512                            MFPI    (R2)            ;<HERE - READ FROM PHYSICAL 60000
3725 032350  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3726 032352  104037                            ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA FETCHED
3727 032354  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3728                                    ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
3729 032356  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3730 032362  004737  002454                    JSR     PC,MFPITS       ;GO DO TEST USING THE MFPI INSTRUCTION FOUND
3731 032366  006522                            MFPI    (R2)+           ;<HERE - READ FROM PHYSICAL 100000
3732 032370  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3733 032372  104037                            ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA FETCHED
3734 032374  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3735                                    ;THE FOLLOWING WILL TEST DSTM-3 MFPI.
3736 032376  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3737 032402  004737  002454                    JSR     PC,MFPITS       ;GO DO TEST USING THE MFPI INSTRUCTION FOUND
3738 032406  006537  100000                    MFPI    @#100000        ;<HERE - READ FROM PHYSICAL 100000
3739 032412  104037                            ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA FETCHED
3740 032414  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3741                                    ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3742 032416  012702  100002                    MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3743 032422  004737  002454                    JSR     PC,MFPITS       ;GO DO TEST USING THE MFPI INSTRUCTION FOUND
3744 032426  006542                            MFPI    -(R2)           ;<HERE - READ FROM PHYSICAL 100000
3745 032430  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3746 032432  104037                            ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA FETCHED
3747 032434  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
```

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

```
3748 032436  112737  000006  172310          MOVB    #6,KIPDR4       ;MAKE KIPDR4 RESIDENT
3749 032444  042737  000002  172516          BIC     #BIT1,MMR3      ;DISABLE SUPERVISOR D-SPACE
3750 032452  000426                          BR      TST30           ;;BRANCH TO NEXT TEST
```

MM TRAP HANDLER FOR ABOVE TEST

```
3752                                                    .SBTTL  MM TRAP HANDLER FOR ABOVE TEST
3753 032454  013737  177572  001264  MFPIV5: MOV     MMR0,WASSR0     ;SAVE MMR0 FOR ERROR TYPEOUT
3754 032462  013737  177574  001266          MOV     MMR1,WASSR1     ;SAVE MMR1 FOR ERROR TYPEOUT
3755 032470  013737  177576  001270          MOV     MMR2,WASSR2     ;SAVE MMR2 FOR ERROR TYPEOUT
3756 032476  013737  172516  001272          MOV     MMR3,WASSR3     ;SAVE MMR3 FOR ERROR TYPEOUT
3757 032504  012637  001260                  MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3758 032510  012637  001262                  MOV     (KSP)+,TRAPPS
3759 032514  104040                          ERROR   +40             ;TRIED TO READ NON-RESIDENT PAGE
3760                                ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI' - 000002
3761 032516  013746  001262                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3762 032522  013746  001260                  MOV     TRAPPC,-(KSP)
3763 032526  000002                          RTI
```

TEST # 30 - MTPI (SUPERVISOR) WITH SUPER. D-SPACE ENABLED

```
3778                                    .SBTTL  TEST # 30 - MTPI (SUPERVISOR) WITH SUPER. D-SPACE ENABLED
                                        ;*****************************************************************
                                        ;*TEST 30       MTPI (SUPERVISOR) WITH SUPER. D-SPACE ENABLED
                                        ;*
                                        ;*      THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE PREVIOUS MODE
                                        ;*      IS CLOCKED CORRECTLY, AND THAT D-SPACE IS NOT ENABLED.  THE TEST ITSELF
                                        ;*      IS CARRIED OUT IN SUBROUTINE MTPITS, WHICH USES THE MTPI INSTRUCTION
                                        ;*      CODE FOLLOWING THE JSR CALL TO EXECUTE THE TEST.  *IMPORTANT* - ALL
                                        ;*      'NOP'S'' FOLLOWING MTPI'S OF MODES 1,2, 4 AND 5 ARE TO BE LEFT ALONE.
                                        ;*      THE SUBROUTINE LOADS THE TWO WORDS AFTER THE JSR CALL, PREPARING FOR
                                        ;*      MODES 3, 6 AND 7.
                                        ;*
                                        ;*      IF THE CORRECT MODE IS NOT ENABLED, A NON-RESIDENT ABORT WILL OCCUR AND
                                        ;*      AND TRAP TO MTPIV3, WHERE THE ERRORS ARE REPORTED.
                                        ;*
                                        ;*****************************************************************
     032530  000004
3779 032532  052737  000002  172516    TST30:  SCOPE
                                                BIS     #BIT1,MMR3      ;ENABLE SUPERVISOR D-SPACE
3780
3781 032540  012737  002644  001110    ;THIS WILL TEST DSTM = 1 MTPI
                                                MOV     #MTPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MTPILP IN SUBROUTINE
3782 032546  012737  010340  002646             MOV     #010340,MTPIPM  ;MAKE PREVIOUS MODE SUPERVISOR IN LOCATION IN SUBRTN
3783 032554  012737  032714  002734             MOV     #MTPIV3,MTPIVC  ;LOAD THIS TEST'S MM TRAP HANDLER TO MTPIVC
3784 032562  012702  100000                     MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3785 032566  012700  125252                     MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3786 032572  004737  002616                     JSR     PC,MTPITS       ;GO DO THE TEST USING MTPI INSTRUCTION FOUND
3787 032576  006612                             MTPI    (R2)            ;LOAD TEST DATA INTO PHYSICAL 100000
3788 032600  000240                             NOP                     ;<HERE - NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3789 032602  000000                             .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3790 032604  104035                             ERROR   +35             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3791 032606  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3792
3793 032610  012700  052525             ;THIS WILL TEST DSTM = 3 MTPI
                                                MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3794 032614  004737  002616                     JSR     PC,MTPITS       ;GO DO THE TEST USING MTPI INSTRUCTION FOUND
3795 032620  006637  100000                     MTPI    @#100000        ;<HERE - LOAD TEST DATA INTO PHYSICAL 100000
3796 032624  000000                             .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3797 032626  104035                             ERROR   +35             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3798 032630  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3799
3800 032632  012700  125252             ;THIS WILL TEST DSTM = 4 MTPI
                                                MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3801 032636  012702  100002                     MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3802 032642  004737  002616                     JSR     PC,MTPITS       ;GO DO THE TEST USING MTPI INSTRUCTION FOUND
3803 032646  006642                             MTPI    -(R2)           ;<HERE - LOAD TEST DATA INTO PHYSICAL 100000
3804 032650  000240                             NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3805 032652  000000                             .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3806 032654  104035                             ERROR   +35             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3807 032656  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3808
3809 032660  012700  052525             ;THIS WILL TEST DSTM = 6 MTPI
                                                MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3810 032664  005002                             CLR     R2              ;MAKE R2 ZERO
3811 032666  004737  002616                     JSR     PC,MTPITS       ;GO DO THE TEST USING MTPI INSTRUCTION FOUND
3812 032672  006662  100000                     MTPI    100000(R2)      ;<HERE - LOAD TEST DATA INTO PHYSICAL 100000
3813 032676  100000                             .WORD   100000          ;ADD 100000 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3814 032700  104035                             ERROR   +35             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3815 032702  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3816 032704  042737  000002  172516             BIC     #BIT1,MMR3      ;DISABLE SUPERVISOR D-SPACE
3817 032712  000410                             BR      TST31           ;;BRANCH TO NEXT TEST
```

MM TRAP HANDLER FOR ABOVE TEST

```
3819                                            .SBTTL  MM TRAP HANDLER FOR ABOVE TEST
3820 032714  013700  177572          MTPIV3: MOV     MMR0,R0         ;SAVE MMR0 FOR ERROR TYPEOUT
3821 032720  013701  177574                  MOV     MMR1,R1         ;SAVE MMR1 FOR ERROR TYPEOUT
3822 032724  013702  177576                  MOV     MMR2,R2         ;SAVE MMR2 FOR ERROR TYPEOUT
3823 032730  104036                          ERROR   +36             ;TRIED TO LOAD A NON-RESIDENT PAGE 4
3824                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI' - 000002
3825 032732  000002                          RTI                     ;RETURN TO TEST
```

TEST # 31 - MTPI (USER) WITH USER D-SPACE ENABLED

```
3826                                    .SBTTL  TEST # 31 - MTPI (USER) WITH USER D-SPACE ENABLED
                                        ;*****************************************************************
                                        ;*TEST 31       MTPI (USER) WITH USER D-SPACE ENABLED
                                        ;*
                                        ;*      THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE PREVIOUS MODE
                                        ;*      IS CLOCKED CORRECTLY, AND THAT D-SPACE IS NOT ENABLED.  THE TEST ITSELF
                                        ;*      IS CARRIED OUT IN SUBROUTINE MTPITS, WHICH USES THE MTPI INSTRUCTION
                                        ;*      CODE FOLLOWING THE JSR CALL TO EXECUTE THE TEST.  *IMPORTANT* - ALL
                                        ;*      'NOP'S'' FOLLOWING MTPI'S OF MODES 1,2, 4 AND 5 ARE TO BE LEFT ALONE.
                                        ;*      THE SUBROUTINE LOADS THE TWO WORDS AFTER THE JSR CALL, PREPARING FOR
                                        ;*      MODES 3, 6 AND 7.
                                        ;*
                                        ;*      IF THE CORRECT MODE IS NOT ENABLED, A NON-RESIDENT ABORT WILL OCCUR AND
                                        ;*      AND TRAP TO MTPIV4, WHERE THE ERRORS ARE REPORTED.
                                        ;*
                                        ;*****************************************************************
032734  000004                  TST31:  SCOPE
3827 032736  012737  077400  172210        MOV     #77400,SIPDR4   ;MAKE SIPDR4 NON-RESIDENT
3828 032744  012737  077406  177610        MOV     #77406,UIPDR4   ;MAKE UIPDR4 RESIDENT
3829 032752  052737  000001  172516        BIS     #BIT0,MMR3      ;ENABLE USER D-SPACE
3830                                    ;THIS WILL TEST DSTM = 1 MTPI
3831 032760  012737  002644  001110        MOV     #MTPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MTPILP IN SUBROUTINE
3832 032766  012737  030340  002646        MOV     #030340,MTPIPM  ;MAKE LOCATION IN SUBROUTINE PREVIOUS MODE USER
3833 032774  012737  033134  002734        MOV     #MTPIV4,MTPIVC  ;PUT THIS TEST'S MM TRAP HANDLER ADDRESS IN LOC MTPIVC
3834 033002  012702  100000                MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3835 033006  012700  125252                MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3836 033012  004737  002616                JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION FOUND
3837 033016  006612                        MTPI    (R2)            ;<HERE - LOAD TEST DATA INTO PHYSICAL 100000
3838 033020  000240                        NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3839 033022  000000                        .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3840 033024  104035                        ERROR   +35             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3841 033026  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3842                                    ;THIS WILL TEST DSTM = 3 MTPI
3843 033030  012700  052525                MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3844 033034  004737  002616                JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION FOUND
3845 033040  006637  100000                MTPI    @#100000        ;<HERE - LOAD TEST DATA INTO PHYSICAL 100000
3846 033044  000000                        .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3847 033046  104035                        ERROR   +35             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3848 033050  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3849                                    ;THIS WILL TEST DSTM = 4 MTPI
3850 033052  012700  125252                MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3851 033056  012702  100002                MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3852 033062  004737  002616                JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION FOUND
3853 033066  006642                        MTPI    -(R2)           ;LOAD TEST DATA INTO PHYSICAL 100000
3854 033070  000240                        NOP                     ;<HERE - MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3855 033072  000000                        .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3856 033074  104035                        ERROR   +35             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3857 033076  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3858                                    ;THIS WILL TEST DSTM = 6 MTPI
3859 033100  012700  052525                MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3860 033104  005002                        CLR     R2              ;MAKE R2 ZERO
3861 033106  004737  002616                JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION FOUND
3862 033112  006662  100000                MTPI    100000(R2)      ;<HERE - LOAD TEST DATA INTO PHYSICAL 100000
3863 033116  100000                        .WORD   100000          ;ADD 100000 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3864 033120  104035                        ERROR   +35             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3865 033122  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3866 033124  042737  000001  172516        BIC     #BIT0,MMR3      ;DISABLE USER D-SPACE
```

TEST # 31 - MTPI (USER) WITH USER D-SPACE ENABLED

```
3867 033132  000410                    BR      TST32          ;;BRANCH TO NEXT TEST
```

MM TRAP CATCHER FOR THE ABOVE TEST

```
3869                                            .SBTTL  MM TRAP CATCHER FOR THE ABOVE TEST
3870 033134  013700  177572          MTPIV4:   MOV     MMR0,R0         ;SAVE MMR0 FOR ERROR TYPEOUT
3871 033140  013701  177574                    MOV     MMR1,R1         ;SAVE MMR1 FOR ERROR TYPEOUT
3872 033144  013702  177576                    MOV     MMR2,R2         ;SAVE MMR2 FOR ERROR TYPEOUT
3873 033150  104036                            ERROR   +36             ;TRIED TO LOAD A NON-RESIDENT PAGE 4
3874                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI' - 000002
3875 033152  000002                            RTI                     ;RETURN TO TEST
```

```
                                        .SBTTL  TEST # 32 - MFPI (PREVIOUS=CURRENT=KERNEL)
                                        ;*****************************************************************
                                        ;*TEST 32       MFPI (PREVIOUS=CURRENT=KERNEL)
                                        ;*
                                        ;*      THIS TEST CHECKS THAT IF BOTH PREVIOUS AND CURRENT MODES ARE KERNEL,
                                        ;*      AND THE SOURCE MODE IS 0, THE DESTINATION STACK IS NOT DECREMENTED
                                        ;*      BEFORE ACCESS. 'MFPI KSP' SHOULD PUSH THE NON-DECREMENTED VALUE OF KSP
                                        ;*      (1100) ONTO THE STACK (AT LOC. 1076).
                                        ;*
                                        ;*****************************************************************
033154  000004                          TST32:  SCOPE
3885 033156  112737  000006  172330     1$:     MOVB    #6,KDPDR4       ;MAKE KDPDR4 RESIDENT
3886 033164  005037  177776             CLR     @#PSW           ;SET PREVIOUS = CURRENT = KERNEL
3887 033170  012700  001100             MOV     #STACK,R0       ;SETUP VALUE FOR STACK POINTER
3888 033174  010006                     MOV     R0,KSP          ;LOAD STACK POINTER
3889 033176  006506                     MFPI    KSP             ;THE VALUE 'STACK' SHOULD BE PUSHED BEFORE BEING DEC'D
3890 033200  011601                     MOV     (KSP),R1        ;READ DATA WHICH WAS PUSHED
3891 033202  020001                     CMP     R0,R1           ;WAS THE ORIGINAL VALUE OF THE STACK POINTER PUSHED?
3892 033204  001401                     BEQ     2$              ;BRANCH IF YES
3893 033206  104037                     ERROR   +37             ;MFPI FETCHED WRONG DATA
3894                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 2$' WITH 'BR 1$' = 000767
3895 033210  005740                     2$:     TST     -(R0)           ;SETUP EXPECTED STACK POINTER VALUE
3896 033212  020600                     CMP     KSP,R0          ;WAS THE STACK POINTER DECREMENTED?
3897 033214  001401                     BEQ     3$              ;BRANCH IF YES
3898 033216  104025                     ERROR   +25             ;STACK NOT PUSHED BY THE MFPI
3899                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 3$' WITH 'BR 1$' = 000760
3900 033220  012706  001100             3$:     MOV     #STACK,KSP      ;RESTORE STACK POINTER
```

TEST # 33 - MFPD (SUPERVISOR) WITH SUPER D-SPACE ENABLED

```
3914                                    .SBTTL  TEST # 33 - MFPD (SUPERVISOR) WITH SUPER D-SPACE ENABLED
                                        ;*****************************************************************
                                        ;*TEST 33       MFPD (SUPERVISOR) WITH SUPER D-SPACE ENABLED
                                        ;*
                                        ;*      THIS TEST CHECKS TO SEE THAT THE REFERENCE IS TO D-SPACE IF THE INSRUC-
                                        ;*      TION IS AN MFPD.  THE TEST ITSELF IS CARRIED OUT IN SUBROUTINE MFPDTS,
                                        ;*      WHICH USES THE MFPD INSTRUCTION CODE FOLLOWING THE JSR CALL TO EXECUTE
                                        ;*      THE TEST.  *IMPORTANT* - ALL 'NOP'S' FOLLOWING MFPD'S OF MODES 1,2,
                                        ;*      4 AND 5 ARE TO BE LEFT ALONE.  THE SUBROUTINE LOADS THE TWO WORDS
                                        ;*      AFTER THE JSR CALL, PREPARING FOR MODES 3, 6 AND 7.
                                        ;*
                                        ;*      IF THE CORRECT MODE IS NOT ENABLED, A NON-RESIDENT ABORT WILL OCCUR
                                        ;*      AND TRAP TO MFPDV1, WHERE THE ERRORS ARE REPORTED.
                                        ;*
                                        ;*****************************************************************
     033224  000004
3915 033226  012737  000600  172270    TST33:  SCOPE
                                                MOV     #600,SDPAR4     ;MAP SDPAR4 TO 12K
3916 033234  012700  077400            20$:    MOV     #77400,R0       ;MAKE PAGE 4 IN ALL BUT SUPERVISOR D
3917                                                                    ;AND KERNAL I NON-RESIDENT
3918 033240  010037  172330                    MOV     R0,KDPDR4       ;KERNAL D-SPACE PAGE 4
3919 033244  010037  172210                    MOV     R0,SIPDR4       ;SUPERVISOR I-SPACE PAGE 4
3920 033250  010037  177630                    MOV     R0,UDPDR4       ;USER D-SPACE PAGE 4
3921 033254  010037  177610                    MOV     R0,UIPDR4       ;USER I-SPACE PAGE 4
3922 033260  012737  077406  172230          MOV     #77406,SDPDR4   ;MAKE SDPDR4 RESIDENT
3923 033266  012700  036514                    MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
3924 033272  010037  100000                    MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHYS. 100000
3925 033276  052737  000002  172516          BIS     #BIT1,MMR3      ;ENABLE SUPERVISOR D-SPACE
3926 033304  105037  172310                    CLRB    KIPDR4          ;MAKE KERNAL I-SPACE PAGE 4 NON-RESIDENT
3927                                    ;THE FOLLOWING WILL TEST DSTM=1 MFPD
3928 033310  012737  002760  001110          MOV     #MFPDLP,$LPERR  ;SET LOOP ON ERROR POINTER TO MFPDLP IN SUBROUTINE
3929 033316  012737  010340  002762          MOV     #010340,MFPDPS  ;MOVE PREVIOUS MODE=SUPERVISOR TO LOCATION IN SUBRTN
3930 033324  012737  033456  003030          MOV     #MFPDV1,MFPDVC  ;PUT ADDRESS OF THIS TEST'S MM TRAP CATCHER IN MFPDVC
3931 033332  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3932 033336  004737  002736                    JSR     PC,MFPDTS       ;GO DO TEST USING THE MFPD INSTRUCTION LOCATED
3933 033342  106512                            MFPD    (R2)            ;<HERE - READ FROM PHYSICAL 100000
3934 033344  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3935 033346  104037                            ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3936 033350  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3937                                    ;THE FOLLOWING WILL TEST DSTM=2 MFPD
3938 033352  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3939 033356  004737  002736                    JSR     PC,MFPDTS       ;GO DO TEST USING THE MFPD INSTRUCTION LOCATED
3940 033362  106522                            MFPD    (R2)+           ;<HERE - READ FROM PHYSICAL 100000
3941 033364  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3942 033366  104037                            ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3943 033370  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3944                                    ;THE FOLLOWING WILL TEST DSTM=3 MFPD
3945 033372  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3946 033376  004737  002736                    JSR     PC,MFPDTS       ;GO DO TEST USING THE MFPD INSTRUCTION LOCATED
3947 033402  106537  100000                    MFPD    @#100000        ;<HERE - READ FROM PHYSICAL 100000
3948 033406  104037                            ERROR   +37             ;WRONG DATA WAS FETCHED
3949 033410  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3950                                    ;THE FOLLOWING WILL TEST DSTM=4 MFPD
3951 033412  012702  100002                    MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3952 033416  004737  002736                    JSR     PC,MFPDTS       ;GO DO TEST USING THE MFPD INSTRUCTION LOCATED
3953 033422  106542                            MFPD    -(R2)           ;<HERE - READ FROM PHYSICAL 100000
3954 033424  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3955 033426  104037                            ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
```

TEST # 33 - MFPD (SUPERVISOR) WITH SUPER D-SPACE ENABLED

```
3956 033430  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3957 033432  042737  000002  172516          BIC     #BIT1,MMR3      ;DISABLE SUPERVISOR D-SPACE
3958 033440  012700  077406                  MOV     #77406,R0       ;SET UP R0 FOR 4K RESIDENT R/W
3959 033444  010037  172310                  MOV     R0,KIPDR4       ;MAKE KIPAR4 RESIDENT
3960 033450  010037  177630                  MOV     R0,UDPDR4       ;MAKE UDPDR4 RESIDENT
3961 033454  000423                          BR      TST34           ;;BRANCH TO NEXT TEST
```

MM TRAP HANDLER FOR THE ABOVE TEST

```
3963                                                    .SBTTL  MM TRAP HANDLER FOR THE ABOVE TEST
3964 033456  013737  177572  001264  MFPDV1: MOV     MMR0,WASSR0     ;SAVE MMR0 FOR ERROR TYPEOUT
3965 033464  013737  177574  001266          MOV     MMR1,WASSR1     ;SAVE MMR1 FOR ERROR TYPEOUT
3966 033472  013737  177576  001270          MOV     MMR2,WASSR2     ;SAVE MMR2 FOR ERROR TYPEOUT
3967 033500  012637  001260                  MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3968 033504  012637  001262                  MOV     (KSP)+,TRAPPS
3969 033510  104040                          ERROR   +40             ;TRIED TO READ NON-RESIDNT PAGE
3970                                 ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI'' = 000002
3971 033512  013746  001262                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3972 033516  013746  001260                  MOV     TRAPPC,-(KSP)
3973 033522  000002                          RTI                     ;RETURN TO TEST
```

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

```
3985                                    .SBTTL  TEST # 34 - MFPI (USER/PREV USER) WITH USER D-SPACE ENABLED
                                        ;;*****************************************************************
                                        ;*TEST 34       MFPI (USER/PREV USER) WITH USER D-SPACE ENABLED
                                        ;*
                                        ;*      THIS TEST CHECKS THAT IF THE INSTRUCTION IS AN MFPI AND BOTH THE
                                        ;*      PRESENT AND PREVIOUS MODES ARE USER, THEN D-SPACE IS USED IF IT IS
                                        ;*      ENABLED.  IN THIS WAY AN OPERATING SYSTEM CAN MAKE PROPRETARY CODE
                                        ;*      "EXECUTE ONLY" FOR THE USER.
                                        ;*
                                        ;*      IF THE CORRECT MODE IS NOT ENABLED, A NON-RESIDENT ABORT WILL OCCUR AND
                                        ;*      TRAP TO MFPIV6, WHERE THE ERRORS ARE REPORTED.
                                        ;*
                                        ;;*****************************************************************
     033524  000004                     TST34:  SCOPE
3986 033526  012737  000600  177670             MOV     #600,UDPAR4     ;MAP UDPAR4 TO 12K
3987 033534  012700  036514             MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
3988 033540  010037  100000             MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHYS. 100000
3989 033544  052737  000001  172516     BIS     #BIT0,MMR3      ;ENABLE USER D-SPACE
3990 033552  105037  172230             CLRB    SDPDR4          ;MAKE SDPDR4 NON-RESIDENT
3991 033556  105037  172310             CLRB    KIPDR4          ;MAKE KERNAL I-SPACE PAGE 4 NON-RESIDENT
3992                                    ;THE FOLLOWING WILL TEST DSTM=1 MFPI
3993 033562  012737  002506  001110     MOV     #MFPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MFPILP IN SUBROUTINE
3994 033570  012737  170340  002510     MOV     #170340,MFPIPS  ;SET PREVIOUS AND CURRENT MODE TO LOCATION IN SUBRTN
3995 033576  012737  033730  002612     MOV     #MFPIV6,MFPIVC  ;PUT ADDRESS OF THIS TEST'S TRAP CATCHER IN MFPIVC
3996 033604  012702  100000             MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3997 033610  004737  002454             JSR     PC,MFPITS       ;GO DO TEST USING THE MFPI INSTRUCTION LOCATED
3998 033614  006512                     MFPI    (R2)            ;<HERE - READ FROM PHYSICAL 100000
3999 033616  000240                     NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
4000 033620  104037                     ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
4001 033622  005726                     TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4002                                    ;THE FOLLOWING WILL TEST DSTM=2 MFPI
4003 033624  012702  100000             MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4004 033630  004737  002454             JSR     PC,MFPITS       ;GO DO TEST USING THE MFPI INSTRUCTION LOCATED
4005 033634  006522                     MFPI    (R2)+           ;<HERE - READ FROM PHYSICAL 100000
4006 033636  000240                     NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
4007 033640  104037                     ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
4008 033642  005726                     TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4009                                    ;THE FOLLOWING WILL TEST DSTM=3 MFPI
4010 033644  012702  100000             MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4011 033650  004737  002454             JSR     PC,MFPITS       ;GO DO TEST USING THE MFPI INSTRUCTION LOCATED
4012 033654  006537  100000             MFPI    @#100000        ;<HERE - READ FROM PHYSICAL 100000
4013 033660  104037                     ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
4014 033662  005726                     TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4015                                    ;THE FOLLOWING WILL TEST DSTM=4 MFPI
4016 033664  012702  100002             MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4017 033670  004737  002454             JSR     PC,MFPITS       ;GO DO TEST USING THE MFPI INSTRUCTION LOCATED
4018 033674  006542                     MFPI    -(R2)           ;<HERE - READ FROM PHYSICAL 100000
4019 033676  000240                     NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
4020 033700  104037                     ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
4021 033702  005726                     TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4022 033704  042737  000001  172516     BIC     #BIT0,MMR3      ;DISABLE USER D-SPACE
4023 033712  012737  000340  177776     MOV     #340,PSW        ;MAKE PRESENT MODE KERNAL
4024 033720  112737  000006  172310     MOVB    #6,KIPDR4       ;MAKE KIPDR4 RESIDENT
4025 033726  000423                     BR      TST35           ;;BRANCH TO NEXT TEST
```

MM TRAP CATCHER FOR ABOVE TEST

```
4027                                                    .SBTTL  MM TRAP CATCHER FOR ABOVE TEST
4028 033730  013737  177572  001264  MFPIV6: MOV     MMR0,WASSR0     ;SAVE MMR0 FOR ERROR TYPEOUT
4029 033736  013737  177574  001266          MOV     MMR1,WASSR1     ;SAVE MMR1 FOR ERROR TYPEOUT
4030 033744  013737  177576  001270          MOV     MMR2,WASSR2     ;SAVE MMR2 FOR ERROR TYPEOUT
4031 033752  012637  001260                  MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
4032 033756  012637  001262                  MOV     (KSP)+,TRAPPS
4033 033762  104040                          ERROR   +40             ;TRIED TO READ NON-RESIDENT PAGE
4034                                  ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI'' = 000002
4035 033764  013746  001262                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
4036 033770  013746  001260                  MOV     TRAPPC,-(KSP)
4037 033774  000002                          RTI                     ;RETURN TO TEST
```

```
 4045                                   .SBTTL  TEST # 35 - TEST CSM INSTRUCTION - ILLEGAL KERNEL MODE
                                        ;;*****************************************************************
                                        ;*TEST 35       TEST CSM INSTRUCTION - ILLEGAL KERNEL MODE
                                        ;;*****************************************************************
                                        ;*      THIS TEST CHECKS OUT THE CSM (CALL SUPERVISOR MODE)
                                        ;*      INSTRUCTION TO MAKE SURE CSM IS ILLEGAL WHEN DISABLED
                                        ;*      IN KERNEL MODE.
                                        ;;*****************************************************************
                                        ;;*****************************************************************
      033776  000004                    TST35:  SCOPE
 4046 034000  005037  172516                     CLR     MMR3            ;MAKE SURE MMR3 IS CLEARED. DISABLING CSM
 4047 034004  005037  177776                     CLR     PSW             ;MAKE SURE PSW PREVIOUS=CURRENT=KERNEL
 4048 034010  013737  000010  001204           MOV     RESVEC,$TMP3    ;SAVE TRAPS TO 10 VECTOR
 4049 034016  012737  034032  000010           MOV     #1$,RESVEC      ;TRAPS TO 10 GO TO 1$
 4050 034024  005237  177572                     INC     MMR0            ;TURN ON MEMORY MANAGEMENT
 4051 034030  007000                           CSM0                    ;CSM    R0
 4052 034032  013737  001204  000010  1$:      MOV     $TMP3,RESVEC    ;RESTORE TRAPS TO 10 VECTOR
 4053 034040  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
 4054 034044  005037  177776                     CLR     PSW             ;RETURN TO KERNEL MODE
 4055 034050  022726  034032                     CMP     #1$,(SP)+       ;SEE IF IT WAS AN EXPECTED TRAP
 4056 034054  001403                           BEQ     2$              ;BRANCH TO RESTORE STACK IF IT WAS
 4057 034056  022626                           CMP     (SP)+,(SP)+     ;CORRECT STACK
 4058 034060  104041                           ERROR   +41             ;ILLEGAL CSM DID NOT TRAP TO 10
 4059 034062  000401                           BR      TST36           ;;BRANCH AROUND STACK CORRECTION
 4060 034064  005726                    2$:     TST     (SP)+           ;CORRECT STACK
```

```
 4068                                   .SBTTL  TEST # 36 - TEST CSM INSTRUCTION - ILLEGAL SUPERVISOR MODE
                                        ;;*****************************************************************
                                        ;*TEST 36       TEST CSM INSTRUCTION - ILLEGAL SUPERVISOR MODE
                                        ;;*****************************************************************
                                        ;*
                                        ;*      THIS TEST CHECKS OUT THE CSM (CALL SUPERVISOR MODE)
                                        ;*      INSTRUCTION TO MAKE SURE CSM IS ILLEGAL WHEN DISABLED
                                        ;*      IN SUPERVISOR MODE.
                                        ;;*****************************************************************
                                        ;;*****************************************************************
      034066  000004                    TST36:  SCOPE
 4069 034070  005037  172516                     CLR     MMR3            ;MAKE SURE MMR3 IS CLEARED, DISABLING CSM
 4070 034074  052737  040000  177776             BIS     #40000,PSW      ;GO TO SUPERVISOR MODE
 4071 034102  013737  000010  001204             MOV     RESVEC,$TMP3    ;SAVE TRAPS TO 10 VECTOR
 4072 034110  012737  034124  000010             MOV     #1$,RESVEC      ;TRAPS TO 10 GO TO 1$
 4073 034116  005237  177572                     INC     MMR0            ;TURN ON MEMORY MANAGEMENT
 4074 034122  007000                             CSM0                    ;CSM    R0
 4075 034124  013737  001204  000010  1$:       MOV     $TMP3,RESVEC    ;RESTORE TRAPS TO 10 VECTOR
 4076 034132  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
 4077 034136  005037  177776                     CLR     PSW             ;GO BACK TO KERNEL MODE
 4078 034142  022726  034124                     CMP     #1$,(SP)+       ;SEE IF IT WAS AN EXPECTED TRAP
 4079 034146  001403                             BEQ     2$              ;BRANCH AROUND ERROR CALL IF IT WAS
 4080 034150  022626                             CMP     (SP)+,(SP)+     ;CORRECT STACK
 4081 034152  104041                             ERROR   +41             ;ILLEGAL CSM DID NOT TRAP TO 10
 4082 034154  000401                             BR      TST37           ;;BRANCH AROUND STACK CORRE TION
 4083 034156  005726                     2$:     TST     (SP)+           ;CORRECT STACK
```

```
                                        .SBTTL  TEST # 37 - TEST CSM INSTRUCTION - ILLEGAL USER MODE
                                        ;;*****************************************************************
                                        ;*TEST 37       TEST CSM INSTRUCTION - ILLEGAL USER MODE
                                        ;;*****************************************************************
                                        ;*      THIS TEST CHECKS OUT THE CSM (CALL SUPERVISOR MODE)
                                        ;*      INSTRUCTION TO MAKE SURE CSM IS ILLEGAL WHEN DISABLED
                                        ;*      IN USER MODE.
                                        ;;*****************************************************************
                                        ;;*****************************************************************
 4091
     034160  000004                    TST37:  SCOPE
 4092 034162  005037  172516                    CLR     MMR3            ;MAKE SURE MMR3 IS CLEARED, DISABLING CSM
 4093 034166  052737  140000  177776            BIS     #140000,PSW     ;GO TO USER MODE
 4094 034174  013737  000010  001204            MOV     RESVEC,$TMP3    ;SAVE TRAPS TO 10 VECTOR
 4095 034202  012737  034216  000010            MOV     #1$,RESVEC      ;TRAPS TO 10 GO TO 1$
 4096 034210  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
 4097 034214  007000                            CSM0                    ;CSM    R0
 4098 034216  013737  001204  000010  1$:     MOV     $TMP3,RESVEC    ;RESTORE TRAPS TO 10 VECTOR
 4099 034224  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
 4100 034230  005037  177776                    CLR     PSW             ;RETURN TO KERNEL MODE
 4101 034234  022726  034216                    CMP     #1$,(SP)+       ;SEE IF IT WAS AN EXPECTED TRAP
 4102 034240  001403                            BEQ     2$              ;BRANCH AROUND ERROR CALL IF IT WAS
 4103 034242  022626                            CMP     (SP)+,(SP)+     ;CORRECT STACK
 4104 034244  104041                            ERROR   +41             ;ILLEGAL CSM DID NOT TRAP TO 10
 4105 034246  000401                            BR      TST40           ;.BRANCH AROUND STACK CORRECTION
 4106 034250  005726                    2$:     TST     (SP)+           ;CORRECT STACK
```

```
  4114                                  .SBTTL  TEST # 40 - TEST CSM INSTRUCTION - ILLEGAL KERNEL MODE
                                        ;;*****************************************************************
                                        ;*TEST 40       TEST CSM INSTRUCTION - ILLEGAL KERNEL MODE
                                        ;;*****************************************************************
                                        ;*      THIS TEST CHECKS OUT THE CSM (CALL SUPERVISOR MODE)
                                        ;*      INSTRUCTION TO MAKE SURE CSM IS ILLEGAL WHEN ENABLED
                                        ;*      IN KERNEL MODE.
                                        ;;*****************************************************************
                                        ;;*****************************************************************
     034252  000004                     TST40:  SCOPE
  4115 034254  052737  000010  172516           BIS     #BIT3,MMR3      ;TURN ON CSM ENABLE
  4116 034262  005037  177776                   CLR     PSW             ;MAKE SURE PSW PREVIOUS=CURRENT-KERNEL
  4117 034266  013737  000010  001204           MOV     RESVEC,$TMP3    ;SAVE TRAPS TO 10 VECTOR
  4118 034274  012737  034310  000010           MOV     #1$,RESVEC      ;TRAPS TO 10 GO TO 1$
  4119 034302  005237  177572                   INC     MMR0            ;TURN ON MEMORY MANAGEMENT
  4120 034306  007000                           CSM0                    ;CSM    R0
  4121 034310  013737  001204  000010  1$:     MOV     $TMP3,RESVEC    ;RESTORE TRAPS TO 10 VECTOR
  4122 034316  005037  177572                   CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
  4123 034322  022726  034310                   CMP     #1$,(SP)+       ;SEE IF IT WAS AN EXPECTED TRAP
  4124 034326  001403                           BEQ     2$              ;BRANCH AROUND ERROR CALL IF IT WAS
  4125 034330  022626                           CMP     (SP)+,(SP)+     ;CORRECT STACK
  4126 034332  104041                           ERROR   +41             ;ILLEGAL CSM DID NOT TRAP TO 10
  4127 034334  000401                           BR      TST41           ;;BRANCH AROUND STACK CORRECTION
  4128 034336  005726                   2$:     TST     (SP)+           ;CORRECT STACK
```

```
 4137                                   .SBTTL  TEST # 41 - TEST CSM INSTRUCTION - I-SPACE SUPERVISOR MODE
                                        ;;*****************************************************************
                                        ;*TEST 41       TEST CSM INSTRUCTION - I-SPACE SUPERVISOR MODE
                                        ;;*****************************************************************
                                        ;*      THIS TEST CHECKS OUT THE CSM (CALL SUPERVISOR MODE)
                                        ;*      INSTRUCTION TO MAKE SURE CSM IS LEGAL WHEN ENABLED
                                        ;*      IN SUPERVISOR MODE, AND FOR A CHECK OF THE PARAMETERS
                                        ;*      PUSHED ON THE STACK.
                                        ;;*****************************************************************
                                        ;;*****************************************************************
     034340  000004                     TST41:  SCOPE
 4138 034342  012700  052525                     MOV     #52525,R0       ;SET UP R0
 4139 034346  004737  002050                     JSR     PC,APRINIT      ;RESET ALL MEMORY MANAGEMENT REGISTERS
 4140 034352  012737  000600  172260             MOV     #600,SDPAR0     ;IF WE GO TO D-SPACE, FLAG AS AN ERROR
 4141 034360  012737  000032  172516             MOV     #32,MMR3        ;ENABLE CSM, 22-BIT, SUPERVISOR SPACE
 4142 034366  012737  040000  035212             MOV     #40000,PCSMPS   ;SET PRECSM PS IN LOCATION
 4143 034374  012706  000700                     MOV     #SUPSTK,SSP     ;SETUP SUPERVISOR STACK POINTER
 4144 034400  004737  034422                     JSR     PC,CSMSUB       ;TEST CSM INSTRUCTION SUBROUTINE
 4145 034404  000000                             .WORD   0               ; 0 ADDED TO ADDRESS OF TRAP VECTOR
 4146 034406  005037  177776                     CLR     PSW             ;RETURN TO KERNEL MODE
 4147 034412  005037  172260                     CLR     SDPAR0          ;RESET SDPAR0 TO FIRST 4K
 4148 034416  000137  035236                     JMP     TST42           ;JUMP OVER THE CSM SUBROUTINE
```

```
4149                                                    .SBTTL  SUBROUTINE TO CHECK OPERATION OF AN EXECUTABLE CSM INST'N
4150 034422  012605                          CSMSUB:  MOV     (SP)+,R5        ;PUT RETURN ADDRESS INTO R5
4151 034424  012737  000340  177776                   MOV     #340,PSW        ;KERNEL MODE
4152 034432  010037  035216                           MOV     R0,PCSMR0       ;STORE PRECSM R0
4153 034436  012504                                   MOV     (R5)+,R4        ;MOVE OFFSET TO R4
4154 034440  010537  035234                           MOV     R5,RETURN       ;SET RETURN LOCATION ADDRESS TO LOCATION 'RETURN'
4155 034444  012737  034576  035210                   MOV     #5$,DOWELP      ;MOVE ADDRESS FOR INITIAL TESTING TO DOWELP
4156 034452  012764  034514  000010                   MOV     #2$,RESVEC(R4)  ;SETUP TRAPS TO 10 VECTOR TO 2$
4157 034460  012737  034506  001110                   MOV     #1$,$LPERR      ;SETUP LOOP ON ERROR TO 1$
4158 034466  053737  035212  177776                   BIS     PCSMPS,PSW      ;PUT USER/SUPERVISOR IN THE PSW
4159 034474  010637  035214                           MOV     SP,PCSMSP       ;MOVE STACK POINTER VALUE TO PCSMSP
4160 034500  162737  000006  035214                   SUB     #6,PCSMSP       ;SUBTRACT 6 FROM PCSMSP - EXPECT 3 PUSHES ON STACK
4161 034506  005237  177572                1$:      INC     MMR0            ;TURN ON MEMORY MANAGEMENT
4162 034512  007000                                   CSM0
4163 034514  012737  000012  000010        2$:      MOV     #RESVEC+2,RESVEC;RESTORE TRAPS TO 10 VECTOR
4164 034522  022716  034514                           CMP     #2$,(SP)        ;SEE IF CSM ABORTED
4165 034526  001011                                   BNE     4$              ;BRANCH IF IT DIDN'T TO EXECUTE TEST
4166 034530  012737  000340  177776        3$:      MOV     #340,PSW        ;GO BACK TO KERNEL PRIORITY 7
4167 034536  005037  177572                           CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4168 034542  022626                                   CMP     (SP)+,(SP)+     ;CORRECT STACK
4169 034544  104051                                   ERROR   +51             ;CSM ABORTED WHEN IT SHOULD NOT HAVE
4170 034546  000177  000462                           JMP     @RETURN         ;JUMP TO ADDRESS PRESTORED IN LOCATION RETURN
4171 034552  010637  035230                4$:      MOV     SSP,ACSMSP      ;SAVE AFTER CSM STACK POINTER
4172 034556  012637  035220                           MOV     (SP)+,CSM1ST    ;POP 1ST WORD OFF STACK
4173 034562  012637  035222                           MOV     (SP)+,CSM2ND    ;POP 2ND WORD OFF STACK
4174 034566  012637  035224                           MOV     (SP)+,CSM3RD    ;POP 3RD WORD OFF STACK
4175 034572  000177  000412                           JMP     @DOWELP         ;JUMP TO LOCATION IN DOWELP
4176 034576  013703  035212                5$:      MOV     PCSMPS,R3       ;PUT PRE-CSM PSW IN R3
4177 034602  042703  037777                           BIC     #37777,R3       ;WIPE OUT ALL BITS EXCEPT <15:14>,
4178 034606  000241                                   CLC                     ;CLEAR THE CARRY BIT
4179 034610  006003                                   ROR     R3              ;MOVE <15:14> OVER TO
4180 034612  006003                                   ROR     R3              ;THE RIGHT TO <13:12>,
4181 034614  052703  040000                           BIS     #BIT14,R3       ;AND SET <14>, PUTTING CURRENT=SUPERVISOR IN EXPECTED LOC
4182 034620  013737  177776  035226                   MOV     PSW,ACSMPS      ;MOVE CURRENT PSW TO ACSMPS
4183 034626  042737  007777  035226                   BIC     #7777,ACSMPS    ;WIPE OUT ALL BITS BUT <15:12>
4184 034634  020337  035226                           CMP     R3,ACSMPS       ;SEE IF PSW STATUS BITS MATCHES CALCULATED VALUE
4185 034640  001403                                   BEQ     6$              ;BRANCH AROUND ERROR CALL IF OK
4186 034642  005037  177572                           CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4187 034646  104042                                   ERROR   +42             ;NOT SUPERVISOR MODE
4188 034650  052737  000001  177572  6$:      BIS     #BIT00,MMR0     ;MAKE SURE MEMORY MANAGEMENT IS ON
4189 034656  013737  177776  035226                   MOV     PSW,ACSMPS      ;MOVE CURRENT PSW TO ACSMPS
4190 034664  042737  037777  035226                   BIC     #37777,ACSMPS   ;CLEAR ALL BITS EXCEPT <15:14>
4191 034672  023737  035232  035226                   CMP     SUPERM,ACSMPS   ;SEE IF SUPERVISOR MODE
4192 034700  001414                                   BEQ     8$              ;BRANCH AROUND ERROR CALL IF OK
4193 034702  012737  034650  035210                   MOV     #6$,DOWELP      ;SET ADDRESS FOR THIS CHECK TO DOWELP
4194 034710  005037  177572                           CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4195 034714  013703  035226                           MOV     ACSMPS,R3       ;MOVE RECEIVED CONTENTS TO R3
4196 034720  052703  040000                           BIS     #BIT14,R3       ;SET BIT 14 AND
4197 034724  042703  100000                           BIC     #BIT15,R3       ;CLEAR BIT 15
4198 034730  104042                                   ERROR   +42             ;NOT SUPERVISOR MODE
4199 034732  052737  000001  177572  8$:      BIS     #BIT00,MMR0     ;MAKE SURE MEMORY MANAGEMENT IS ON
4200 034740  013704  177776                           MOV     PSW,R4          ;MOVE CURRENT PSW TO R4
4201 034744  042704  147777                           BIC     #147777,R4      ;CLEAR ALL BITS EXCEPT <13:12>
4202 034750  006304                                   ASL     R4              ;MOVE <13:12> BITS IN R4
4203 034752  006304                                   ASL     R4              ;TO THE <15:14> POSITION
4204 034754  013702  035212                           MOV     PCSMPS,R2       ;MOVE PRECSM PSW TO R2
4205 034760  042702  037777                           BIC     #37777,R2       ;CLEAR BITS 13 TO 0
```

```
4206 034764  020204                          CMP     R2,R4           ;SEE IF PREVIOUS MODE IS OK
4207 034766  001406                          BEQ     10$             ;BRANCH AROUND ERROR CALL IF OK
4208 034770  012737  034732  035210          MOV     #8$,DOWELP      ;SET ADDRESS FOR THIS CHECK TO DOWELP
4209 034776  005037  177572                  CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4210 035002  104043                          ERROR   +43             ;WRONG PREVIOUS MODE
4211 035004  052737  000001  177572  10$:    BIS     #BIT00,MMR0     ;MAKE SURE MEMORY MANAGEMENT IS ON
4212 035012  023737  035230  035214          CMP     ACSMSP,PCSMSP   ;SEE IF STACK POINTER VALUE WAS TRANSFERED
4213 035020  001406                          BEQ     12$             ;BRANCH AROUND ERROR CALL IF SO
4214 035022  012737  035004  035210          MOV     #10$,DOWELP     ;SET ADDRESS FOR THIS CHECK TO DOWELP
4215 035030  005037  177572                  CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4216 035034  104044                          ERROR   +44             ;INCORRECT STACK
4217 035036  052737  000001  177572  12$:    BIS     #BIT00,MMR0     ;MAKE SURE MEMORY MANAGEMENT IS ON
4218 035044  020037  035220                  CMP     R0,CSM1ST       ;COMPARE R0 WITH THE ARGUMENT THAT WAS ON STACK
4219 035050  001406                          BEQ     14$             ;IF EQUAL, BRANCH AROUND ERROR
4220 035052  012737  035036  035210          MOV     #12$,DOWELP     ;SET ADDRESS FOR THIS CHECK TO DOWELP
4221 035060  005037  177572                  CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4222 035064  104045                          ERROR   +45             ;INCORRECT ARGUMENT
4223 035066  052737  000001  177572  14$:    BIS     #BIT00,MMR0     ;MAKE SURE MEMORY MANAGEMENT IS ON
4224 035074  022737  034514  035222          CMP     #2$,CSM2ND      ;SEE IF UPDATED PC WAS CORRECT
4225 035102  001411                          BEQ     16$             ;BRANCH AROUND ERROR IF OK
4226 035104  012737  035066  035210          MOV     #14$,DOWELP     ;SET ADDRESS FOR THIS CHECK TO DOWELP
4227 035112  005037  177572                  CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4228 035116  012737  034514  001176          MOV     #2$,$TMP0       ;MOVE EXPECTED UPDATED PC TO $TMP0
4229 035124  104046                          ERROR   +46             ;WRONG PC
4230 035126  052737  000001  177572  16$:    BIS     #BIT00,MMR0     ;MAKE SURE MEMORY MANAGEMENT IS ON
4231 035134  032737  000017  035224          BIT     #17,CSM3RD      ;SEE IF PSW <3:0> WERE CLEARED
4232 035142  001406                          BEQ     18$             ;BRANCH AROUND ERROR CALL IF OK
4233 035144  012737  035126  035210          MOV     #16$,DOWELP     ;SET ADDRESS FOR THIS CHECK TO DOWELP
4234 035152  005037  177572                  CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4235 035156  104047                          ERROR   +47             ;BITS <3:0> SET IN PSW
4236 035160  022737  034576  035210  18$:    CMP     #5$,DOWELP      ;SEE IF ANY ERRORS THIS TIME AROUND
4237 035166  001406                          BEQ     19$             ;BRANCH TO NORMAL RETURN JUMP IF NOT
4238 035170  032777  001000  143742          BIT     #BIT09,@SWR     ;SEE IF LOOP ON ERROR IS SET
4239 035176  001402                          BEQ     19$             ;BRANCH IF SO
4240 035200  000137  034506                  JMP     1$              ;JUMP BACK FOR LOOP ON ERROR
4241 035204  000177  000024          19$:    JMP     @RETURN         ;JUMP BACK TO RETURN ADDRESS IN LOCATION RETURN
4242
4243 035210  000000                  DOWELP: .WORD   0               ;LOCATION HOLDING ADDRESS TO SHORTEN POSSIBLE ERROR LOOP
4244 035212  000000                  PCSMPS: .WORD   0               ;LOCATION TO STORE PRE-CSM PSW
4245 035214  000000                  PCSMSP: .WORD   0               ;LOCATION TO STORE PRE-CSM STACK POINTER
4246 035216  000000                  PCSMR0: .WORD   0               ;LOCATION TO STORE PRE-CSM R0 VALUE
4247 035220  000000                  CSM1ST: .WORD   0               ;LOCATION TO STORE 1ST WORD POPPED AFTER CSM EXECUTION
4248 035222  000000                  CSM2ND: .WORD   0               ;LOCATION TO STORE 2ND WORD POPPED AFTER CSM EXECUTION
4249 035224  000000                  CSM3RD: .WORD   0               ;LOCATION TO STORE 3RD WORD POPPED AFTER CSM EXECUTION
4250 035226  000000                  ACSMPS: .WORD   0               ;LOCATION TO STORE POST-CSM PSW
4251 035230  000000                  ACSMSP: .WORD   0               ;LOCATION TO STORE POST-CSM STACK POINTER
4252 035232  040000                  SUPERM: .WORD   40000           ;LOCATION TO STORE VALUE OF PRESENT MODE=SUPERVISOR
4253 035234  000000                  RETURN: .WORD   0               ;LOCATION TO STORE RETURN ADDRESS OF THE CSMSUB
```

```
4262                                    .SBTTL  TEST # 42 - TEST CSM INSTRUCTION - D-SPACE SUPERVISOR MODE
                                        ;*****************************************************************
                                        ;*TEST 42       TEST CSM INSTRUCTION - D-SPACE SUPERVISOR MODE
                                        ;*****************************************************************
                                        ;*      THIS TEST CHECKS OUT THE CSM (CALL SUPERVISOR MODE)
                                        ;*      INSTRUCTION TO MAKE SURE CSM IS LEGAL WHEN ENABLED
                                        ;*      IN SUPERVISOR MODE, AND FOR A CHECK OF THE PARAMETERS
                                        ;*      PUSHED ON THE STACK.
                                        ;*****************************************************************
                                        ;*****************************************************************
    035236  000004                      TST42:  SCOPE
4263 035240  012737  000032  172516             MOV     #32,MMR3        ;ENABLE 22-BIT, CSM, SUPERVISOR
4264 035246  012737  040000  035212             MOV     #40000,PCSMPS   ;LOAD PRECSM LOCATION
4265 035254  052737  040000  177776             BIS     #BIT14,PSW      ;GO TO SUPERVISOR MODE
4266 035262  012737  000600  172240             MOV     #600,SIPAR0     ;IF WE GO TO I-SPACE, FLAG IT AS AN ERROR
4267 035270  012706  000700                     MOV     #SUPSTK,SSP     ;SETUP SUPERVISOR STACK POINTER
4268 035274  004737  034422                     JSR     PC,CSMSUB       ;TEST CSM INSTRUCTION SUBROUTINE
4269 035300  060000                             .WORD   60000           ; 60000 ADDED TO ADDRESS OF TRAP VECTOR
4270 035302  005037  177776                     CLR     PSW             ;RETURN TO KERNEL MODE
4271 035306  005037  172240                     CLR     SIPAR0          ;RESET SIPAR0 TO FIRST 4K
```

```
 4280                                   .SBTTL  TEST # 43 - TEST CSM INSTRUCTION - I-SPACE USER MODE
                                        ;;*****************************************************************
                                        ;*TEST 43       TEST CSM INSTRUCTION - I-SPACE USER MODE
                                        ;;*****************************************************************
                                        ;*      THIS TEST CHECKS OUT THE CSM (CALL SUPERVISOR MODE)
                                        ;*      INSTRUCTION TO MAKE SURE CSM IS LEGAL WHEN ENABLED
                                        ;*      IN USER MODE, AND FOR A CHECK OF THE PARAMETERS
                                        ;*      PUSHED ON THE STACK.
                                        ;;*****************************************************************
                                        ;;*****************************************************************
     035312  000004                     TST43:  SCOPE
 4281 035314  012737  000031  172516            MOV     #31,MMR3        ;ENABLE 22-BIT, CSM USER
 4282 035322  012737  140000  035212            MOV     #140000,PCSMPS  ;LOAD PRECSM LOCATION
 4283 035330  052737  140000  177776            BIS     #140000,PSW     ;GO TO USER MODE
 4284 035336  012737  000600  177660            MOV     #600,UDPAR0     ;IF WE GO TO D-SPACE, FLAG AS AN ERROR
 4285 035344  012706  000600                    MOV     #USESTK,SSP     ;SETUP USER STACK POINTER
 4286 035350  004737  034422                    JSR     PC,CSMSUB       ;TEST CSM INSTRUCTION SUBROUTINE
 4287 035354  000000                            .WORD   0               ; 0 ADDED TO ADDRESS OF TRAP VECTOR
 4288 035356  005037  177776                    CLR     PSW             ;RETURN TO KERNEL MODE
 4289 035362  005037  177660                    CLR     UDPAR0          ;RESET UDPAR0 TO FIRST 4K
```

```
 4298                                   .SBTTL  TEST # 44 - TEST CSM INSTRUCTION - D-SPACE USER MODE
                                        ;;*****************************************************************
                                        ;*TEST 44       TEST CSM INSTRUCTION - D-SPACE USER MODE
                                        ;;*****************************************************************
                                        ;*      THIS TEST CHECKS OUT THE CSM (CALL SUPERVISOR MODE)
                                        ;*      INSTRUCTION TO MAKE SURE CSM IS LEGAL WHEN ENABLED
                                        ;*      IN USER MODE, AND FOR A CHECK OF THE PARAMETERS
                                        ;*      PUSHED ON THE STACK.
                                        ;;*****************************************************************
                                        ;;*****************************************************************
      035366  000004                    TST44:  SCOPE
 4299 035370  012737  000031  172516            MOV     #31,MMR3        ;ENABLE 22-BIT, CSM USER
 4300 035376  012737  140000  035212            MOV     #140000,PCSMPS  ;LOAD PRECSM LOCATION
 4301 035404  012737  000600  177640            MOV     #600,UIPAR0     ;IF WE GO TO I-SPACE, FLAG AS AN ERROR
 4302 035412  012706  000600                    MOV     #USESTK,SSP     ;SETUP USER STACK POINTER
 4303 035416  004737  034422                    JSR     PC,CSMSUB       ;TEST CSM INSTRUCTION SUBROUTINE
 4304 035422  000000                            .WORD   0               ; 0 ADDED TO ADDRESS OF TRAP VECTOR
 4305 035424  005037  177776                    CLR     PSW             ;RETURN TO KERNEL MODE
 4306 035430  005037  177640                    CLR     UIPAR0          ;RESET UIPAR0 TO FIRST 4K
```

```
4312                                    .SBTTL  TEST # 45 - TEST CSM INSTRUCTION - MODE TESTS
                                        ;;*****************************************************************
                                        ;*TEST 45       TEST CSM INSTRUCTION - MODE TESTS
                                        ;*      THIS TEST EXECUTES THE CSM INSTRUCTION LEGALLY IN MODES 1 THROUGH 7,
                                        ;*      CHECKING FOR PROPER REGISTER INCREMENT/DECREMENT AND THAT ARGUMENT
                                        ;*      CONTENTS IS PROPERLY ON THE STACK
                                        ;;*****************************************************************
    035434  000004                      TST45:  SCOPE
4313 035436  012737  040000  177776             MOV     #40000,PSW      ;GO TO SUPERVISOR MODE
4314 035444  012737  000032  172516             MOV     #32,MMR3        ;ENABLE 22-BIT, CSM, SUPERVISOR
4315 035452  012701  000007                     MOV     #7,R1           ;DO 7 MODES
4316 035456  012702  035636                     MOV     #CSMINS,R2      ;LOAD CSM INSTRUCTION POINTER INTO R2
4317 035462  012703  035662                     MOV     #REGCHG,R3      ;LOAD REGISTER CHANGE TABLE POINTER TO R3
4318 035466  012704  035700                     MOV     #REGDAT,R4      ;LOAD ARGUMENT EXPECTED STACK POINTER TO R4
4319 035472  012737  035514  000010             MOV     #3$,RESVEC      ;SET RETURNS TO 3$
4320 035500  011237  035510              1$:    MOV     (R2),2$         ;MOVE CSM INSTRUCTION TO TEST LOCATION
4321 035504  012700  035656                     MOV     #REGLAB,R0      ;RESET R0
4322 035510  000000  000000              2$:    .WORD   0,0             ;1ST WORD FOR CSM INSTRUCTION, 2ND FOR MODES 6 & 7
4323 035514  011637  001200              3$:    MOV     (SP),$TMP1      ;MOVE ARGUMENT WORD TO $TMP1
4324 035520  021300                             CMP     (R3),R0         ;SEE IF REGISTER CHANGED PROPERLY
4325 035522  001407                             BEQ     4$              ;BRANCH IF OK
4326 035524  062706  000006                     ADD     #6,SP           ;POP STACK FOR POSSIBLE ERROR LOOPING
4327 035530  011337  001176                     MOV     (R3),$TMP0      ;MOVE EXPECTED DATA TO $TMP0
4328 035534  104052                             ERROR   +52             ;CSM FAILED TO INCREMENT/DECREMENT REGISTER PROPERLY
4329 035536  162706  000006                     SUB     #6,SP           ;RESTORE STACK POINTER TO PRE-ERROR STATE
4330 035542  021437  001200              4$:    CMP     (R4),$TMP1      ;SEE IF CORRECT ARGUMENT WAS LOADED
4331 035546  001417                             BEQ     5$              ;BRANCH IF OK
4332 035550  012637  001204                     MOV     (SP)+,$TMP3     ;POP STACK FOR POSSIBLE ERROR LOOPING
4333 035554  012637  001206                     MOV     (SP)+,$TMP4
4334 035560  012637  001210                     MOV     (SP)+,$TMP5
4335 035564  011437  001202                     MOV     (R4),$TMP2      ;MOVE EXPECTED DATA TO $TMP2
4336 035570  104053                             ERROR   +53             ;CSM FAILED TO PUT PROPER ARGUMENT ON STACK
4337 035572  013746  001210                     MOV     $TMP5,-(SP)     ;RESTORE STACK - NO ERROR LOOPING
4338 035576  013746  001206                     MOV     $TMP4,-(SP)
4339 035602  013746  001204                     MOV     $TMP3,-(SP)
4340 035606  022223                      5$:    CMP     (R2)+,(R3)+     ;INCREMENT R2 AND R3 TO NEXT LOCATIONS
4341 035610  005724                             TST     (R4)+           ;INCREMENT R4 TO NEXT LOCATION
4342 035612  062706  000006                     ADD     #6,SP           ;POP STACK OF CSM PUSHES
4343 035616  077150                             SOB     R1,1$           ;SUBTRACT 1 AND BRANCH IF NOT ALL MODES CHECKED
4344 035620  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4345 035624  005037  172516                     CLR     MMR3            ;CLEAR MMR3
4346 035630  005037  177776                     CLR     PSW             ;RETURN MODE BACK TO KERNEL
4347 035634  000430                             BR      $EOP            ;SKIP OVER TABLES AND LOCATIONS
4348 035636  007010  007020  007030  CSMINS:    .WORD   CSM1,CSM2,CSM3,CSM4,CSM5,CSM6,CSM7
4349 035654  035660                             .WORD   REGLAB+2
4350 035656  035654                      REGLAB: .WORD   REGLAB-2
4351 035660  052525                             .WORD   52525
4352 035662  035656  035660  035660  REGCHG:    .WORD   REGLAB,REGLAB+2,REGLAB+2,REGLAB-2,REGLAB-2,REGLAB,REGLAB
4353 035700  035654  035654  035660  REGDAT:    .WORD   REGLAB-2,REGLAB-2,REGLAB+2,REGLAB+2,52525,REGLAB-2,REGLAB+2
```

END OF PASS ROUTINE

```
 4354                                   .SBTTL  END OF PASS ROUTINE
                                        ;;*****************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY''
                                        ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
                                        ;*IF SW12=1 INHIBIT TRACE TRAP
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO LOOP
035716                                  $EOP:
035716  000004                                  SCOPE
035720  005037  001102                          CLR     $TSTNM          ;;ZERO THE TEST NUMBER
035724  005237  001232                          INC     $PASS           ;;INCREMENT THE PASS NUMBER
035730  042737  100000  001232                  BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
035736  005327                                  DEC     (PC)+           ;;LOOP?
035740  000001                          $EOPCT: .WORD   1
035742  003072                                  BGT     $DOAGN          ;;YES
035744  012737                                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
035746  000001                          $ENDCT: .WORD   1
035750  035740                                  $EOPCT
035752  104401  035760                          TYPE    ,65$            ;;TYPE ASCIZ STRING
035756  000407                                  BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  <12><15>/END PASS #/
035776                                  64$:
035776  013746  001232                          MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                                                        ;;TYPE PASS NUMBER
036002  104405                                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
036004  104401  036012                          TYPE    ,67$            ;;TYPE ASCIZ STRING
036010  000421                                  BR      66$             ;;GET OVER THE ASCIZ
                                        ;;67$:  .ASCIZ  /  TOTAL ERRORS SINCE LAST REPORT /
036054                                  66$:
036054  013746  001112                          MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
                                                                        ;;TOTAL NUMBER OF ERRORS
036060  104405                                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
036062  104401  001221                          TYPE    ,$CRLF          ;;TYPE CARRIAGE RETURN, LINE FEED
036066  005037  001112                          CLR     $ERTTL          ;;CLEAR ERROR TOTAL
036072  013700  000042                  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
036076  001414                                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
036100  005046                                  CLR     -(SP)           ;;INSURE THE 'T' BIT IS CLEAR
036102  012746  036110                          MOV     #$CLR.T,-(SP)   ;;SETUP FOR AN RTI OR RTT
036106  000426                                  BR      $RTRN           ;;GO DO AN RTI OR RTT TO LOAD THE PSW
                                                                        ;;WITH A CLEARED 'T' BIT
036110                                  $CLR.T:
036110  013700  000042                          MOV     @#42,R0         ;;INSURE R0 CONTAINS THE MONITORS
036114  001405                                  BEQ     $DOAGN          ;;RETURN ADDRESS
036116  000005                                  RESET                   ;;CLEAR THE WORLD
036120  004710                          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
036122  000240                                  NOP                     ;;SAVE ROOM
036124  000240                                  NOP                     ;;FOR
036126  000240                                  NOP                     ;;ACT11
036130                                  $DOAGN:
036130  104400                                  TRAP                    ;;PUSH OLD PSW AND PC ON STACK
036132  042716  000020                          BIC     #20,(SP)        ;;CLEAR THE 'T' BIT
036136  032777  010000  142774                  BIT     #BIT12,@SWR     ;;RUN WITH TRACE TRAP?
036144  001005                                  BNE     1$              ;;BR IF NO
036146  005137  001310                          COM     $TBIT           ;;IS IT TIME FOR TRACE TRAP
036152  100402                                  BMI     1$              ;;BR IF NO
036154  052716  000020                          BIS     #20,(SP)        ;;SET TRACE TRAP
```

END OF PASS ROUTINE

```
        036160  012746  036166          1$:     MOV     #$LOOP,-(SP)    ;;JUMP TO START OF TEST
        036164  000002                   $RTRN:  RTI                     ;;RETURN--THIS IS CHANGED TO
                                                                         ;;AN 'RTT' IF 'RTT' IS A LEGAL
                                                                         ;;INSTRUCTION
        036166                           $LOOP:
        036166  000137                           JMP     @(PC)+          ;;RETURN
        036170  020456                   $RTNAD: .WORD   LOOP
        036172     377      377     000  $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
                                                 .EVEN
   4355         000001                           .END
```

SYMBOL TABLE

```
ABASE = 000000        BIT00 = 000001        DH14   013026        DT55   015574        KDPAR4= 172370
ACDW1 = 000000        BIT01 = 000002        DH15   013106        DT57   015606        KDPAR5= 172372
ACDW2 = 000000        BIT02 = 000004        DH16   013146        EMTVEC= 000030       KDPAR6= 172374
ACPUOP= 000000        BIT03 = 000010        DH17   013226        EM1    007512        KDPAR7= 172376
ACSMPS  035226        BIT04 = 000020        DH2    012536        EM10   007621        KDPDR0= 172320
ACSMSP  035230        BIT05 = 000040        DH20   013306        EM11   007671        KDPDR1= 172322
ADDW0 = 000000        BIT06 = 000100        DH21   013346        EM12   007740        KDPDR2= 172324
ADDW1 = 000000        BIT07 = 000200        DH22   013463        EM13   010012        KDPDR3= 172326
ADDW10= 000000        BIT08 = 000400        DH24   013523        EM14   010063        KDPDR4= 172330
ADDW11= 000000        BIT09 = 001000        DH26   013563        EM15   010144        KDPDR5= 172332
ADDW12= 000000        BIT1  = 000002        DH27   013675        EM16   010227        KDPDR6= 172334
ADDW13= 000000        BIT10 = 002000        DH30   014072        EM2    007552        KDPDR7= 172336
ADDW14= 000000        BIT11 = 004000        DH32   014146        EM21   010305        KERSTK= 001100
ADDW15= 000000        BIT12 = 010000        DH33   014165        EM22   010352        KIPAR0= 172340
ADDW2 = 000000        BIT13 = 020000        DH34   014225        EM23   010421        KIPAR1= 172342
ADDW3 = 000000        BIT14 = 040000        DH36   014316        EM24   010454        KIPAR2= 172344
ADDW4 = 000000        BIT15 = 100000        DH37   014356        EM25   010513        KIPAR3= 172346
ADDW5 = 000000        BIT2  = 000004        DH40   014416        EM26   010560        KIPAR4= 172350
ADDW6 = 000000        BIT3  = 000010        DH41   014467        EM27   010626        KIPAR5= 172352
ADDW7 = 000000        BIT4  = 000020        DH42   014573        EM30   010703        KIPAR6= 172354
ADDW8 = 000000        BIT5  = 000040        DH43   014633        EM31   010746        KIPAR7= 172356
ADDW9 = 000000        BIT6  = 000100        DH44   014703        EM32   011011        KIPDR0= 172300
ADEVCT= 000000        BIT7  = 000200        DH45   014743        EM33   011047        KIPDR1= 172302
ADEVM = 000000        BIT8  = 000400        DH55   014773        EM34   011125        KIPDR2= 172304
AENV  = 000000        BIT9  = 001000        DH57   015032        EM35   011206        KIPDR3= 172306
AENVM = 000000        BPTVEC= 000014        DISPLA 001142        EM36   011242        KIPDR4= 172310
AFATAL= 000000        CKSWR = 104410        DISPRE 000174        EM37   011317        KIPDR5= 172312
AMADR1= 000000        CMSG    005351        DOWELP 035210        EM40   011362        KIPDR6= 172314
AMADR2= 000000        CNTRLC  005272        DSWR  = 177570       EM41   011447        KIPDR7= 172316
AMADR3= 000000        CPSAVE  003256        DT1    015072        EM42   011513        KSP   =%000006
AMADR4= 000000        CPUERR= 177766        DT10   015130        EM43   011556        LF    = 000012
AMAMS1= 000000        CR    = 000015        DT12   015142        EM44   011623        LOOP    020456
AMAMS2= 000000        CRLF  = 000200        DT13   015160        EM45   011671        MFPDLD  002774
AMAMS3= 000000        CSMINS  035636        DT14   015176        EM46   011730        MFPDLP  002760
AMAMS4= 000000        CSMSUB  034422        DT15   015214        EM47   011772        MFPDPS  002762
AMSGAD= 000000        CSM0  = 007000        DT16   015226        EM50   012031        MFPDTS  002736
AMSGLG= 000000        CSM1  = 007010        DT17   015244        EM51   012060        MFPDVC  003030
AMSGTY= 000000        CSM1ST  035220        DT2    015110        EM52   012116        MFPDV1  033456
AMTYP1= 000000        CSM2  = 007020        DT20   015262        EM53   012142        MFPILD  002522
AMTYP2= 000000        CSM2ND  035222        DT21   015274        EM54   012207        MFPILP  002506
AMTYP3= 000000        CSM3  = 007030        DT22   015312        EM55   012253        MFPIPS  002510
AMTYP4= 000000        CSM3RD  035224        DT24   015324        EM56   012317        MFPITS  002454
APASS = 000000        CSM4  = 007040        DT26   015336        EM57   012403        MFPIVC  002612
APRINI  002050        CSM5  = 007050        DT27   015354        ERLOOP  002614       MFPIV1  027460
APRIOR= 000000        CSM6  = 007060        DT30   015372        ERPREP  002434       MFPIV2  030056
APTCSU= 000040        CSM7  = 007070        DT32   015404        ERROR = 104000       MFPIV3  031566
APTENV= 000001        DDISP = 177570        DT34   015412        ERRTYP  003640       MFPIV4  032176
APTSIZ= 000200        DF1     015620        DT36   015426        ERRVEC= 000004       MFPIV5  032454
APTSPO= 000100        DF12    015643        DT40   015440        GTSWR = 104407       MFPIV6  033730
ASWREG= 000000        DF2     015625        DT41   015450        HT    = 000011       MGMERR  015726
ATESTN= 000000        DF3     015634        DT42   015464        IBSAVE  003636       MGMFLG  015730
AUNIT = 000000        DF32    015651        DT43   015476        INIT    020564       MMR0  = 177572
AUSWR = 000000        DF5     015640        DT45   015512        IOTVEC= 000020       MMR1  = 177574
AVECT1= 000000        DH1     012456        DT46   015526        KDPAR0= 172360       MMR2  = 177576
AVECT2= 000000        DH10    012626        DT47   015540        KDPAR1= 172362       MMR3  = 172516
BADPC   001304        DH12    012666        DT50   015552        KDPAR2= 172364       MMVEC = 000250
BIT0  - 000001        DH13    012746        DT52   015564        KDPAR3- 172366       MTPILD  002666
```

SYMBOL TABLE

```
MTPILP  002644
MTPIPM  002646
MTPITA  002710
MTPITS  002616
MTPIVC  002734
MTPIV1  030506
MTPIV2  031152
MTPIV3  032714
MTPIV4  033134
NDFLAG  002232
NODSPA  002222
PATCH   016016
PBAHI   001302
PBALO   001300
PCSMPS  035212
PCSMR0  035216
PCSMSP  035214
PFECDF  004272
PFECDH  004232
PFECDT  004262
PFECEM  004172
PFECWS  004162
PIRQ  - 177772
PIRQVE= 000240
PR0   = 000000
PR1   = 000040
PR2   = 000100
PR3   = 000140
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
PWRMSG  007450
PWRVEC= 000024
RDCHR = 104411
RDLIN = 104412
REGCHG  035662
REGDAT  035700
REGLAB  035656
RESREG- 104414
RESVEC= 000010
RETURN  035234
R6    =%000006
R7    =%000007
SAVREG= 104413
SCOPE = 000004
SDPAR0= 172260
SDPAR1= 172262
SDPAR2= 172264
SDPAR3= 172266
SDPAR4= 172270
SDPAR5= 172272
SDPAR6= 172274
SDPAR7= 172276
SDPDR0= 172220
SDPDR1= 172222
SDPDR2= 172224
SDPDR3= 172226
SDPDR4= 172230
SDPDR5= 172232
SDPDR6= 172234
SDPDR7= 172236
SIPAR0= 172240
SIPAR1= 172242
SIPAR2= 172244
SIPAR3= 172246
SIPAR4= 172250
SIPAR5= 172252
SIPAR6= 172254
SIPAR7= 172256
SIPDR0= 172200
SIPDR1= 172202
SIPDR2= 172204
SIPDR3= 172206
SIPDR4= 172210
SIPDR5= 172212
SIPDR6= 172214
SIPDR7= 172216
SR0   = 177572
SR1   = 177574
SR2   = 177576
SR3   = 172516
SSP   =%000006
STACK = 001100
START   020000
STKLMT= 177774
SUPERM  035232
SUPSTK= 000700
SWR     001140
SWREG   000176
SW0   = 000001
SW00  = 000001
SW01  = 000002
SW02  = 000004
SW03  = 000010
SW04  = 000020
SW05  = 000040
SW06  = 000100
SW07  = 000200
SW08  = 000400
SW09  = 001000
SW1   = 000002
SW10  = 002000
SW11  = 004000
SW12  = 010000
SW13  = 020000
SW14  = 040000
SW15  = 100000
SW2   = 000004
SW3   = 000010
SW4   = 000020
SW5   - 000040
SW6   = 000100
SW7   = 000200
SW8   = 000400
SW9   = 001000
TBIT  = 000020
TBITPS  001274
TBITVE= 000014
TESTNO  001254
TIMERR  015654
TIMFLG  015656
TKVEC = 000060
TOFF    002346
TON     002402
TPVEC = 000064
TRAPPC  001260
TRAPPS  001262
TRAPVE= 000034
TRTVEC= 000014
TST1    020602
TST10   023052
TST11   023354
TST12   023444
TST13   023634
TST14   024150
TST15   025126
TST16   026164
TST17   026434
TST2    021120
TST20   026710
TST21   027124
TST22   027526
TST23   030124
TST24   030554
TST25   031220
TST26   031634
TST27   032244
TST3    021354
TST30   032530
TST31   032734
TST32   033154
TST33   033224
TST34   033524
TST35   033776
TST36   034066
TST37   034160
TST4    021524
TST40   034252
TST41   034340
TST42   035236
TST43   035312
TST44   035366
TST45   035434
TST5    022020
TST6    022300
TST7    022460
TYPBN - 104406
TYPDS = 104405
TYPE  = 104401
TYPOC = 104402
TYPON = 104404
TYPOS = 104403
UDPAR0= 177660
UDPAR1= 177662
UDPAR2= 177664
UDPAR3= 177666
UDPAR4= 177670
UDPAR5= 177672
UDPAR6= 177674
UDPAR7= 177676
UDPDR0= 177620
UDPDR1= 177622
UDPDR2= 177624
UDPDR3= 177626
UDPDR4= 177630
UDPDR5= 177632
UDPDR6= 177634
UDPDR7= 177636
UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646
UIPAR4= 177650
UIPAR5= 177652
UIPAR6= 177654
UIPAR7= 177656
UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
USESTK= 000600
USP   =%000006
VIRT1   001276
WASR6   001256
WASSR0  001264
WASSR1  001266
WASSR2  001270
WASSR3  001272
WBIT  = 000100
$APTHD  000204
$ATYC   006012
$ATY1   005766
$ATY3   005774
$ATY4   006004
$AUTOB  001134
$BDADR  001122
$BDDAT  001126
$BELL   001214
$BIN    006306
$CHARC  005762
$CKSWR  004276
$CLR.T  036110
$CMTAG  001100
$CM1  = 000006
$CM2  = 000014
$CM3  = 000006
$CM4  = 000006
$CNTLC  001312
$CNTLG  005243
$CNTLU  005236
$CPUOP  001252
$CRLF   001221
$DBLK   006752
$DB20   007056
$DEVCT  001234
$DOAGN  036130
$DTBL   006742
$ENDAD  036120
$ENDCT  035746
$ENULL  036172
$ENV    001244
$ENVM   001245
$EOP    035716
$EOPCT  035740
$ERFLG  001103
$ERMAX  001115
$ERROR  003260
$ERRPC  001116
$ERRTB  001320
$ERTTL  001112
$ESCAP  001212
$ETABL  001244
$ETEND  001254
$FATAL  001226
$FFLG   006232
$FILLC  001156
$FILLS  001155
$GDADR  001120
$GDDAT  001124
$GET42  036072
$GTSWR  004346
$HD   = 000000
$HIBTS  000204
$ICNT   001104
$ILLUP  007442
$INTAG  001135
$ITEMB  001114
$LF     001222
$LFLG   006231
$LOOP   036166
$LPADR  001106
$LPERR  001110
$MAIL   001224
$MBADR  000206
$MFLG   006230
$MNEW   005261
$MSGAD  001240
$MSGLG  001242
```

SYMBOL TABLE

```
$MSGTY  001224          $RDCHR  004620          $SETUP= 000137          $TMP4   001206          $TYPE   005422
$MSWR   005250          $RDLIN  004750          $STUP = 177777          $TMP5   001210          $TYPEC  005634
$MXCNT  001306          $RDSZ = 000010          $SVLAD  003206          $TN   = 000046          $TYPEX  005764
$NULL   001154          $REGAD  001160          $SVPC = 000204          $TPB    001152          $TYPOC  006334
$NWTST= 000001          $REG0   001162          $SWR  = 173400          $TPFLG  001157          $TYPON  006350
$OCNT   006532          $REG1   001164          $SWREG  001246          $TPS    001150          $TYPOS  006310
$OCTVL  007160          $REG2   001166          $SWRMK= 000000          $TRAP   007176          $UNIT   001236
$OMODE  006534          $REG3   001170          $TBIT   001310          $TRAP2  007220          $UNITM  000214
$OVER   003242          $REG4   001172          $TESTN  001230          $TRP  = 000015          $USWR   001250
$PASS   001232          $REG5   001174          $TKB    001146          $TRPAD  007232          $XOFF = 000023
$PASTM  000212          $RESRE  007020          $TKS    001144          $TSTM   000210          $XON  = 000021
$PWRAD  007424          $RTNAD  036170          $TMP0   001176          $TSTNM  001102          $XTSTR  003044
$PWRDN  007264          $RTRN   036164          $TMP1   001200          $TTYIN  005226          $$GET4= 000001
$PWRMG  007420          $SAVRE  006762          $TMP2   001202          $TYPBN  006234          $OFILL  006533
$PWRUP  007336          $SAVR6  007446          $TMP3   001204          $TYPDS  006536          .$X   = 000204
$QUES   001220          $SCOPE  003032

. ABS.  036176      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  54848 WORDS  ( 215 PAGES)
DYNAMIC MEMORY:  20034 WORDS  ( 77 PAGES)
ELAPSED TIME:  00:05:20
CKKTBC.BIN,CKKTBC/CR/-SP/NL:TOC=CKKTBC.MLB/ML,CKKTBC.P11
```