```
1                                   .REM    @
2
3
4
5
6
7
8
9                                           IDENTIFICATION
10                                          --------------
11
12                                  PRODUCT CODE:   AC-F612D-MC
13
14                                  PRODUCT NAME:   CKKTBD0 11/44 MEM MGMT PRT B
15
16                                  DATE:           JANUARY, 1982
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, 1982 BY DIGITAL EQUIPMENT CORPORATION
```

```
43
44
45
46
47
48
49
50
51
52
53
54
55
```

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. MODIFIED CODE TO ACCOMODATE ECO # 8.
JANUARY, 1982	D	ECO # 8 WAS MODIFIED, CODE CHANGED TO REFLECT THE MODIFICATION.

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 THE LOGIC.
THE DIAGNOSTIC WILL PROVIDE ENOUGH INFORMATION SUCH THAT
BY DEDUCTION, THE FAILURE CAN BE ISOLATED TO A SMALL
SEGMENT OF THE MEMORY MANAGEMENT LOGIC.

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

1.2 REQUIREMENTS

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

1.3 RELATED DOCUMENTS AND STANDARDS

1. ACT11/XXDP PROGRAMMING SPECIFICATION
2. STANDARD APT SYSTEM TO A PDP11 DIAGNOSTIC INTERFACE
3. 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

THE PROGRAM IS STARTED BY LOADING ADDRESS 200 AND
STARTING. THE SWITCH REGISTER SHOULD BE SET ACCORDING TO
SECTION 2.3 BEFORE THE PROGRAM IS STARTED. HOWEVER, IF
DESIRED, THE PROGRAM WILL USE THE SOFTWARE SWITCH REGISTER
AT LOCATION 176 (LOCATION 174 WILL BE USED AS THE SOFTWARE
DISPLAY REGISTER). IN THAT CASE, THE PROGRAM WILL ASK FOR
THE INITIAL SWITCH REGISTER VALUE BY TYPING "SWR= XXXXXX
NEW= " AFTER TYPING THE NAME OF THE PROGRAM (XXXXXX =
THE OCTAL CONTENTS OF LOCATION 176). (SEE SECTION 2.4)

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

2.3 CONTROL SWITCH SETTINGS

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

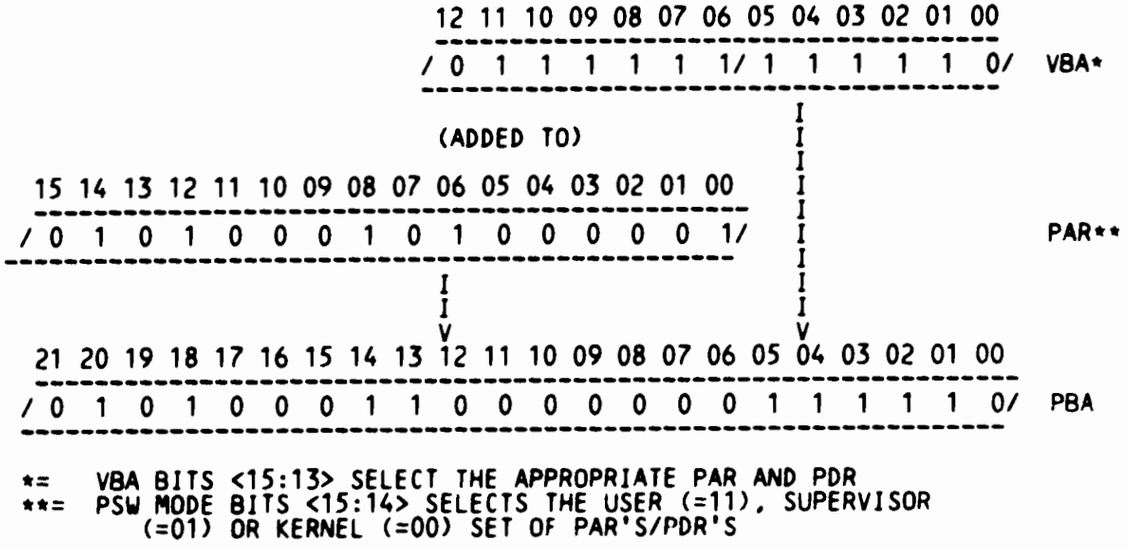
THE "T-BIT" (BIT 4) IN THE PROCESSOR STATUS WORD IS SET
BY AN "RTI" IN THE END-OF-PASS ROUTINE FOR EVERY OTHER PASS
BEGINNING WITH THE THIRD PASS (PASSES 3,5,7,9...). T-BIT
TRAPPING CAN BE INHIBITED BY SETTING BIT 12 = 1 IN THE SWITCH
REGISTER (SEE SECTION 2.4).

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".

@

```
1194
1195                            .TITLE  CKKTBD0 11/44 MEM MGMT PRT B
                                ;*COPYRIGHT (C) 1982
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.
                                ;*
1196                            .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>
1197                            .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
```

```
                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
   1198                         .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
```

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

```
                172352                  KIPAR5= 172352
                172354                  KIPAR6= 172354
                172356                  KIPAR7= 172356
                                        ;*KERNEL 'D' PAGE ADDRESS REGISTERS
                172360                  KDPAR0= 172360
                172362                  KDPAR1= 172362
                172364                  KDPAR2= 172364
                172366                  KDPAR3= 172366
                172370                  KDPAR4= 172370
                172372                  KDPAR5= 172372
                172374                  KDPAR6= 172374
                172376                  KDPAR7= 172376
   1199                                 ;*ADDITIONAL DEFINITIONS
   1200                                 ;*
   1201         177572                          MMR0=SR0
   1202         177574                          MMR1=SR1
   1203         177576                          MMR2=SR2
   1204         172516                          MMR3=SR3
   1205         000006                          KSP=SP
   1206         000006                          SSP=SP
   1207         000006                          USP=SP
   1208         000020                          TBIT=BIT4
   1209         000100                          WBIT=BIT6
   1210         177766                  CPUERR=177766
   1211         001100                  KERSTK=STACK
   1212         000700                  SUPSTK=STACK-200
   1213         000600                  USESTK=STACK-300
   1214
```

TRAP CATCHER

```
   1246
                                        .SBTTL  TRAP CATCHER
                000000                  .=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
```

```
                                .SBTTL  ACT11 HOOKS
                                ;;*****************************************************************
                                ;HOOKS REQUIRED BY ACT11
000204                                  $SVPC=.                 ;SAVE PC
000046                                  .=46
000046  036252                          $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
```

APT PARAMETER BLOCK

```
1248                                    .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)
```

```
1249                                    .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
```

ERROR POINTER TABLE

```
                                .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                     $ERRTB:
1250
1251 001320  007522             ;*ITEM 1
1252 001322  012672                     EM1                     ;UNEXPECTED CPU TRAP TO LOC. 004
1253 001324  015306                     DH1                     ;OLD PC  OLD PSW R6 WAS  CPUERR  TESTNO  ERRORPC
1254 001326  016057                     DT1                     ;TRAPPC,TRAPPS,WASR6,CPUERR,TESTNO,$ERPPC,0
1255                                    DF12                    ;0,0,0,0,0,0
1256
1257 001330  007562             ;*ITEM 2
1258 001332  012752                     EM2                     ;UNEXPECTED MEM. MGMT. TRAP TO LOC. 250
1259 001334  015324                     DH2                     ;OLD PC  OLD PSW R6 WAS  SR0     SR2     TESTNO  ERRORPC
1260 001336  016041                     DT2                     ;TRAPPC, TRAPPS, WASR6, WASSR0, WASSR2, TESTNO, $ERRPC, 0
1261                                    DF2                     ;0,0,0,0,0,0,0
1262
1263 001340  007631             ;*ITEM 3
1264 001342  013042                     EM10                    ;MEMORY MGMT. ACCESS ABORT DID NOT OCCUR
1265 001344  015344                     DH10                    ;PDR 4   PSW     TESTNO  ERRORPC
1266 001346  016050                     DT10                    ;$REG2,$TMP0,TESTNO,$ERRPC,0
1267                                    DF3                     ;0,0,0,0
1268
1269 001350  007701             ;*ITEM 4
1270 001352  013042                     EM11                    ;ACCESS ERROR DID NOT ABORT INSTRUCTION
1271 001354  015344                     DH10                    ;PDR 4   PSW     TESTNO  ERRORPC
1272 001356  016050                     DT10                    ;$REG2,$TMP0,TESTNO,$ERRPC,0
1273                                    DF3                     ;0,0,0,0
1274
1275 001360  007750             ;*ITEM 5
1276 001362  013102                     EM12                    ;SR0 DID NOT REPORT ACCESS ERROR CORRECTLY
1277 001364  015356                     DH12                    ;SR0 WAS EXPECTD PDR 4   PSW     TESTNO ERRORPC
1278 001366  016057                     DT12                    ;WASSR0,$REG3,$REG2,$TMP0,TESTNO,$ERRPC,0
1279                                    DF12                    ;0,0,0,0,0,0
1280
1281 001370  010022             ;*ITEM 6
1282 001372  013162                     EM13                    ;SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.
1283 001374  015374                     DH13                    ;SR2 WAS EXPECTD PDR 4   PSW     TESTNO ERRORPC
1284 001376  016057                     DT13                    ;WASSR2,$REG4,$REG2,$TMP0,TESTNO,$ERRPC,0
1285                                    DF12                    ;0,0,0,0,0,0
1286
1287 001400  010073             ;*ITEM 7
1288 001402  013242                     EM14                    ;PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE
1289 001404  015412                     DH14                    ;V.B.A.  KIPDR4  SR0 WAS SR2 WAS TESTNO  ERRORPC
1290 001406  016057                     DT14                    ;$REG0,$REG4,WASSR0,WASSR2,TESTNO,$ERRPC,0
                                        DF12                    ;0,0,0,0,0,0
```

ERROR POINTER TABLE

```
1291                                    ;*ITEM 10
1292 001410  010154                             EM15            ;PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE
1293 001412  013322                             DH15            ;V.B.A.  KIPDR4  TESTNO  ERRORPC
1294 001414  015430                             DT15            ;$REG0,$REG4,TESTNO,$ERRPC,0
1295 001416  016050                             DF3             ;0,0,0,0
1296
1297                                    ;*ITEM 11
1298 001420  010237                             EM16            ;SR0 DID NOT REPORT PAGE LGTH. ABORT CORRECTLY
1299 001422  013362                             DH16            ;V.B.A.  KIPDR4  SR0 WAS EXPECTD TESTNO  ERRORPC
1300 001424  015442                             DT16            ;$REG0,$REG4,WASSR0,$REG2,TESTNO,$ERRPC,0
1301 001426  016057                             DF12            ;0,0,0,0,0,0
1302
1303                                    ;*ITEM 12
1304 001430  010022                             EM13            ;SR2 DID NOT LOCKUP CORRECT VIRUAL ADDR.
1305 001432  013442                             DH17            ;V.B.A.  KIPDR4  SR2 WAS EXPECTD TESTNO  ERRORPC
1306 001434  015460                             DT17            ;$REG0,$REG4,WASSR2,$REG3,TESTNO,$ERRPC,0
1307 001436  016057                             DF12            ;0,0,0,0,0,0
1308
1309                                    ;*ITEM 13
1310 001440  010022                             EM13            ;SR2 DID NOT LOCKUP CORRECT VIRUAL ADDR.
1311 001442  013522                             DH20            ;SR2 WAS EXPECTD TESTNO  ERRORPC
1312 001444  015476                             DT20            ;WASSR2,$REG1,TESTNO,$ERRPC,0
1313 001446  016050                             DF3             ;0,0,0,0
1314
1315                                    ;*ITEM 14
1316 001450  010315                             EM21            ;SR0 OR SR2 CHANGED BY A SECOND ABORT
1317 001452  013562                             DH21            ;FIRST ABORT     SECOND ABORT
1318                                                            ;SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO  ERRORPC
1319 001454  015510                             DT21            ;$TMP0,$TMP2,WASSR0,WASSR2,TESTNO,$ERRPC,0
1320 001456  016057                             DF12            ;0,0,0,0,0,0
1321
1322                                    ;*ITEM 15
1323 001460  010362                             EM22            ;SR0 OR SR2 WAS NOT 'RESET' BY A RESET
1324 001462  013677                             DH22            ;SR0 WAS SR2 WAS TESTNO  ERRORPC
1325 001464  015526                             DT22            ;WASSR0,WASSR2,TESTNO,$ERRPC,0
1326 001466  016050                             DF3             ;0,0,0,0
1327
1328                                    ;*ITEM 16
1329 001470  010431                             EM23            ;SR2 NOT TRACKING CORRECTLY
1330 001472  013522                             DH20            ;SR2 WAS EXPECTD TESTNO  ERROPC
1331 001474  015476                             DT20            ;WASSR2,$REG1,TESTNO,$ERRPC,0
1332 001476  016050                             DF3             ;0,0,0,0
1333
1334                                    ;*ITEM 17
1335 001500  010464                             EM24            ;DID NOT TRAP THRU KERNEL SPACE
1336 001502  013737                             DH24            ;PSW WAS R6 WAS  TESTNO  ERRORPC
1337 001504  015540                             DT24            ;$REG1,$REG2,TESTNO,$ERRPC,0
1338 001506  016050                             DF3             ;0,0,0,0
```

ERROR POINTER TABLE

```
1339                                    ;*ITEM 20
1340 001510  010523                             EM25            ;KT ERROR SERVICED ON ODD ADDR. ERROR
1341 001512  013522                             DH20            ;PDR     TESTNO  ERRORPC
1342 001514  015476                             DT20            ;$REG5,TESTNO,$ERRPC,0
1343 001516  016054                             DF5             ;0,0,0
1344
1345                                    ;*ITEM 21
1346 001520  010570                             EM26            ;SR0 OR SR2 CHANGED BY ODD ADDR. ERROR
1347 001522  013777                             DH26            ;EXPECTED        RECEIVED
1348                                                             ;SR0     SR2     SR0 WAS SR2 WAS TESTNO  ERRORPC
1349 001524  015552                             DT26            ;$REG0,$REG1,WASSR0,WASSR2,TESTNO,$ERRPC,0
1350 001526  016057                             DF12            ;0,0,0,0,0,0
1351
1352                                    ;*ITEM 22
1353 001530  010636                             EM27            ;ERROR DURING "DOUBLE ERROR" (KT & ODD ADDR.)
1354 001532  014111                             DH27            ;EXPECTED:
1355                                                             ;PSW     PC       SR0     SR2
1356                                                             ;170017  (3$+4)  020147  (3$)
1357                                                             ;RECEIVED
1358                                                             ;PSW     PC       SR0     SR2     TESTNO  ERRORPC
1359 001534  015570                             DT27            ;$REG1,$REG3,WASSR0,WASSR2,TESTNO,$ERRPC,0
1360 001536  016057                             DF12            ;0,0,0,0,0,0
1361
1362                                    ;*ITEM 23
1363 001540  010713                             EM30            ;MFPI INSTRUCTION PUSHED WRONG DATA
1364 001542  014306                             DH30            ;DATA    DATA
1365                                                             ;EXPECTD RECEIVD TESTNO  ERRORPC
1366 001544  015606                             DT30            ;$REG0,$REG1,TESTNO,$ERRPC,0
1367 001546  016050                             DF3             ;0,0,0,0
1368
1369                                    ;*ITEM 24
1370 001550  010756                             EM31            ;MTPI INSTRUCTION LOADED WRONG DATA
1371 001552  014306                             DH30            ;DATA    DATA
1372                                                             ;EXPECTD RECEIVD TESTNO  ERRORPC
1373 001554  015606                             DT30            ;$REG0,$REG1,TESTNO,$ERRPC,0
1374 001556  016050                             DF3             ;0,0,0,0
1375
1376                                    ;*ITEM 25
1377 001560  011021                             EM32            ;STACK NOT PUSHED BY MFPI-MTPI
1378 001562  014362                             DH32            ;TESTNO  ERRORPC
1379 001564  015620                             DT32            ;TESTNO,$ERRPC,0
1380 001566  016065                             DF32            ;0,0
1381
1382                                    ;*ITEM 26
1383 001570  011057                             EM33            ;KERNEL PAGE ACCESSED INSTEAD OF USER: MFPI-MTPI
1384 001572  014401                             DH33            ;SR0 WAS SR2 WAS TESTNO  ERRORPC
1385 001574  015526                             DT22            ;WASSR0,WASSR2,TESTNO,$ERRPC,0
1386 001576  016050                             DF3             ;0,0,0,0
1387
1388                                    ;*ITEM 27
1389 001600  011135                             EM34            ;M.M. ABORT IN KERNAL D-SPACE HAD WRONG CONDITION
1390 001602  014441                             DH34            ;(MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC EXPECTING 020031
1391 001604  015626                             DT34            ;$REG1,$REG2,$REG3,TESTNO,$ERRPC,0
1392 001606  016034                             DF1             ;0,0,0,0,0
1393
1394                                    ;*ITEM 30
1395 001610  011216                             EM35            ;ILLEGAL MODE 10 NOT ABORTED
```

ERROR POINTER TABLE

```
1396 001612  014362                DH32          ;TESTNO  ERRORPC
1397 001614  01562U                DT32          ;TESTNO,$ERRPC,0
1398 001616  016065                DF32          ;0,0
```

ERROR POINTER TABLE

```
1399                                    ;*ITEM 31
1400 001620  011252                             EM36            ;SR0 DID NOT REPORT ILLEGAL MODE 10 CORRECTLY
1401 001622  014532                             DH36            ;SR0 WAS EXPECTD TESTNO  ERRORPC
1402 001624  015642                             DT36            ;WASSR0,$REG1,TESTNO,$ERRPC,0
1403 001626  016050                             DF3             ;0,0,0,0
1404
1405                                    ;*ITEM 32
1406 001630  011327                             EM37            ;PSW CHANGED BY AN RTI IN USER MODE
1407 001632  014572                             DH37            ;PSW WAS EXPECTD TESTNO  ERRORPC
1408 001634  015540                             DT24            ;$REG1,$REG2,TESTNO,$ERRPC,0
1409 001636  016050                             DF3             ;0,0,0,0
1410
1411                                    ;*ITEM 33
1412 001640  011372                             EM40            ;ABORT IN KERNAL D-SPACE PICKED UP VECTOR FROM I-SPACE
1413 001642  014632                             DH40            ;(PSW)   TESTNO  ERRORPC EXPECTING XXX340
1414 001644  015654                             DT40            ;$REG0,TESTNO,$ERRPC,0
1415 001646  016054                             DF5             ;0,0,0
1416
1417                                    ;*ITEM 34
1418 001650  011457                             EM41            ;D SPACE ENABLE CIRCUITRY HAS FAILED
1419 001652  014703                             DH41            ;ERROR   AUTOI/D VIRTUAL
1420                                                             ;REGISTR REGISTR ADDRESS TESTNO  PC AT ABORT
1421 001654  015664                             DT41            ;WASSR0,WASSR1,WASSR2,TESTNO,BADPC,0
1422 001656  016034                             DF1             ;0,0,0,0,0
1423
1424                                    ;*ITEM 35
1425 001660  011523                             EM42            ;INCORRECT STORE BY MTP INSTRUCTION
1426 001662  015007                             DH42            ;GDDATA  STORED  TESTNO  ERRORPC
1427 001664  015700                             DT42            ;$REG3,$REG4,TESTNO,$ERRPC,0
1428 001666  016050                             DF3             ;0,0,0,0
1429
1430                                    ;*ITEM 36
1431 001670  011566                             EM43            ;TRIED TO REFERENCE NON-RESIDENT PAGE
1432 001672  015047                             DH43            ;(MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC
1433 001674  015712                             DT43            ;$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
1434 001676  016034                             DF1
1435
1436                                    ;*ITEM 37
1437 001700  011633                             EM44            ;WRONG DATA FETCHED BY MFP INSTRUCTION
1438 001702  014306                             DH30            ;DATA    DATA
1439                                                             ;EXPECTD RECEIVD TESTNO  ERRORPC
1440 001704  015606                             DT30            ;$REG0,$REG1,TESTNO,$ERRPC,0
1441 001706  016050                             DF3             ;0,0,0,0
1442
1443                                    ;*ITEM 40
1444 001710  011566                             EM43            ;TRIED TO REFERENCE NON-RESIDENT PAGE
1445 001712  015047                             DH43            ;(MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC
1446 001714  015726                             DT45            ;WASSR0,WASSR1,WASSR2,TESTNO,$ERRPC,0
1447 001716  016034                             DF1             ;0,0,0,0,0
1448
1449                                    ;*ITEM 41
1450 001720  011701                             EM45            ;ILLEGAL CSM DID NOT TRAP TO 10
1451 001722  014362                             DH32            ;TESTNO  ERRORPC
1452 001724  015620                             DT32            ;TESTNO,$ERRPC,0
1453 001726  016041                             DF2             ;0,0
```

ERROR POINTER TABLE

```
1454
1455 001730  011740                    ;*ITEM 42
1456 001732  015117                            EM46            ;CSM DID NOT ENTER SUPERVISOR MODE
1457 001734  015742                            DH44            ;EXPECTD (PSW)   TESTNO  ERR PC
1458 001736  016034                            DT46            ;$REG3,ACSMPS,TESTNO,$ERRPC,0
1459                                           DF1             ;0,0,0,0,0
1460
1461 001740  012002                    ;*ITEM 43
1462 001742  015117                            EM47            ;CSM SET UP WRONG PREVIOUS MODE
1463 001744  015742                            DH44            ;EXPECTD (PSW)   TESTNO  ERR PC
1464 001746  016034                            DT46            ;$REG3,ACSMPS,TESTNO,$ERRPC,0
1465                                           DF1             ;0,0,0,0,0
1466
1467 001750  012041                    ;*ITEM 44
1468 001752  014306                            EM50            ;CSM SET UP STACK WRONG
1469                                           DH30            ;DATA    DATA
1470 001754  015606                                            ;EXPECTD RECEIVD TESTNO  ERR PC
1471 001756  016034                            DT30            ;ACSMSP,$TMP0,TESTNO,$ERRPC,0
1472                                           DF1             ;0,0,0,0,0
1473
1474 001760  012070                    ;*ITEM 45
1475 001762  014306                            EM51            ;CSM PUSHED INCORRECT ARGUMENT
1476                                           DH30            ;DATA    DATA
1477 001764  015754                                            ;EXPECTD RECEIVD TESTNO  ERR PC
1478 001766  016034                            DT47            ;$REG0,CSM1ST,TESTNO,$ERRPC,0
1479                                           DF1             ;0,0,0,0,0
1480
1481 001770  012126                    ;*ITEM 46
1482 001772  014306                            EM52            ;CSM PUSHED WRONG PC
1483                                           DH30            ;DATA    DATA
1484 001774  015766                                            ;EXPECTD RECEIVD TESTNO  ERR PC
1485 001776  016034                            DT50            ;$TMP0,CSM2ND,TESTNO,$ERRPC,0
1486                                           DF1             ;0,0,0,0,0
1487
1488 002000  012152                    ;*ITEM 47
1489 002002  015117                            EM53            ;CSM DID NOT CLEAR OLD PSW BITS <3:0>
1490 002004  016000                            DH44            ;OLDPSW  TESTNO  ERR PC
1491 002006  016050                            DT52            ;CSM3RD,TESTNO,$ERRPC,0
1492                                           DF3             ;0,0,0,0
1493
1494 002010  012217                    ;*ITEM 50
1495 002012  014362                            EM54            ;CSM ACCESSED WRONG SUPERVISOR SPACE
1496 002014  015620                            DH32            ;TESTNO  ERR PC
1497 002016  016054                            DT32            ;TESTNO,$ERRPC,0
1498                                           DF5             ;0,0,0
1499
1500 002020  012263                    ;*ITEM 51
1501 002022  014362                            EM55            ;CSM ABORTED WHEN IT SHOULD NOT HAVE
1502 002024  015620                            DH32            ;TESTNO  ERR PC
1503 002026  016054                            DT32            ;TESTNO,$ERRPC,0
1504                                           DF5             ;0,0,0
1505
1506 002030  012327                    ;*ITEM 52
1507 002032  015207                            EM56            ;CSM FAILED TO INCREMENT/DECREMENT REGISTER PROPERLY
1508 002034  016010                            DH55            ;TESTNO  ERR PC  R0 EXP  R0 RCV
1509 002036  016050                            DT55            ;TESTNO,$ERRPC,$TMP0,$REG0,0
                                               DF3             ;0,0,0,0
```

ERROR POINTER TABLE

```
1510                                    ;*ITEM 53
1511 002040  012413                             EM57            ;CSM FAILED TO PUT PROPER ARGUMENT ON STACK
1512 002042  015246                             DH57            ;TESTNO  ERR PC  ARGEXP  ARGRCV
1513 002044  016022                             DT57            ;TESTNO,$ERRPC,$TMP1,$TMP2,0
1514 002046  016050                             DF3             ;0,0,0,0
1515
1516                                    ;*ITEM 54
1517 002050  012466                             EM58            ;SR2 LOCKED UP AN 11/34 COMPATIBLE ADDRESS THAT
1518                                                            ;DOES NOT MAKE IT COMPATIBLE WITH AN 11/70 SINCE
1519                                                            ;THE OPTIONAL M7095 ECO #8 IS MISSING
1520 002052  013522                             DH20            ;SR2 WAS EXPECTD TESTNO  ERRORPC
1521 002054  015476                             DT20            ;WASSR2,$REG1,TESTNO,$ERRPC,0
1522 002056  016050                             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
                                ;*
                                ;;**************************************************************
002060                          APRINIT:
002060  012700  077406                MOV     #77406,R0       ;MAKE ALL PDR'S 4K, READ/WRITE, UPWARDS
                                                                ;EXPANDING, 200 BLOCKS
002064  012702  172300                MOV     #KIPDR0,R2      ;LOAD THE ADDRESS OF THE FIRST KERNAL PDR
002070  012701  000020        1$:     MOV     #20,R1          ;LOAD R1 WITH 16
002074  010022                2$:     MOV     R0,(R2)+        ;LOAD EACH PDR IN TURN
002076  077102                        SOB     R1,2$           ;LOOP UNTIL ALL ARE LOADED
002100  020227  172340                CMP     R2,#KDPDR7+2    ;HAVE WE LOADED ALL KERNAL PDR'S
002104  001003                        BNE     3$              ;BRANCH IF KERNAL & SUPER HAVE BEEN LOADED
002106  012702  172200                MOV     #SIPDR0,R2      ;LOAD ALL SUPERVISOR PDR'S
002112  000766                        BR      1$              ;BRANCH TO LOOP
002114  020227  172240        3$:     CMP     R2,#SDPDR7+2    ;HAVE USER PDR'S BEEN DONE
002120  001003                        BNE     4$              ;BRANCH IF THEY HAVE
002122  012702  177600                MOV     #UIPDR0,R2      ;LOAD ALL USER PDR'S
002126  000760                        BR      1$              ;BRANCH TO LOOP
002130  012701  172340        4$:     MOV     #KIPAR0,R1      ;LOAD R1 WITH ADDRESS OF KIPAR0
002134  012702  172360                MOV     #KDPAR0,R2      ;LOAD R2 WITH ADDRESS OF KDPAR0
002140  012703  000007        5$:     MOV     #7,R3           ;LOAD LOOP COUNTER WITH 7
002144  005000                        CLR     R0              ;CLEAR PAR VALUE REGISTER
002146  010021                6$:     MOV     R0,(R1)+        ;LOAD AN I-SPACE PAR
002150  010022                        MOV     R0,(R2)+        ;LOAD A D-SPACE PAR
002152  062700  000200                ADD     #200,R0         ;INCREASE THE PAR VALUE BY 200
002156  077305                        SOB     R3,6$           ;LOOP UNTIL 7 PAR'S ARE LOADED
002160  012711  177600                MOV     #177600,(R1)    ;MAP I-SPACE PAR7 TO I/O PAGE
002164  012712  177600                MOV     #177600,(R2)    ;MAP D-SPACE PAR7 TO I/O PAGE
002170  020127  172356                CMP     R1,#KIPAR7
002174  001005                        BNE     7$
002176  012701  172240                MOV     #SIPAR0,R1
002202  012702  172260                MOV     #SDPAR0,R2
002206  000754                        BR      5$
002210  020127  172256        7$:     CMP     R1,#SIPAR7
002214  001401                        BEQ     8$              ;BRANCH TO USER LOAD ROUTINE
002216  000207                        RTS     PC              ;RETURN TO CALLING ROUTINE
002220  012701  177640        8$:     MOV     #UIPAR0,R1
002224  012702  177660                MOV     #UDPAR0,R2
002230  000743                        BR      5$
```

D-SPACE TESTS MEMORY MANAGEMENT ABORT SERVICE ROUTINE

```
                                        .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.
                                        ;;*****************************************************************
002232                                  NODSPAC:                        ;STARTING ADDRESS FOR ABORT SERVICE ROUTINE
002232  042737  000004  172516                BIC     #BIT2,MMR3      ;TURN OFF D-SPACE BEFORE DOING ROUTINE
002240  005227                                INC     (PC)+           ;MAKE FLAG ZERO IF THE FIRST TIME
002242  177777                        NDFLAG: .WORD   -1              ;FLAG SHOULD BE -1
002244  001401                                BEQ     10$             ;BRANCH IF FIRST TIME IN ROUTINE
002246  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.
002250  011637  001304                10$:      MOV     (KSP),BADPC     ;SAVE PC AT TIME OF ABORT OR TRAP
002254  012637  001260                          MOV     (KSP)+,TRAPPC   ;SAVE RETURN ADDRESS IN CASE OF LOOP
002260  012637  001262                          MOV     (KSP)+,TRAPPS   ;SAVE OLD PSW IN CASE OF LOOP
002264  013737  177572  001264                MOV     MMR0,WASSR0     ;SAVE STATUS REGISTER
002272  013737  177574  001266                MOV     MMR1,WASSR1     ;SAVE AUTO INCR/DECR REGISTER
002300  013737  177576  001270                MOV     MMR2,WASSR2     ;SAVE VIRTUAL ADDRESS REGISTER
002306  005737  001264                        TST     WASSR0          ;WAS ABORT NON-RESIDENT?
002312  100002                                BPL     1$              ;BRANCH IF ABORT NOT EXPECTED
002314  104034                                ERROR   +34             ;D-SPACE ENABLE FAULTY
002316  000401                                BR      2$              ;BRANCH TO EXIT
002320  104002                        1$:     ERROR   +2              ;UNEXPECTED M.M. ABORT
002322  042737  177376  177572        2$:     BIC     #177376,MMR0    ;CLEAR ALL BITS EXCEPT 0 AND 8
002330  012737  177777  002242                MOV     #-1,NDFLAG      ;MOVE A -1 TO THE FLAG
002336  013746  001262                        MOV     TRAPPS,-(KSP)   ;PUSH OLD PSW ONTO STACK
002342  013746  001260                        MOV     TRAPPC,-(KSP)   ;PUSH OLD PC ONTO STACK
002346  052737  000004  172516                BIS     #BIT2,MMR3      ;TURN D-SPACE BACK ON
002354  000006                                RTT                     ;RETURN TO MAIN PROGRAM
```

TURN OFF T-BIT AND SAVE CURRENT PSW

```
                                        .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.
                                        ;*
                                        ;;*****************************************************************
002356  033727  177776  000020  TOFF:   BIT     PSW,#TBIT       ;IS THE T-BIT SET IN THE PSW?
002364  001411                          BEQ     1$              ;EXIT IF NO
002366  013746  177776                  MOV     PSW,-(SP)       ;PUSH PRESENT PSW ON THE STACK
002372  011637  001274                  MOV     (SP),TBITPS     ;ALSO SAVE IT IN "TBITPS" FOR
                                                                ;RESTORING LATER
002376  042716  000020                  BIC     #TBIT,(SP)      ;CLEAR THE T-BIT (BIT 4) IN THE PSW
002402  012746  002410                  MOV     #1$,-(SP)       ;PUSH PC OF "RTS" ON STACK
002406  000006                          RTT                     ;"RETURN" TO 1$ WITH T-BIT OFF
002410  000207                  1$:     RTS     PC              ;RETURN TO PROGRAM
```

TURN ON T-BIT AND RESTORE PREVIOUS PSW

```
                                        .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.
                                        ;*
                                        ;*****************************************************************
002412  033727  001274  000020  TON:    BIT     TBITPS,#TBIT    ;WAS T-BIT ON IN THE PREVIOUS PSW?
002420  001410                          BEQ     1$              ;EXIT IF NO
002422  013746  001274                  MOV     TBITPS,-(SP)    ;PUSH PREVIOUS PSW ON THE STACK
002426  012737  000340  001274          MOV     #340,TBITPS     ;RESET THE "TBITPS" LOCATION
002434  012746  002442                  MOV     #1$,-(SP)       ;PUSH PC OF "RTS" ON STACK
002440  000006                          RTT                     ;"RETURN" TO 1$ WITH T-BIT RESTORED
002442  000207                  1$:     RTS     PC              ;RETURN TO PROGRAM
```

```
                                        .SBTTL  SUBROUTINE TO PREPARE ERLOOP, THE STACK AND EXIT TO ERROR
                                ;;**********************************************************************
                                ;*
                                ;*      THIS SUBROUTINE IS USED BY THE THREE SUBROUTINES MFPITS, MTPITS AND
                                ;*      MFPDTS WHEN AN ERROR IS CALLED.  THE RETURN ADDRESS IS POPPED AS
                                ;*      RETURN IS NOT BACK TO THE SUBROUTINE, BUT TO THE TEST ERROR CALL.  THIS
                                ;*      ROUTINE SETS LOCATION ERLOOP, DUPLICATES THE RETURN ADDRESS ON THE
                                ;*      STACK FOR POSSIBLE LOOP ON ERROR, AND CORRECTS THE RETURN ADDRESS TO
                                ;*      POINT TO THE ERROR CALL 4 LOCATIONS BACK FROM THE "TEST PASSED" LOCA-
                                ;*      TION DUPLICATED.  NOTE THE "TEST PASSED" RETURN ON THE STACK IS NOT
                                ;*      TOUCHED FOR PROBABLE LATER USE WHEN LOOPING IS NO LONGER ENABLED.
                                ;*
                                ;;**********************************************************************
002444  005726                  ERPREP: TST     (SP)+           ;POP RETURN OF THIS ROUTINE - NOT USED
002446  012737  000001  002624          MOV     #1,ERLOOP       ;SET ERROR LOOPING FLAG INDICATING AN ERROR
002454  011646                          MOV     (SP),-(SP)      ;DUPLICATE RETURN PC AND
002456  162716  000004                  SUB     #4,(SP)         ;FUDGE FOR ERROR RETURN
002462  000207                          RTS     PC              ;RETURN TO THE ERROR CALL IN THE TEST
```

```
SUBROUTINE TO TEST MFPI INSTRUCTION

                                        .SBTTL  SUBROUTINE TO TEST MFPI INSTRUCTION
                                    ;;**********************************************************************
                                    ;*
                                    ;*      USAGE OF THE SUBROUTINE BELOW IS AS FOLLOWS:
                                    ;*
                                    ;*      MOV     #MFPILP,$LPERR  ;PUT ADDRESS OF LOOPING LOCATION IN $LPERR
                                    ;*      MOV     #(NUMB),MFPIPS  ;PUT PS PREVIOUS/CURRENT MODE VALUE IN THIS LOCATION
                                    ;*      MOV     #TRAPRTN,MFPIVC ;LOAD THE TEST EXCLUSIVE TRAP ROUTINE TO MFPIVC
                                    ;*      MOV     #(NUMB),R2      ;SETUP ADDRESS IN R2
                                    ;*   (IT IS ASSUMED THAT ALL PDR'S/PAR'S WILL ALSO BE SET UP PROPERLY)
                                    ;*      JSR     PC,MFPITS       ;GO DO THE TEST
                                    ;*      MFPI    (MODE)          ;MFPI INSTRUCTION TO TEST IS PUT HERE
                                    ;*      NOP                     ;NEEDED FOR MODES 1,2,4, & 5 *ONLY*
                                    ;*      ERROR   +(NUMB)         ;RETURN IS HERE FOR ERROR CALLS
                                    ;*      TST     (SP)+           ;POP STACK - EXTRA RETURN INSTALLED BY SUBRTN NOT NEEDED
                                    ;*
                                    ;;**********************************************************************
002464  005037  002624            MFPITS: CLR     ERLOOP          ;CLEAR ERROR LOOPING FLAG
002470  011605                          MOV     (SP),R5         ;MOVE STACK POINTER TO R5 FOR SUBROUTINE LOADING
002472  012537  002532                  MOV     (R5)+,MFPILD    ;MOVE MFPI INSTRUCTION TO LOCATION
002476  012537  002534                  MOV     (R5)+,MFPILD+2  ;MOVE NEXT WORD TO LOCATION
002502  062716  000010                  ADD     #10,(SP)        ;FUDGE RETURN TO "TEST PASSED" LOCATION
002506  010637  001176                  MOV     SP,$TMP0        ;SAVE THE STACK POINTER AND
002512  011637  001200                  MOV     (SP),$TMP1      ;SAVE THE RETURN ADDRESS
002516  012737                  MFPILP: MOV     (PC)+,@(PC)+    ;SETUP PSW AS DEFINED BY LOADED VALUE IN PRVMD1
002520  000000                  MFPIPS: .WORD   0               ;LOCATION TO HOLD SUPER/USER PREVIOUS MODE
002522  177776                          .WORD   PSW             ;LOAD THE PSW
002524  017737  000072  000250          MOV     @MFPIVC,MMVEC   ;SET M.M. VECTOR TO MFPIV1
002532  000000  000000          MFPILD: .WORD   0,0             ;LOCATIONS TO HOLD MFPI INSTRUCTION UNDER TEST
002536  012737  016142  000250          MOV     #MGMERR,MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
002544  012601                          MOV     (SP)+,R1        ;POP SUPERVISOR/USER/KERNEL STACK INTO R1
002546  020001                          CMP     R0,R1           ;WAS DATA FETCHED SAME AS STORED
002550  001411                          BEQ     1$              ;BRANCH IF CORRECT DATA WAS FETCHED
002552  012737  000340  177776          MOV     #340,PSW        ;GO TO KERNEL MODE
002560  013706  001176                  MOV     $TMP0,SP        ;RESET SP
002564  013716  001200                  MOV     $TMP1,(SP)      ;RESET RETURN ADDRESS
002570  004737  002444                  JSR     PC,ERPREP       ;GO PREPARE LOCATION ERLOOP, RETURN PC & EXIT TO ERROR
                                ;FOR TIGHTER SCOPE LOOP, REPLACE "JSR PC,ERPREP" WITH "BR MFPILP" = 000767
002574  005737  002624            1$:     TST     ERLOOP          ;CHECK TO SEE IF THIS "PASSED" IS IN AN ERROR LOOP
002600  001346                          BNE     MFPILP          ;BRANCH BACK IF SO
002602  012737  000340  177776          MOV     #340,PSW        ;GET BACK TO KERNEL MODE
002610  013706  001176                  MOV     $TMP0,SP        ;RESET SP
002614  013716  001200                  MOV     $TMP1,(SP)      ;RESET RETURN ADDRESS
002620  000207                          RTS     PC              ;EXIT - TEST PASSED
002622  000000                  MFPIVC: .WORD   0               ;LOCATION TO HOLD ADDRESS OF MM TRAP CATCHER
002624  000000                  ERLOOP: .WORD   0               ;LOCATION USED TO FLAG AN ERROR CONDITION
```

```
                                        .SBTTL  SUBROUTINE TO CHECK THE MTPI INSTRUCTION
                                ;;**************************************************************
                                ;*
                                ;*      USAGE OF THE SUBROUTINE BELOW IS AS FOLLOWS:
                                ;*
                                ;*      MOV     #MTPILP,$LPERR  ;PUT ADDRESS OF LOOPING LOCATION IN $LPERR
                                ;*      MOV     #(NUMB),MTPIPS  ;PUT PS PREVIOUS/CURRENT MODE VALUE IN THIS LOCATION
                                ;*      MOV     #TRAPRTN,MTPIVC ;LOAD THE TEST EXCLUSIVE TRAP ROUTINE TO MTPIVC
                                ;*      MOV     #(NUMB),R2      ;SETUP ADDRESS IN R2
                                ;*   (IT IS ASSUMED THAT ALL PDR'S/PAR'S WILL ALSO BE SET UP PROPERLY)
                                ;*      JSR     PC,MTPITS       ;GO DO THE TEST
                                ;*      MTPI    (MODE)          ;MTPI 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
                                ;*
                                ;;**************************************************************
002626  005037  002624        MTPITS: CLR     ERLOOP          ;CLEAR ERROR LOOPING FLAG
002632  011605                        MOV     (SP),R5         ;MOVE STACK POINTER TO R5 FOR LOADING
002634  012537  002676                MOV     (R5)+,MTPILD    ;MOVE MTPI TO LOCATION
002640  012537  002700                MOV     (R5)+,MTPILD+2  ;MOVE NEXT WORD TO LOCATION
002644  012537  002720                MOV     (R5)+,MTPITA    ;MOVE NUMBER TO ADD TO R2 TO LOCATION
002650  062716  000012                ADD     #12,(SP)        ;FUDGE RETURN TO "TEST PASSED" LOCATION
002654  012737                MTPILP: MOV     (PC)+,@(PC)+    ;MAKE PREVIOUS MODE USER/SUPERVISOR
002656  000000                MTPIPM: .WORD   0               ;LOCATION TO HOLD PREVIOUS MODE USER OR SUPER
002660  177776                        .WORD   PSW             ;LOAD THE PSW
002662  010046                        MOV     R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
002664  105037  172310                CLRB    KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
002670  017737  000050  000250        MOV     @MTPIVC,MMVEC   ;SET MM TO VECTOR IN MTPIVC
002676  000000  000000        MTPILD: .WORD   0,0             ;LOCATIONS USED TO PLACE THE MPTI INSTRUCTION
002702  012737  016142  000250        MOV     #MGMERR,MMVEC   ;RESTORE MM VECTOR TO NORMAL ROUTINE
002710  112737  000006  172310        MOVB    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
002716  062702                        ADD     (PC)+,R2        ;ADD NUMBER TO R2 TO UNDO INCREMENT/DECREMENT
002720  000000                MTPITA: .WORD   0               ;LOCATION TO HOLD -2, 0, +2 OR 100000
002722  011201                        MOV     (R2),R1         ;READ FROM ADDRESS 60000
002724  020001                        CMP     R0,R1           ;SEE IF DATA WAS STORED AT CORRECT PLACE
002726  001402                        BEQ     1$              ;BRANCH IF IT WAS
002730  004737  002444                JSR     PC,ERPREP       ;GO PREPARE LOCATION ERLOOP, RETURN PC & EXIT TO ERROR
                                ;FOR TIGHTER ERROR LOOP, REPLACE "JSR PC,ERPREP" WITH "BR MTPILP" = 000754
002734  005737  002624        1$:     TST     ERLOOP          ;SEE IF THIS "PASSED" WAS IN AN ERROR LOOP
002740  001345                        BNE     MTPILP          ;BRANCH BACK IF SO
002742  000207                        RTS     PC              ;EXIT - TEST PASSED
002744  000000                MTPIVC: .WORD   0               ;LOCATION TO HOLD MM VECTOR TO LOAD
```

SUBROUTINE TO CHECK THE MFPD INSTRUCTION

```
                                        .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
                                ;*
                                ;*****************************************************************
002746  005037  002624        MFPDTS: CLR     ERLOOP          ;CLEAR THE ERROR LOOPING FLAG
002752  011605                        MOV     (SP),R5         ;MOVE RETURN ADDRESS TO R5 FOR LOADING
002754  012537  003004                MOV     (R5)+,MFPDLD    ;MOVE MFPD INSTRUCTION TO THE LOCATION BELOW
002760  012537  003006                MOV     (R5)+,MFPDLD+2  ;MOVE NEXT WORD TO THE NEXT LOCATION
002764  062716  000010                ADD     #10,(SP)        ;FUDGE RETURN TO THE "TEST PASSED" LOCATION
002770  012737                MFPDLP: MOV     (PC)+,@(PC)+    ;SET UP PSW AS LOADED BY TEST RUNNING THIS SUBROUTINE
002772  000000                MFPDPS: .WORD   0               ;LOCATION TO HOLD NUMBER TO LOAD THE PSW
002774  177776                        .WORD   PSW             ;LOAD THE PSW
002776  017737  000036  000250        MOV     @MFPDVC,MMVEC   ;SET M.M. VECTOR TO TRAP CATCHER OF THE TEST
003004  000000  000000        MFPDLD: .WORD   0,0             ;LOCATIONS TO HOLD THE MFPD INSTRUCTION
003010  012737  016142  000250        MOV     #MGMERR,MMVEC   ;RESTORE MM VECTOR TO NORMAL ROUTINE
003016  012601                        MOV     (SP)+,R1        ;POP K/S/U STACK INTO R1
003020  020001                        CMP     R0,R1           ;WAS DATA FETCHED SAME AS DATA STORED
003022  001402                        BEQ     1$              ;BRANCH IF CORRECT DATA WAS FETCHED
003024  004737  002444                JSR     PC,ERPREP       ;GO PREPARE ERLOOP, STACK AND EXIT TO ERROR CALL
003030  005737  002624        1$:     TST     ERLOOP          ;SEE IF THIS "PASSED" IS IN AN ERROR LOOP
003034  001355                        BNE     MFPDLP          ;BRANCH BACK IF SO
003036  000207                        RTS     PC              ;EXIT - TEST PASSED
003040  000000                MFPDVC: .WORD   0               ;LOCATION TO HOLD MM VECTOR OF TEST
```

```
 1785                                   .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

 003042                                 $SCOPE:
 003042  104410                         1$:     CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
 003044  032777  040000  176066                 BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
 003052  001077                                 BNE     $OVER           ;;YES IF SW14=1
                                        ;#####START OF CODE FOR THE XOR TESTER#####
 003054  000416                         $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                        ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
 003056  013746  000004                         MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
 003062  012737  003102  000004                 MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
 003070  005737  177060                         TST     @#177060        ;;TIME OUT ON XOR?
 003074  012637  000004                         MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 003100  000446                                 BR      $SVLAD          ;;GO TO THE NEXT TEST
 003102  022626                         5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
 003104  012637  000004                         MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 003110  000434                                 BR      7$              ;;LOOP ON THE PRESENT TEST
 003112                                 6$:;#####END OF CODE FOR THE XOR TESTER#####
 003112  032777  000400  176020                 BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
 003120  001404                                 BEQ     2$              ;;BR IF NO
 003122  127737  176012  001102                 CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
 003130  001450                                 BEQ     $OVER           ;;BR IF YES
 003132  013737  177766  003266  2$:     MOV     177766,CPSAVE   ;MOVE CPU ERR REG VALUE TO LOC FOR TST  ;DPM001
 003140  032737  000001  003266                 BIT     #BIT00,CPSAVE   ;SEE IF THE POWER MONITOR BIT IS ON     ;DPM001
 003146  001406                                 BEQ     2000$           ;BRANCH TO CONTINUE ROUTINE IF CLEAR    ;DPM001
 003150  042737  000001  177766                 BIC     #BIT00,177766   ;CLEAR THE BIT FOUND TO BE SET          ;DPM001
 003156  104177                                 EMT     +177            ;CALL SPECIAL POWER FAIL BIT ERROR CALL ;DPM001
 003160  105037  001103                         CLRB    $ERFLG          ;CLEAR THE ERROR FLAG                   ;DPM001
 003164  105737  001103                 2000$:  TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
 003170  001412                                 BEQ     $SVLAD          ;;BR IF NO
 003172  032777  001000  175740                 BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
 003200  001404                                 BEQ     4$              ;;BR IF NO
 003202  013737  001110  001106  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
 003210  000420                                 BR      $OVER
 003212  105037  001103                 4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
 003216  105237  001102                 $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
 003222  113737  001102  001230                 MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
 003230  011637  001106                         MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
 003234  011637  001110                         MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
 003240  005037  001212                         CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
 003244  112737  000001  001115                 MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
 003252  013777  001102  175662  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
 003260  013716  001106                         MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
 003264  000002                                 RTI                     ;;FIXES PS
 003266  000000                         CPSAVE: .WORD   0               ;LOCATION TO SAVE CPU ERR REG CONTENTS  ;DPM001
```

```
1787                                    .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

003270  105037  003646          $ERROR: CLRB    IBSAVE          ;CLEAR THE ITEM BYTE SAVE LOCATION      ;DPM001
003274  104410                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
003276  010037  001162                  MOV     R0,$REG0        ;SAVE THE CONTENTS OF R0
003302  010137  001164                  MOV     R1,$REG1        ;SAVE THE CONTENTS OF R1
003306  010237  001166                  MOV     R2,$REG2        ;SAVE THE CONTENTS OF R2
003312  010337  001170                  MOV     R3,$REG3        ;SAVE THE CONTENTS OF R3
003316  010437  001172                  MOV     R4,$REG4        ;SAVE THE CONTENTS OF R4
003322  010537  001174                  MOV     R5,$REG5        ;SAVE THE CONTENTS OF R5
003326  113737  001102  001254          MOVB    $TSTNM,TESTNO   ;SAVE THE TEST NUMBER
003334  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
003340  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
003342  013777  001102  175572          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
003350  032777  002000  175562          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
003356  001402                          BEQ     1$              ;;NO - SKIP
003360  104401  001214                  TYPE    ,$BELL          ;;RING BELL
003364  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
003370  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
003374  162737  000002  001116          SUB     #2,$ERRPC
003402  117737  175510  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
003410  122737  000177  001114          CMPB    #177,$ITEMB     ;SEE IF THIS IS THE POWER FAIL CALL      ;DPM001
003416  001426                          BEQ     1001$           ;BRANCH AROUND ROUTINE IF IT IS         ;DPM001
003420  105737  003646                  TSTB    IBSAVE          ;SEE IF THIS IS THE 2ND ERROR CALL      ;DPM001
003424  001021                          BNE     1000$           ;BRANCH IF SO                           ;DPM001
003426  013737  177766  003266          MOV     177766,CPSAVE   ;MOVE CPU ERR REG TO CPSAVE FOR TEST    ;DPM001
003434  032737  000001  003266          BIT     #BIT00,CPSAVE   ;SEE IF POWER MONITOR BIT IS SET        ;DPM001
003442  001414                          BEQ     1001$           ;BRANCH IF OK                           ;DPM001
003444  042737  000001  177766          BIC     #BIT00,177766   ;CLEAR THE BIT FOUND SET                ;DPM001
003452  113737  001114  003646          MOVB    $ITEMB,IBSAVE   ;MAKE IBSAVE NON-ZERO FOR DUAL CALL     ;DPM001
003460  112737  000177  001114          MOVB    #177,$ITEMB     ;SET $ITEMB TO SPECIAL POWER FAIL PNTR  ;DPM001
003466  000402                          BR      1001$           ;BRANCH OVER IBSAVE CLEARING            ;DPM001
003470  105037  003646          1000$:  CLRB    IBSAVE          ;CLEAR IBSAVE SO AFTER 2ND ERROR, EXIT  ;DPM001
003474                          1001$:
003474  032777  020000  175436          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
003502  001004                          BNE     20$             ;;SKIP TYPEOUTS
003504  004737  003650                  JSR     PC,ERRTYP       ;;GO TO USER ERROR ROUTINE
003510  104401  001221                  TYPE    ,$CRLF
003514                          20$:
003514  122737  000001  001244          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
003522  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
003524  113737  001114  003536          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
003532  004737  006014                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
003536     000                  21$:    .BYTE   0
003537     000                          .BYTE   0
```

```
003540  000777                  22$:    BR      22$             ;;APT ERROR LOOP
003542  105737  003646          2$:     TSTB    IBSAVE          ;SEE IF POWER FAIL ERROR CALL           ;DPM001
003546  001005                          BNE     3$              ;BRANCH IF NOT - HALT NOT ALLOWED       ;DPM001
003550  005777  175364                  TST     @SWR            ;;HALT ON ERROR
003554  100002                          BPL     3$              ;;SKIP IF CONTINUE
003556  000000                          HALT                    ;;HALT ON ERROR.
003560  104410                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
003562  032777  001000  175350  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
003570  001405                          BEQ     4$              ;;BR IF NO
003572  105737  003646                  TSTB    IBSAVE          ;SEE IF THIS IS THE PWR MNTR BIT ERROR  ;DPM001
003576  001256                          BNE     7$              ;BRANCH BACK IF SO - FUDGING NOT ALLOWED;DPM001
003600  013716  001110                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
003604  005737  001212          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
003610  001405                          BEQ     5$              ;;BR IF NONE
003612  105737  003646                  TSTB    IBSAVE          ;SEE IF THIS IS THE PWR MNTR BIT ERROR  ;DPM001
003616  001246                          BNE     7$              ;BRANCH BACK IF SO - FUDGING NOT ALLOWED;DPM001
003620  013716  001212                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
003624                          5$:
003624  022737  036252  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
003632  001001                          BNE     6$              ;;BRANCH IF NO
003634  000000                          HALT                    ;;YES
003636                          6$:
003636  105737  003646                  TSTB    IBSAVE          ;SEE IF THIS IS THE PWR FAIL ERROR CALL ;DPM001
003642  001234                          BNE     7$              ;BRANCH BACK TO CALL ORIGINAL ERR IF SO ;DPM001
003644  000002                          RTI                     ;;RETURN
003646  000000                  IBSAVE: .WORD   0               ;LOC'N TO HOLD $ITEMB DURING DUAL ERR   ;DPM001
```

```
1789 003650  104401  001221          ERRTYP: TYPE    ,$CRLF          ;TYPE <CRLF>
1790 003654  010046                          MOV     R0,-(KSP)       ;SAVE R0.
1791 003656  005000                          CLR     R0              ;PICKUP THE ITEM INDEX
1792 003660  153700  001114                  BISB    @#$ITEMB,R0
1793 003664  001004                          BNE     1$              ;IF ITEM NUMBER IS ZERO, JUST
1794                                                                 ;TYPE THE PC OF THE ERROR
1795 003666  013746  001116                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
1796                                                                 ;;ERROR ADDRESS
1797 003672  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1798 003674  000530                          BR      13$             ;GET OUT
1799 003676  122700  000177          1$:     CMPB    #177,R0         ;SEE IF THIS ERROR CALL IS THE POWER MONITOR BIT CALL
1800 003702  001003                          BNE     100$            ;BRANCH IF NOT
1801 003704  012700  004172                  MOV     #PFECWS,R0      ;MOVE ADDRESS OF POWER MONITOR BIT ERROR TO R0
1802 003710  000406                          BR      110$            ;BRANCH TO CALL THE ERROR
1803 003712  005300                  100$:   DEC     R0              ;ADJUST THE INDEX SO THAT IT WILL
1804 003714  006300                          ASL     R0              ;WORK FOR THE ERROR TABLE.
1805 003716  006300                          ASL     R0
1806 003720  006300                          ASL     R0
1807 003722  062700  001320                  ADD     #$ERRTB,R0      ;FORM TABLE POINTER
1808 003726  012037  003736          110$:   MOV     (R0)+,2$        ;PICKUP "ERROR MESSAGE" POINTER
1809 003732  001404                          BEQ     3$              ;SKIP TYPEOUT IF NO POINTER
1810 003734  104401                          TYPE                    ;TYPE THE "ERROR MESSAGE"
1811 003736  000000                  2$:     .WORD   0               ;"ERROR MESSAGE" POINTER GOES HERE
1812 003740  104401  001221                  TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
1813 003744  012037  003754          3$:     MOV     (R0)+,4$        ;PICKUP "DATA HEADER" POINTER
1814 003750  001404                          BEQ     5$              ;SKIP TYPEOUT IF 0
1815 003752  104401                          TYPE                    ;TYPE THE "DATA HEADER"
1816 003754  000000                  4$:     .WORD   0               ;"DATA HEADER" POINTER GOES HERE
1817 003756  104401  001221                  TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
1818 003762  010146                  5$:     MOV     R1,-(KSP)       ;SAVE R1
1819 003764  012001                          MOV     (R0)+,R1        ;PICKUP "DATA TABLE" POINTER
1820 003766  001472                          BEQ     12$             ;BR IF NO DATA TO BE TYPED
1821 003770  012000                          MOV     (R0)+,R0        ;PICKUP "DATA FORMAT" POINTER
1822 003772  105710                  6$:     TSTB    (R0)            ;IS IT FORMAT 0?
1823 003774  001003                          BNE     7$              ;BR IF NO
1824                                 ;*THIS CODE IS FOR OCTAL (16-BIT) FORMAT (DF=0)
1825 003776  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FPR TYPEOUT
1826 004000  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1827 004002  000456                          BR      11$
1828                                 ;*THIS CODE IS FOR DECIMAL FORMAT (DF=1)
1829 004004  121027  000001          7$:     CMPB    (R0),#1         ;IS IT FORMAT 1?
1830 004010  001003                          BNE     8$              ;BRANCH IF NO
1831 004012  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
1832 004014  104405                          TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
1833 004016  000450                          BR      11$
1834                                 ;*THIS CODE IS FOR BINARY FORMAT (DF=2)
1835 004020  121027  000002          8$:     CMPB    (R0),#2         ;IS IT FORMAT 2
1836 004024  001003                          BNE     9$              ;BRANCH IF NO
1837 004026  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
1838 004030  104406                          TYPBN                   ;;GO TYPE--BINARY ASCII
1839 004032  000442                          BR      11$
1840                                 ;*THIS CODE IS FOR OCTAL (22-BIT) FORMAT (DF=3)
1841 004034  121027  000003          9$:     CMPB    (R0),#3         ;IS IT FORMAT 3?
1842 004040  001011                          BNE     15$             ;BRANCH IF NO
1843 004042  012146                          MOV     (R1)+,-(KSP)    ;PUT ADDRESS OF FIRST LOC. ON STACK
1844 004044  004737  007066                  JSR     PC,$DB2O        ;CONVERT TWO LOCS. TO AN ASCII STRING
1845 004050  062716  000003                  ADD     #3,(KSP)        ;ONLY NEED 8 CHARACTERS NOT 11
```

ERROR HANDLER ROUTINE

```
1846 004054  012637  004062                MOV     (KSP)+,10$      ;PUT ADDRESS OF ASCII CHARS. AT 10$
1847 004060  104401                        TYPE                    ;TYPE OCTAL VALUE OF 22-BIT BINARY NO.
1848 004062  000000              10$:      .WORD   0
1849                             ;*THIS CODE IS FOR OCTAL (22-BIT) FORMAT FOR A PAR LEFT SHIFTED 6 (DF=4)
1850 004064  010246              15$:      MOV     R2,-(KSP)       ;SAVE R2 ON STACK
1851 004066  010346                        MOV     R3,-(KSP)       ;SAVE R3 ON STACK
1852 004070  013103                        MOV     @(R1)+,R3       ;LOAD DATA WORD INTO R3
1853 004072  005002                        CLR     R2              ;R2 HOLDS UPPER SIX BITS OF NUMBER
1854 004074  073227  000006                ASHC    #6,R2           ;SHIFT VALUE LEFT 6 TIMES
1855 004100  010237  001206                MOV     R2,$TMP4        ;HOLDS LOWER 16 BITS OF ADDRESS
1856 004104  010337  001210                MOV     R3,$TMP5        ;HOLDS UPPER 6 BITS OF ADDRESS
1857 004110  012746  001206                MOV     #$TMP4,-(KSP)   ;PUT ADDRESS OF LOWER BITS ONTO STACK
1858 004114  004737  007066                JSR     PC,$DB2O        ;CONVERT TWO LOCS. TO AN ASCII STRING
1859 004120  062716  000003                ADD     #3,(KSP)        ;ONLY NEED 8 CHARACTERS NOT 11
1860 004124  012637  004132                MOV     (KSP)+,16$      ;PUT ADDRESS OF ASCII CHARS. AT 16$
1861 004130  104401                        TYPE                    ;TYPE OCTAL VALUE OF 22-BIT BINARY NO.
1862 004132  000000              16$:      .WORD   0
1863 004134  012603                        MOV     (KSP)+,R3       ;RESTORE R3
1864 004136  012602                        MOV     (KSP)+,R2       ;RESTORE R2
1865 004140  005711              11$:      TST     (R1)            ;IS THERE ANOTHER NUMBER?
1866 004142  001404                        BEQ     12$             ;BR IF NO
1867 004144  104401  004166                TYPE    ,14$            ;TYPE TWO(2) SPACES
1868 004150  105720                        TSTB    (R0)+           ;POINT TO NEW "DATA FORMAT"
1869 004152  000707                        BR      6$              ;LOOP
1870 004154  012601              12$:      MOV     (KSP)+,R1       ;RESTORE R1
1871 004156  012600              13$:      MOV     (KSP)+,R0       ;RESTORE R0
1872 004160  104401  001221                TYPE    ,$CRLF          ;"CARRIAGE RETURN" & "LINE FEED"
1873 004164  000207                        RTS     PC              ;RETURN
1874 004166     040     040     000  14$:  .ASCIZ  /  /            ;TWO(2) SPACES
1875 004171     000                        .BYTE   0
1876 004172  004202  004242  004272  PFECWS: .WORD  PFECEM,PFECDH,PFECDT,PFECDF
     004200  004302
1877 004202     120     117     127  PFECEM: .ASCIZ  ?POWER MONITOR BIT WAS FOUND SET?
     004205     105     122     040
     004210     115     117     116
     004213     111     124     117
     004216     122     040     102
     004221     111     124     040
     004224     127     101     123
     004227     040     106     117
     004232     125     116     104
     004235     040     123     105
     004240     124     000
1878 004242     124     105     123  PFECDH: .ASCIZ  ?TESTNO  ERR PC  CPUERR?
     004245     124     116     117
     004250     040     040     105
     004253     122     122     040
     004256     120     103     040
     004261     040     103     120
     004264     125     105     122
     004267     122     000
1879                                  .EVEN
1880 004272  001230  001116  003266  PFECDT: .WORD  $TESTN,$ERRPC,CPSAVE,0
     004300  000000
1881 004302     000     000     000  PFECDF: .BYTE  0,0,0,0
     004305     000
```

```
1882                                    .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.
004306  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
004314  001114                        BNE     15$             ;;BRANCH IF NO
004316  105777  174622                TSTB    @$TKS           ;;CHAR THERE?
004322  100111                        BPL     15$             ;;IF NO, DON'T WAIT AROUND
004324  117746  174616                MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
004330  042716  177600                BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
004334  022726  000007                CMP     #7,(SP)+        ;;IS IT A CONTROL G?
004340  001102                        BNE     15$             ;;NO, RETURN TO USER
004342  123727  001134  000001        CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
004350  001476                        BEQ     15$             ;;BRANCH IF YES
004352  104401  005253                TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
004356  104401  005260        $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
004362  013746  000176                MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
004366  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
004370  104401  005271                TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
004374  005046                19$:    CLR     -(SP)           ;;CLEAR COUNTER
004376  005046                        CLR     -(SP)           ;;THE NEW SWR
004400  105777  174540        7$:     TSTB    @$TKS           ;;CHAR THERE?
004404  100375                        BPL     7$              ;;IF NOT TRY AGAIN
004406  117746  174534                MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
004412  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
004416  021627  000003                CMP     (SP),#3         ;;IS IT A CONTROL-C?
004422  001015                        BNE     9$              ;;BRANCH IF NOT
004424  104401  001312                TYPE    ,$CNTLC         ;;YES, ECHO CONTROL-C (^C)
004430  062706  000006                ADD     #6,SP           ;;CLEAN UP STACK
004434  123727  001135  000001        CMPB    $INTAG,#1       ;;REENABLE TTY KEYBOARD INTERRUPTS?
004442  001003                        BNE     8$              ;;BRANCH IF NO
004444  012777  000100  174472        MOV     #100,@$TKS      ;;ALLOW TTY KEYBOARD INTERRUPTS
004452  000137  005302        8$:     JMP     CNTRLC          ;;CONTROL-C RESTART
004456  021627  000025        9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
004462  001005                        BNE     10$             ;;BRANCH IF NOT
004464  104401  005246                TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
004470  062706  000006        20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
004474  000737                        BR      19$             ;;LET'S TRY IT AGAIN
004476  021627  000015        10$:    CMP     (SP),#15        ;;IS IT A <CR>?
004502  001022                        BNE     16$             ;;BRANCH IF NO
004504  005766  000004                TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
004510  001403                        BEQ     11$             ;;BRANCH IF YES
004512  016677  000002  174420        MOV     2(SP),@SWR      ;;SAVE NEW SWR
004520  062706  000006        11$:    ADD     #6,SP           ;;CLEAR UP STACK
004524  104401  001221        14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
004530  123727  001135  000001        CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
004536  001003                        BNE     15$             ;;BRANCH IF NOT
004540  012777  000100  174376        MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
004546  000002                15$:    RTI                     ;;RETURN
004550  004737  005644        16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
004554  021627  000060                CMP     (SP),#60        ;;CHAR < 0?
004560  002420                        BLT     18$             ;;BRANCH IF YES
004562  021627  000067                CMP     (SP),#67        ;;CHAR > 7?
```

```
004566  003015                          BGT     18$             ;;BRANCH IF YES
004570  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
004574  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
004600  001403                          BEQ     17$             ;;BRANCH IF YES
004602  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
004604  006316                          ASL     (SP)            ;;    CHAR OVER TO MAKE
004606  006316                          ASL     (SP)            ;;    ROOM FOR NEW ONE.
004610  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
004614  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
004620  000667                          BR      7$              ;;GET THE NEXT ONE
004622  104401  001220          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
004626  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
                                ;
004630  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
004632  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
004640  105777  174300          1$:     TSTB    @$TKS           ;;WAIT FOR
004644  100375                          BPL     1$              ;;A CHARACTER
004646  117766  174274  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
004654  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
004662  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
004670  001013                          BNE     3$              ;;BRANCH IF NO
004672  105777  174246          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
004676  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
004700  117746  174242                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
004704  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
004710  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
004714  001366                          BNE     2$              ;;IF NOT DISCARD IT
004716  000750                          BR      1$              ;;YES, RESUME
004720  026627  000004  000021  3$:     CMP     4(SP),#$XON     ;;IS IT A RANDOM XON?           ;RAN001
004726  001744                          BEQ     1$              ;;BRANCH IF YES                 ;RAN001
004730  026627  000004  000140          CMP     4(SP),#140      ;;IS IT UPPER CASE?
004736  002407                          BLT     4$              ;;BRANCH IF YES
004740  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
004746  003003                          BGT     4$              ;;BRANCH IF YES
004750  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
004756  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
004760  010346          $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
004762  005046                  CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
004764  012703  005236          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
004770  022703  005246          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
004774  101467                          BLOS    4$              ;;BR IF YES
004776  104411                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
005000  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
005002  122713  000003                  CMPB    #3,(R3)         ;;IS IT A CONTROL-C?
```

```
005006  001006                          BNE     10$             ;;BRANCH IF NO
005010  104401  001312                  TYPE    ,$CNTLC         ;;TYPE A CONTROL-C (^C)
005014  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY OFF OF THE STACK
005016  012603                          MOV     (SP)+,R3        ;;RESTORE R3
005020  000137  005302                  JMP     CNTRLC          ;;GOTO CONTROL-C RESTART
005024  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
005030  001022                          BNE     5$              ;;BR IF NO
005032  005716                          TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
005034  001007                          BNE     6$              ;;BR IF NO
005036  112737  000134  005234          MOVB    #'\,9$          ;;TYPE A BACK SLASH
005044  104401  005234                  TYPE    ,9$
005050  012716  177777                  MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
005054  005303                  6$:     DEC     R3              ;;BACKUP BY ONE
005056  020327  005236                  CMP     R3,#$TTYIN      ;;STACK EMPTY?
005062  103434                          BLO     4$              ;;BR IF YES
005064  111337  005234                  MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
005070  104401  005234                  TYPE    ,9$             ;;GO TYPE
005074  000735                          BR      2$              ;;GO READ ANOTHER CHAR.
005076  005716                  5$:     TST     (SP)            ;;RUBOUT KEY SET?
005100  001406                          BEQ     7$              ;;BR IF NO
005102  112737  000134  005234          MOVB    #'\,9$          ;;TYPE A BACK SLASH
005110  104401  005234                  TYPE    ,9$
005114  005016                          CLR     (SP)            ;;CLEAR THE RUBOUT KEY
005116  122713  000025          7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
005122  001003                          BNE     8$              ;;BR IF NO
005124  104401  005246                  TYPE    ,$CNTLU         ;;TYPE A CONTROL 'U'
005130  000715                          BR      1$              ;;GO START OVER
005132  122713  000022          8$:     CMPB    #22,(R3)        ;;IS CHARACTER A ''^R''?
005136  001011                          BNE     3$              ;;BRANCH IF NO
005140  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
005142  104401  001221                  TYPE    ,$CRLF          ;;TYPE A ''CR'' & ''LF''
005146  104401  005236                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
005152  000706                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
005154  104401  001220          4$:     TYPE    ,$QUES          ;;TYPE A '?'
005160  000701                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
005162  111337  005234          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
005166  104401  005234                  TYPE    ,9$
005172  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
005176  001274                          BNE     2$              ;;LOOP IF NOT RETURN
005200  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
005204  104401  001222                  TYPE    ,$LF            ;;TYPE A LINE FEED
005210  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
005212  012603                          MOV     (SP)+,R3        ;;RESTORE R3
005214  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
005216  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
005224  012766  005236  000004          MOV     #$TTYIN,4(SP)
005232  000002                          RTI                     ;;RETURN
005234     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
005235     000                          .BYTE   0               ;;TERMINATOR
005236                          $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
005246     136     125     015  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL 'U'
005251     012     000
005253     136     107     015  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL 'G'
005256     012     000
005260     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR = /
005263     127     122     040
005266     075     040     000
```

```
005271  040  040  116  SMNEW:  .ASCIZ  /  NEW = /
005274  105  127  040
005277  075  040  000
```

```
1884                                            .SBTTL  CONTROL-C SERVICING ROUTINE
1885
1886 005302  013737  001232  001210  CNTRLC: MOV     $PASS,$TMP5     ;GET THE VALUE OF '$PASS'
1887 005310  005237  001210                  INC     $TMP5           ;FORM CURRENT PASS #
1888 005314  104401  005361                  TYPE    ,CMSG           ;TYPE THE TEST STOPS HERE
1889 005320  113737  001102  005354          MOVB    $TSTNM,1$       ;SAVE TEST NUMBER
1890 005326  013746  005354                  MOV     1$,-(SP)        ;SAVE 1$ FO TYPEOUT
1891 005332  104402                          TYPOC
1892 005334  104401  005356                  TYPE    ,2$
1893 005340  013746  001210                  MOV     $TMP5,-(SP)     ;SAVE $TMP5 FOR TYPEOUT
1894 005344  104405                          TYPDS                   ;TYPE ASCII DECIMAL WITH SIGN
1895 005346  104407                          GTSWR                   ;ASK FOR NEW SWR VALUE
1896 005350  000137  036026                  JMP     $EOP+2          ;JUMP TO END OF PASS + 2
1897 005354  000000                  1$:     .WORD   0               ;TEST # BUFFER
1898 005356     040     040     000  2$:     .ASCIZ  /  /            ;2 SPACES & STOP MESSAGE
1899 005361     112     125     115  CMSG:   .ASCII  /JUMPING TO END OF PASS/<15><12>
     005364     120     111     116
     005367     107     040     124
     005372     117     040     105
     005375     116     104     040
     005400     117     106     040
     005403     120     101     123
     005406     123     015     012
1900 005411     124     105     123          .ASCIZ  /TESTNO PASSNO/<15><12>
     005414     124     116     117
     005417     011     120     101
     005422     123     123     116
     005425     117     015     012
     005430     000
1901                                            .EVEN
```

```
 1903                                   .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
                                        ;*
005432  105737  001157          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
005436  100002                          BPL     1$              ;;BR IF YES
005440  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
005442  000430                          BR      3$              ;;LEAVE
005444  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
005446  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
005452  122737  000001  001244          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
005460  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
005462  132737  000100  001245          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
005470  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
005472  010037  005502                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
005476  004737  006004                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
005502  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
005504  132737  000040  001245  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
005512  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
005514  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
005516  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
005520  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
005522  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
005524  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
005530  000002                          RTI                     ;;RETURN
005532  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
005536  001430                          BEQ     8$
005540  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
005544  001006                          BNE     5$
005546  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
005550  104401                          TYPE                    ;;TYPE A CR AND LF
005552  001221                          $CRLF
005554  105037  005772                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
005560  000755                          BR      2$              ;;GET NEXT CHARACTER
005562  004737  005644          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
005566  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
005572  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
005574  013746  001154                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
005600  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
005604  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
005606  004737  005644                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
005612  105337  005772                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
005616  000770                          BR      7$              ;;LOOP
                                        ;HORIZONTAL TAB PROCESSOR
005620  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
```

```
005624  004737  005644          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
005630  132737  000007  005772          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
005636  001372                          BNE     9$              ;;TAB STOP
005640  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
005642  000724                          BR      2$              ;;GET NEXT CHARACTER
005644                          $TYPEC:
005644  105777  173274                  TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?          ;MJD001
005650  100022                          BPL     10$             ;;BR IF NOT                     ;MJD001
005652  017746  173270                  MOV     @$TKB,-(SP)     ;;GET CHAR                      ;MJD001
005656  042716  177600                  BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS         ;MJD001
005662  122716  000023                  CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF                 ;MJD001
005666  001012                          BNE     102$            ;;BR IF NOT                     ;MJD001
005670                          101$:                                                           ;MJD001
005670  105777  173250                  TSTB    @$TKS           ;;WAIT FOR CHAR                 ;MJD001
005674  100375                          BPL     101$                                            ;MJD001
005676  117716  173244                  MOVB    @$TKB,(SP)      ;;GET CHAR                      ;MJD001
005702  042716  177600                  BIC     #177600,(SP)    ;;STRIP IT                      ;MJD001
005706  122716  000021                  CMPB    #$XON,(SP)      ;;WAS IT XON?                   ;MJD001
005712  001366                          BNE     101$            ;;BR IF NOT                     ;MJD001
005714                          102$:                                                           ;MJD001
005714  005726                          TST     (SP)+           ;;FIX STACK                     ;MJD001
005716                          10$:                                                            ;MJD001
005716  105777  173226                  TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
005722  100375                          BPL     10$                                             ;MJD001
005724  126627  000002  000021          CMPB    2(SP),#$XON     ;;IS CHARACTER A RANDOM XON?    ;RAN001
005732  001420                          BEQ     $TYPEX          ;;BRANCH IF YES                 ;RAN001
005734  116677  000002  173210          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
005742  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
005750  001003                          BNE     1$              ;;BRANCH IF NO
005752  105037  005772                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
005756  000406                          BR      $TYPEX          ;;EXIT
005760  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
005766  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
005770  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
005772  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
005774  000207                  $TYPEX: RTS     PC
```

```
1904                                    .SBTTL  APT COMMUNICATIONS ROUTINE
                                        ;;*****************************************************************
005776  112737  000001  006242  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
006004  112737  000001  006240  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
006012  000403                  BR      $ATYC
006014  112737  000001  006242  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
006022                          $ATYC:
006022  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
006024  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
006026  105737  006240          TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
006032  001450                  BEQ     5$              ;;IF NOT:  BR
006034  122737  000001  001244  CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
006042  001031                  BNE     3$              ;;IF NOT:  BR
006044  132737  000100  001245  BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
006052  001425                  BEQ     3$              ;;IF NOT:  BR
006054  017600  000004          MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
006060  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
006066  005737  001224  1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
006072  001375                  BNE     1$              ;;IF NOT:  WAIT
006074  010037  001240          MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
006100  105720          2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
006102  001376                  BNE     2$
006104  163700  001240          SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
006110  006200                  ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
006112  010037  001242          MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
006116  012737  000004  001224  MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
006124  000413                  BR      5$
006126  017637  000004  006152  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
006134  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
006142  013746  177776          MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
006146  004737  005432          JSR     PC,$TYPE        ;;CALL TYPE MACRO
006152  000000          4$:     .WORD   0
006154                  5$:
006154  105737  006242  10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
006160  001416                  BEQ     12$             ;;IF NOT:  BR
006162  005737  001244          TST     $ENV            ;;RUNNING UNDER APT?
006166  001413                  BEQ     12$             ;;IF NOT:  BR
006170  005737  001224  11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
006174  001375                  BNE     11$             ;;IF NOT:  WAIT
006176  017637  000004  001226  MOV     @4(SP),$FATAL   ;;GET ERROR #
006204  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
006212  005237  001224          INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
006216  105037  006242  12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
006222  105037  006241          CLRB    $LFLG           ;;CLEAR LOG FLAG
006226  105037  006240          CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
006232  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
006234  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
006236  000207                  RTS     PC              ;;RETURN
006240     000  $MFLG:  .BYTE   0               ;;MESSG. FLAG
006241     000  $LFLG:  .BYTE   0               ;;LOG FLAG
006242     000  $FFLG:  .BYTE   0               ;;FATAL FLAG
                        .EVEN
        000200          APTSIZE=200
        000001          APTENV=001
        000100          APTSPOOL=100
        000040          APTCSUP=040
```

```
1905                                    .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
006244  010146                          $TYPBN: MOV     R1,-(SP)        ;;SAVE R1 ON THE STACK
006246  016601  000006                          MOV     6(SP),R1        ;;GET THE INPUT NUMBER
006252  000261                                 SEC                     ;;SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
006254  112737  000060  006316  1$:             MOVB    #'0,$BIN        ;;SET CHARACTER TO AN ASCII "0".
006262  006101                                 ROL     R1              ;;GET THIS BIT
006264  001406                                 BEQ     2$              ;;DONE?
006266  105537  006316                          ADCB    $BIN            ;;NO--SET THE CHARACTER EQUAL TO THIS BIT
006272  104401  006316                          TYPE    ,$BIN           ;;GO TYPE THIS BIT
006276  000241                                 CLC                     ;;CLEAR "C" SO CAN KEEP TRACK OF BITS
006300  000765                                 BR      1$              ;;GO DO THE NEXT BIT
006302  012601                          2$:     MOV     (SP)+,R1        ;;POP THE STACK INTO R1
006304  016666  000002  000004                  MOV     2(SP),4(SP)     ;;ADJUST THE STACK
006312  012616                                 MOV     (SP)+,(SP)
006314  000002                                 RTI                     ;;RETURN TO USER
006316     000     000                  $BIN:   .BYTE   0,0             ;;STORAGE FOR ASCII CHAR. AND TERMINATOR
```

```
1906                                    .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
006320  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
006324  116637  000001  006543          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
006332  112637  006545                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
006336  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
006342  000406                          BR      $TYPON
006344  112737  000001  006543  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
006352  112737  000006  006545          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
006360  112737  000005  006542  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
006366  010346                          MOV     R3,-(SP)        ;;SAVE R3
006370  010446                          MOV     R4,-(SP)        ;;SAVE R4
006372  010546                          MOV     R5,-(SP)        ;;SAVE R5
006374  113704  006545                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
006400  005404                          NEG     R4
006402  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
006406  110437  006544                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
006412  113704  006543                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
006416  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
006422  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
006424  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
006426  000404                          BR      3$              ;;GO DO MSB
006430  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
006432  006105                          ROL     R5
006434  006105                          ROL     R5
006436  010503                          MOV     R5,R3
006440  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
006442  105337  006544                  DECB    $OMODE          ;;TYPE THIS DIGIT?
006446  100016                          BPL     7$              ;;BR IF NO
006450  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
006454  001002                          BNE     4$              ;;TEST FOR 0
006456  005704                          TST     R4              ;;SUPPRESS THIS 0?
006460  001403                          BEQ     5$              ;;BR IF YES
006462  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
006464  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
006470  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
```

BINARY TO OCTAL (ASCII) AND TYPE

```
006474  110337  006540                  MOVB    R3,8$           ;;SAVE FOR TYPING
006500  104401  006540                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
006504  105337  006542          7$:     DECB    $OCNT           ;;COUNT BY 1
006510  003347                  BGT     2$                      ;;BR IF MORE TO DO
006512  002402                  BLT     6$                      ;;BR IF DONE
006514  005204                  INC     R4                      ;;INSURE LAST DIGIT ISN'T A BLANK
006516  000744                  BR      2$                      ;;GO DO THE LAST DIGIT
006520  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
006522  012604                  MOV     (SP)+,R4                ;;RESTORE R4
006524  012603                  MOV     (SP)+,R3                ;;RESTORE R3
006526  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
006534  012616                  MOV     (SP)+,(SP)
006536  000002                  RTI                             ;;RETURN
006540     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
006541     000                  .BYTE   0                       ;;TERMINATOR FOR TYPE ROUTINE
006542     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
006543     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
006544  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
```

```
1907                                    .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
006546                                  $TYPDS:
006546  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
006550  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
006552  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
006554  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
006556  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
006560  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
006564  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
006570  100004                          BPL     1$              ;;BR IF INPUT IS POS.
006572  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
006574  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
006602  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
006604  012703  006762                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
006610  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
006614  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
006616  016001  006752                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
006622  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
006624  002402                          BLT     4$              ;;BR IF DONE
006626  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
006630  000774                          BR      3$
006632  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
006634  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
006636  001002                          BNE     5$              ;;FALL THROUGH IF 0
006640  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
006642  100407                          BMI     7$              ;;BR IF YES
006644  106316                  5$:     ASLB    (SP)            ;;MSD?
006646  103003                          BCC     6$              ;;BR IF NO
006650  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
006656  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
006662  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
006666  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
006670  005720                          TST     (R0)+           ;;JUST INCREMENTING
006672  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
006676  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
006700  003002                          BGT     8$              ;;GO TO EXIT
006702  010502                          MOV     R5,R2           ;;GET THE LSD
006704  000764                          BR      6$              ;;GO CHANGE TO ASCII
006706  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
006710  100003                          BPL     9$              ;;BR IF NO
006712  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
006720  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
006722  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
006724  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
006726  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
006730  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
006732  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
006734  104401  006762                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
```

```
006740  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
006746  012616                          MOV     (SP)+,(SP)
006750  000002                          RTI                     ;;RETURN TO USER
006752  023420                  $DTBL:  10000.
006754  001750                          1000.
006756  000144                          100.
006760  000012                          10.
006762                          $DBLK:  .BLKW   4
```

```
 1908                                   .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
006772                                  $SAVREG:
006772  010046                                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
006774  010146                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
006776  010246                                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
007000  010346                                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
007002  010446                                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
007004  010546                                  MOV     R5,-(SP)        ;;PUSH R5 ON STACK
007006  016646  000022                          MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
007012  016646  000022                          MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
007016  016646  000022                          MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
007022  016646  000022                          MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
007026  000002                                  RTI
                                        ;*RESTORE R0-R5
                                        ;*CALL:
                                        ;*      RESREG
007030                                  $RESREG:
007030  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
007034  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
007040  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
007044  012666  000022                          MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
007050  012605                                  MOV     (SP)+,R5        ;;POP STACK INTO R5
007052  012604                                  MOV     (SP)+,R4        ;;POP STACK INTO R4
007054  012603                                  MOV     (SP)+,R3        ;;POP STACK INTO R3
007056  012602                                  MOV     (SP)+,R2        ;;POP STACK INTO R2
007060  012601                                  MOV     (SP)+,R1        ;;POP STACK INTO R1
007062  012600                                  MOV     (SP)+,R0        ;;POP STACK INTO R0
007064  000002                                  RTI
```

```
1909                                    .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,@#$DB2O      ;;CALL THE ROUTINE
                                        ;*      RETURN                  ;;THE ADDRESS OF THE FIRST ASCIZ CHAR. IS ON THE STACK
 007066  104413                         $DB2O:  SAVREG                  ;;SAVE ALL REGISTERS
 007070  016601  000002                         MOV     2(SP),R1        ;;PICKUP THE POINTER TO LOW WORD
 007074  012705  007205                         MOV     #$OCTVL+13.,R5  ;;POINTER TO DATA TABLE
 007100  012704  000014                         MOV     #12.,R4         ;;DO ELEVEN CHARACTERS
 007104  012703  177770                         MOV     #^C7,R3         ;;MASK
 007110  012100                                 MOV     (R1)+,R0        ;;LOWER WORD
 007112  012101                                 MOV     (R1)+,R1        ;;HIGH WORD
 007114  005002                                 CLR     R2              ;;TERMINATOR
 007116  110245                         1$:     MOVB    R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
 007120  010002                                 MOV     R0,R2           ;;GET THIS DIGIT
 007122  005304                                 DEC     R4              ;;COUNT THIS CHARACTER
 007124  003007                                 BGT     3$              ;;BR IF NOT THE LAST DIGIT
 007126  001405                                 BEQ     2$              ;;BR IF IT IS THE LAST DIGIT
 007130  005205                                 INC     R5              ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
 007132  010566  000002                         MOV     R5,2(SP)        ;;ASCIZ CHAR. & PUT IT ON THE STACK
 007136  104414                                 RESREG                  ;;RESTORE ALL REGISTERS
 007140  000207                                 RTS     PC              ;;RETURN TO USER
 007142  006203                         2$:     ASR     R3              ;;POSITION THE MASK FOR THE LAST DIGIT
 007144  006001                         3$:     ROR     R1              ;;POSITION THE BINARY NUMBER FOR
 007146  006000                                 ROR     R0              ;;      THE NEXT OCTAL DIGIT
 007150  006001                                 ROR     R1
 007152  006000                                 ROR     R0
 007154  006001                                 ROR     R1
 007156  006000                                 ROR     R0
 007160  040302                                 BIC     R3,R2           ;;MASK OUT ALL JUNK
 007162  062702  000060                         ADD     #'0,R2          ;;MAKE THIS CHAR. ASCII
 007166  000753                                 BR      1$              ;;GO PUT IT IN THE DATA TABLE
 007170                                 $OCTVL: .BLKB   14.             ;;RESERVE DATA TABLE
```

```
1910                                    .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.
007206  010046                          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
007210  016600  000002                          MOV     2(SP),R0        ;;GET TRAP ADDRESS
007214  005740                          $TRAP1: TST     -(R0)           ;;BACKUP BY 2
007216  111000                                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
007220  006300                                  ASL     R0              ;;POSITION FOR INDEXING
007222  016000  007242                          MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
007226  000200                                  RTS     R0              ;;GO TO ROUTINE
                                        ;;THIS IS USE TO HANDLE THE ''GETPRI'' MACRO
007230  011646                          $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
007232  016666  000004  000002                  MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
007240  000002                                  RTI                     ;;RESTORE THE PSW
                                        .SBTTL  TRAP TABLE
                                        ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                        ;*BY THE ''TRAP'' INSTRUCTION.
                                        ;       ROUTINE
                                        ;       -------
007242  007230                          $TRPAD: .WORD   $TRAP2
007244  005432                                  $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
007246  006344                                  $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
007250  006320                                  $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
007252  006360                                  $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
007254  006546                                  $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
007256  006244                                  $TYPBN  ;;CALL=TYPBN    TRAP+6(104406)  TYPE BINARY (ASCII) NUMBER
007260  004356                                  $GTSWR  ;;CALL=GTSWR    TRAP+7(104407)  GET SOFT-SWR SETTING
007262  004306                                  $CKSWR  ;;CALL=CKSWR    TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
007264  004630                                  $RDCHR  ;;CALL=RDCHR    TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
007266  004760                                  $RDLIN  ;;CALL=RDLIN    TRAP+12(104412) TTY TYPEIN STRING ROUTINE
007270  006772                                  $SAVREG ;;CALL=SAVREG   TRAP+13(104413) SAVE R0-R5 ROUTINE
007272  007030                                  $RESREG ;;CALL=RESREG   TRAP+14(104414) RESTORE R0-R5 ROUTINE
```

```
1911                                    .SBTTL  POWER DOWN AND UP ROUTINES
                                        ;;*****************************************************************
                                        ;POWER DOWN ROUTINE
007274  012737  007452  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
007302  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
007310  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
007312  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
007314  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
007316  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
007320  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
007322  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
007324  017746  171610                  MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
007330  010637  007456                  MOV     SP,$SAVR6       ;;SAVE SP
007334  012737  007346  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
007342  000000                          HALT
007344  000776                          BR      .-2             ;;HANG UP
                                        ;;*****************************************************************
                                        ;POWER UP ROUTINE
007346  012737  007452  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
007354  013706  007456                  MOV     $SAVR6,SP       ;;GET SP
007360  005037  007456                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
007364  005237  007456          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
007370  001375                          BNE     1$              ;;OF  WORD
007372  012677  171542                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
007376  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
007400  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
007402  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
007404  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
007406  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
007410  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
007412  012737  007274  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
007420  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
007426  104401                          TYPE                    ;;REPORT THE POWER FAILURE
007430  007460          $PWRMG: .WORD   PWRMSG          ;;POWER FAIL MESSAGE POINTER
007432  012716                          MOV     (PC)+,(SP)      ;;RESTART AT START
007434  020000          $PWRAD: .WORD   START           ;;RESTART ADDRESS
007436  042766  000020  000002          BIC     #20,2(SP)       ;;CLEAR "T" BIT
007444  005037  001310                  CLR     $TBIT           ;;CLEAR THE "T" BIT FLAG
007450  000002                          RTI
007452  000000          $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
007454  000776                  BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
007456  000000          $SAVR6: 0                       ;;PUT THE SP HERE
1912 007460     012     015     040  PWRMSG: .ASCIZ  <12><15>? POWER FAILURE - RESTARTING ?<12><15>
     007463     120     117     127
     007466     105     122     040
     007471     106     101     111
     007474     114     125     122
     007477     105     040     055
     007502     040     122     105
     007505     123     124     101
     007510     122     124     111
     007513     116     107     040
     007516     012     015     000
1913                                    .EVEN
```

```
1915                                        .SBTTL  ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS
1916                                                .NLIST  BEX
1917 007522  125  116  105  EM1:    .ASCIZ  /UNEXPECTED CPU TRAP TO LOC. 004/
1918 007562  125  116  105  EM2:    .ASCIZ  /UNEXPECTED MEM. MGMT. TRAP TO LOC. 250/
1919 007631  115  105  115  EM10:   .ASCIZ  /MEMORY MGMT. ACCESS ABORT DID NOT OCCUR/
1920 007701  101  103  103  EM11:   .ASCIZ  /ACCESS ERROR DID NOT ABORT INSTRUCTION/
1921 007750  123  122  060  EM12:   .ASCIZ  /SR0 DID NOT REPORT ACCESS ERROR CORRECTLY/
1922 010022  123  122  062  EM13:   .ASCIZ  /SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR./
1923 010073  120  101  107  EM14:   .ASCIZ  /PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE/
1924 010154  120  101  107  EM15:   .ASCIZ  /PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE/
1925 010237  123  122  060  EM16:   .ASCIZ  /SR0 DID NOT REPORT PAGE LGTH. ABORT CORRECTLY/
1926 010315  123  122  060  EM21:   .ASCIZ  /SR0 OR SR2 CHANGED BY A SECOND ABORT/
1927 010362  123  122  060  EM22:   .ASCIZ  /SR0 OR SR2 WERE NOT "RESET" BY A RESET/
1928 010431  123  122  062  EM23:   .ASCIZ  /SR2 NOT TRACKING CORRECTLY/
1929 010464  104  111  104  EM24:   .ASCIZ  /DID NOT TRAP THRU KERNEL SPACE/
1930 010523  113  124  040  EM25:   .ASCIZ  /KT ERROR SERVICED ON ODD ADDR. ERROR/
1931 010570  123  122  060  EM26:   .ASCIZ  /SR0 OR SR2 CHANGED BY ODD ADDR. ERROR/
1932 010636  105  122  122  EM27:   .ASCIZ  /ERROR DURING "DOUBLE ERROR" (KT & ODD ADDR.)/
1933 010713  115  106  120  EM30:   .ASCIZ  /MFPI INSTRUCTION PUSHED WRONG DATA/
1934 010756  115  124  120  EM31:   .ASCIZ  /MTPI INSTRUCTION LOADED WRONG DATA/
1935 011021  123  124  101  EM32:   .ASCIZ  /STACK NOT PUSHED BY MFPI-MTPI/
1936 011057  113  105  122  EM33:   .ASCIZ  /KERNEL PAGE ACCESS INSTEAD OF USER: MFPI-MTPI/
1937 011135  115  056  115  EM34:   .ASCIZ  /M.M. ABORT IN KERNAL D-SPACE HAD WRONG CONDITION/
1938 011216  111  114  114  EM35:   .ASCIZ  /ILLEGAL MODE 10 NOT ABORTED/
1939 011252  123  122  060  EM36:   .ASCIZ  /SR0 DID NOT REPORT ILLEGAL MODE 10 CORRECTLY/
1940 011327  120  123  127  EM37:   .ASCIZ  /PSW CHANGED BY AN RTI IN USER MODE/
1941 011372  101  102  117  EM40:   .ASCIZ  /ABORT I KERNAL D-SPACE PICKED UP VECTOR FROM I-SPACE/
1942 011457  104  040  123  EM41:   .ASCIZ  /D SPACE ENABLE CIRCUITRY HAS FAILED/
1943 011523  111  116  103  EM42:   .ASCIZ  /INCORRECT STORE BY MTP INSTRUCTION/
1944 011566  124  122  111  EM43:   .ASCIZ  /TRIED TO REFERENCE NON-RESIDENT PAGE/
1945 011633  127  122  117  EM44:   .ASCIZ  /WRONG DATA FETCHED BY MFP INSTRUCTION/
1946 011701  111  114  114  EM45:   .ASCIZ  /ILLEGAL CSM DID NOT TRAP TO 10/
1947 011740  103  123  115  EM46:   .ASCIZ  /CSM DID NOT ENTER SUPERVISOR MODE/
1948 012002  103  123  115  EM47:   .ASCIZ  /CSM SET UP WRONG PREVIOUS MODE/
1949 012041  103  123  115  EM50:   .ASCIZ  /CSM SET UP STACK WRONG/
1950 012070  103  123  115  EM51:   .ASCIZ  /CSM PUSHED INCORRECT ARGUMENT/
1951 012126  103  123  115  EM52:   .ASCIZ  /CSM PUSHED WRONG PC/
1952 012152  103  123  115  EM53:   .ASCIZ  /CSM DID NOT CLEAR OLD PSW BITS <3:0>/
1953 012217  103  123  115  EM54:   .ASCIZ  /CSM ACCESSED WRONG SUPERVISOR SPACE/
1954 012263  103  123  115  EM55:   .ASCIZ  /CSM ABORTED WHEN IT SHOULD NOT HAVE/
1955 012327  103  123  115  EM56:   .ASCIZ  ?CSM FAILED TO INCREMENT/DECREMENT REGISTER PROPERLY?
1956 012413  103  123  115  EM57:   .ASCIZ  /CSM FAILED TO PUT PROPER ARGUMENT ON STACK/
1957 012466  123  122  062  EM58:   .ASCII  !SR2 LOCKED UP AN 11/34 COMPATIBLE ADDRESS THAT!<CRLF>
1958 012545  104  117  105          .ASCII  !DOES NOT MAKE IT COMPATIBLE WITH AN 11/70 SINCE!<CRLF>
1959 012625  124  110  105          .ASCIZ  !THE OPTIONAL M7095 ECO #8 IS MISSING!
1960                                        .EVEN
```

DATA HEADERS

```
1962                                        .SBTTL  DATA HEADERS
1963 012672  117  114  104  DH1:    .ASCIZ  /OLD PC  OLD PSW R6 WAS  CPUERR  TESTNO  ERRORPC/
1964 012752  117  114  104  DH2:    .ASCIZ  /OLD PC  OLD PSW R6 WAS  SR0     SR2     TESTNO  ERRORPC/
1965 013042  120  104  122  DH10:   .ASCIZ  /PDR 4   PSW     TESTNO  ERRORPC/
1966 013102  123  122  060  DH12:   .ASCIZ  /SR0 WAS EXPECTD PDR 4   PSW     TESTNO  ERRORPC/
1967 013162  123  122  062  DH13:   .ASCIZ  /SR2 WAS EXPECTD PDR 4   PSW     TESTNO  ERRORPC/
1968 013242  126  056  102  DH14:   .ASCIZ  /V.B.A.  KIPDR4  SR0 WAS SR2 WAS TESTNO  ERRORPC/
1969 013322  126  056  102  DH15:   .ASCIZ  /V.B.A.  KIPDR4  TESTNO  ERRORPC/
1970 013362  126  056  102  DH16:   .ASCIZ  /V.B.A.  KIPDR4  SR0 WAS EXPECTD TESTNO  ERRORPC/
1971 013442  126  056  102  DH17:   .ASCIZ  /V.B.A.  KIPDR4  SR2 WAS EXPECTD TESTNO  ERRORPC/
1972 013522  123  122  062  DH20:   .ASCIZ  /SR2 WAS EXPECTD TESTNO  ERRORPC/
1973 013562  106  111  122  DH21:   .ASCII  /FIRST ABORT     SECOND ABORT/<CRLF>
1974 013617  123  122  060          .ASCIZ  /SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO  ERRORPC/
1975 013677  123  122  060  DH22:   .ASCIZ  /SR0 WAS SR2 WAS TESTNO  ERRORPC/
1976 013737  120  123  127  DH24:   .ASCIZ  /PSW WAS R6 WAS  TESTNO  ERRORPC/
1977 013777  105  130  120  DH26:   .ASCII  /EXPECTED         RECEIVED/<CRLF>
1978 014031  123  122  060          .ASCIZ  /SR0     SR2     SR0 WAS SR2 WAS TESTNO  ERRORPC/
1979 014111  105  130  120  DH27:   .ASCII  /EXPECTED:/<CRLF>
1980 014123  120  123  127          .ASCII  /PSW     PC      SR0     SR2/<CRLF>
1981 014157  061  067  060          .ASCII  /170017  (3$+4)  020147  (3$)/<CRLF>
1982 014214  122  105  103          .ASCII  /RECEIVED:/<CRLF>
1983 014226  120  123  127          .ASCIZ  /PSW     PC      SR0     SR2     TESTNO  ERRORPC/
1984 014306  104  101  124  DH30:   .ASCII  /DATA    DATA/<CRLF>
1985 014323  105  130  120          .ASCIZ  /EXPECTD RECEIVD TESTNO  ERR PC/
1986 014362  124  105  123  DH32:   .ASCIZ  /TESTNO  ERR PC/
1987 014401  123  122  060  DH33:   .ASCIZ  /SR0 WAS SR2 WAS TESTNO  ERRORPC/
1988 014441  050  115  115  DH34:   .ASCIZ  /(MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC EXPECTING 020031/
1989 014532  123  122  060  DH36:   .ASCIZ  /SR0 WAS EXPECTD TESTNO  ERRORPC/
1990 014572  120  123  127  DH37:   .ASCIZ  /PSW WAS EXPECTD TESTNO  ERRORPC/
1991 014632  050  120  123  DH40:   .ASCIZ  /(PSW)   TESTNO  ERRORPC EXPECTING XXX340/
1992 014703  105  122  122  DH41:   .ASCII  /ERROR   AUTOI-D VIRTUAL/<CRLF>
1993 014733  122  105  107          .ASCIZ  /REGISTR REGISTR ADDRESS TESTNO  PC AT ABORT/
1994 015007  107  104  104  DH42:   .ASCIZ  /GDDATA  STORED  TESTNO  ERRORPC/
1995 015047  050  115  115  DH43:   .ASCIZ  /(MMR0)  (MMR1)  (MMR2)  TESTNO  ERRORPC/
1996 015117  105  130  120  DH44:   .ASCIZ  /EXPECTD (PSW)   TESTNO  ERRORPC/
1997 015157  117  114  104  DH45:   .ASCIZ  /OLDPSW  TESTNO  ERRORPC/
1998 015207  124  105  123  DH55:   .ASCIZ  /TESTNO  ERR PC  R0 EXP  R0 RCV/
1999 015246  124  105  123  DH57:   .ASCIZ  /TESTNO  ERR PC  ARGEXP  ARGRCV/
2000                                        .EVEN
```

```
2002                                            .SBTTL  DATA TABLES
2003 015306  001260  001262  001256  DT1:      .WORD   TRAPPC,TRAPPS,WASR6,CPUERR,TESTNO,$ERRPC,0
2004 015324  001260  001262  001256  DT2:      .WORD   TRAPPC,TRAPPS,WASR6,WASSR0,WASSR2,TESTNO,$ERRPC,0
2005 015344  001166  001176  001254  DT10:     .WORD   $REG2,$TMP0,TESTNO,$ERRPC,0
2006 015356  001264  001170  001166  DT12:     .WORD   WASSR0,$REG3,$REG2,$TMP0,TESTNO,$ERRPC,0
2007 015374  001270  001172  001166  DT13:     .WORD   WASSR2,$REG4,$REG2,$TMP0,TESTNO,$ERRPC,0
2008 015412  001162  001172  001264  DT14:     .WORD   $REG0,$REG4,WASSR0,WASSR2,TESTNO,$ERRPC,0
2009 015430  001162  001172  001254  DT15:     .WORD   $REG0,$REG4,TESTNO,$ERRPC,0
2010 015442  001162  001172  001264  DT16:     .WORD   $REG0,$REG4,WASSR0,$REG2,TESTNO,$ERRPC,0
2011 015460  001162  001172  001270  DT17:     .WORD   $REG0,$REG4,WASSR2,$REG3,TESTNO,$ERRPC,0
2012 015476  001270  001164  001254  DT20:     .WORD   WASSR2,$REG1,TESTNO,$ERRPC,0
2013 015510  001176  001202  001264  DT21:     .WORD   $TMP0,$TMP2,WASSR0,WASSR2,TESTNO,$ERRPC,0
2014 015526  001264  001270  001254  DT22:     .WORD   WASSR0,WASSR2,TESTNO,$ERRPC,0
2015 015540  001164  001166  001254  DT24:     .WORD   $REG1,$REG2,TESTNO,$ERRPC,0
2016 015552  001162  001164  001264  DT26:     .WORD   $REG0,$REG1,WASSR0,WASSR2,TESTNO,$ERRPC,0
2017 015570  001164  001170  001264  DT27:     .WORD   $REG1,$REG3,WASSR0,WASSR2,TESTNO,$ERRPC,0
2018 015606  035336  001176  001254  DT30:     .WORD   ACSMSP,$TMP0,TESTNO,$ERRPC,0
2019 015620  001254  001116  000000  DT32:     .WORD   TESTNO,$ERRPC,0
2020 015626  001164  001166  001170  DT34:     .WORD   $REG1,$REG2,$REG3,TESTNO,$ERRPC,0
2021 015642  001264  001164  001254  DT36:     .WORD   WASSR0,$REG1,TESTNO,$ERRPC,0
2022 015654  001162  001254  001116  DT40:     .WORD   $REG0,TESTNO,$ERRPC,0
2023 015664  001264  001266  001270  DT41:     .WORD   WASSR0,WASSR1,WASSR2,TESTNO,BADPC,0
2024 015700  001162  001164  001254  DT42:     .WORD   $REG0,$REG1,TESTNO,$ERRPC,0
2025 015712  001162  001164  001166  DT43:     .WORD   $REG0,$REG1,$REG2,TESTNO,$ERRPC,0
2026 015726  001264  001266  001270  DT45:     .WORD   WASSR0,WASSR1,WASSR2,TESTNO,$ERRPC,0
2027 015742  001170  035334  001254  DT46:     .WORD   $REG3,ACSMPS,TESTNO,$ERRPC,0
2028 015754  001162  035326  001254  DT47:     .WORD   $REG0,CSM1ST,TESTNO,$ERRPC,0
2029 015766  001176  035330  001254  DT50:     .WORD   $TMP0,CSM2ND,TESTNO,$ERRPC,0
2030 016000  035332  001254  001116  DT52:     .WORD   CSM3RD,TESTNO,$ERRPC,0
2031 016010  001254  001116  001176  DT55:     .WORD   TESTNO,$ERRPC,$TMP0,$REG0,0
2032 016022  001254  001116  001176  DT57:     .WORD   TESTNO,$ERRPC,$TMP0,$TMP1,0
```

DATA FORMAT BYTE STRINGS

```
2034                                            .SBTTL  DATA FORMAT BYTE STRINGS
2035 016074  000  000  000  DF1:    .BYTE   0,0,0,0,0
2036 016041  000  000  000  DF2:    .BYTE   0,0,0,0,0,0,0
2037 016050  000  000  000  DF3:    .BYTE   0,0,0,0
2038 016054  000  000  000  DF5:    .BYTE   0,0,0
2039 016057  000  000  000  DF12:   .BYTE   0,0,0,0,0,0
2040 016065  000  000       DF32:   .BYTE   0,0
2041                                            .EVEN
```

```
                                        .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.
                                ;*
                                ;;*****************************************************************
016070  005227                  TIMERR: INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME THRU
016072  177777                  TIMFLG: .WORD   -1              ;NEGATIVE ONE FOR "HAVE ENTERED" FLAG
016074  001401                          BEQ     1$              ;BRANCH IF FIRST TIME IN
016076  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.
016100  012637  001260          1$:     MOV     (KSP)+,TRAPPC   ;SAVE PC+2 AT TIME OF ABORT
016104  012637  001262                  MOV     (KSP)+,TRAPPS   ;SAVE PS AT TIME OF ABORT
016110  010637  001256                  MOV     KSP,WASR6       ;SAVE STACK POINTER VALUE
016114  104001                          ERROR   +1              ;UNEXPECTED TRAP OR ABORT TO LOC. 4
016116  012737  177777  016072        MOV     #-1,TIMFLG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
016124  005037  177766                  CLR     CPUERR          ;CLEAR THE CPU ERROR REGISTER
016130  013746  001262                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
016134  013746  001260                  MOV     TRAPPC,-(KSP)
016140  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.
                                        ;*
                                        ;;*****************************************************************
016142  005227                        MGMERR: INC     (PC)+           ;MAKE FLAG ZERO IF FIRST TIME THRU
016144  177777                        MGMFLG: .WORD   -1              ;NEGATIVE ONE FOR 'HAVE ENTERED' FLAG
016146  001401                                BEQ     1$              ;BRANCH IF FIRST TIME IN
016150  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.
016152  012637  001260                1$:     MOV     (KSP)+,TRAPPC   ;SAVE PC+2 AT TIME OF ABORT
016156  012637  001262                        MOV     (KSP)+,TRAPPS   ;SAVE PS AT TIME OF ABORT
016162  010637  001256                        MOV     KSP,WASR6       ;SAVE STACK POINTER VALUE
016166  013737  177572  001264              MOV     SR0,WASSR0      ;SAVE CONTENTS OF KT STATUS REG. 0
016174  013737  177576  001270              MOV     SR2,WASSR2      ;SAVE CONTENTS OF KT STATUS REG. 2
016202  042737  160000  177572              BIC     #160000,SR0     ;CLEAR ERROR BITS IN STATUS REG 0
016210  104002                            ERROR   +2              ;UNEXPECTED TRAP OR ABORT TO LOC. 250
016212  012737  177777  016144              MOV     #-1,MGMFLG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
016220  013746  001262                        MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
016224  013746  001260                        MOV     TRAPPC,-(KSP)
016230  000006                            RTT                     ;RETURN FROM INTERRUPT OR ABORT.

                                        ;THE NEXT LOCATIONS ARE RESERVED FOR ANY PERMANENT PATCHES:
016232  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
016326  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
```

```
2102                                    ;*      *****   STARTING POINT OF TEST  *****
2103                                    ;*      *****   STARTING ADDRESS OF 200 *****
2104          020000                    .=20000
2105
2106 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  003042  000020       MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
      020026  012737  000340  000022       MOV     #340,@#IOTVEC+2 ;;LEVEL 7
      020034  012737  003270  000030       MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
      020042  012737  000340  000032       MOV     #340,@#EMTVEC+2 ;;LEVEL 7
      020050  012737  007206  000034       MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
      020056  012737  000340  000036       MOV     #340,@#TRAPVEC+2;LEVEL 7
      020064  012737  007274  000024       MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
      020072  012737  000340  000026       MOV     #340,@#PWRVEC+2 ;;LEVEL 7
      020100  013737  036054  036046       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  036316  000014       MOV     #$RTRN,@#TBITVEC ;;SET "T" BIT VECTOR TO $RTRN
      020126  012737  000340  000016       MOV     #340,@#TBITVEC+2 ;;LEVEL 7
      020134  012737  000002  036316       MOV     #RTI,$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$,-(SP)      ;;AND PC
      020156  000006                       RTT                     ;;TRY THE RTT
      020160  012737  000006  036316 64$:  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
```

```
        020322  001403                          BEQ     70$             ;;YES,USE NON-APT SWITCH
        020324  012737  001246  001140          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
        020332                          70$:
2107                                    .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  036252  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$:  .ASCIZ  <CRLF>#CKKTBDO 11/44 MEM MGMT PRT B#<CRLF>
        020456                          71$:
2108    020456  104401  020464                  TYPE    ,76$            ;;TYPE ASCIZ STRING
        020462  000436                          BR      75$             ;;GET OVER THE ASCIZ
                                        ;;76$:  .ASCIZ  #EOP MESSAGES WILL PRINT EVERY 64 PASSES OR ABOUT 11 SECONDS#
        020560                          75$:
2109    020560                          LOOP:
2110    020560  012706  001100                  MOV     #STACK,KSP      ;INITIALIZE THE STACK POINTER
2111    020564  012737  040000  177776          MOV     #40000,PSW      ;TURN ON SUPERVISOR MODE
2112    020572  012706  000700                  MOV     #SUPSTK,SSP     ;INITIALIZE SUPER. STACK POINTER
2113    020576  012737  140000  177776          MOV     #140000,PSW     ;TURN ON USER MODE
2114    020604  012706  000600                  MOV     #USESTK,USP     ;INITIALIZE USER STACK POINTER
2115    020610  005037  177776                  CLR     PSW             ;RETURN TO KERNAL MODE
2116    020614  012737  016070  000004          MOV     #TIMERR,ERRVEC  ;LOAD CPU SERVICE ROUTINE INTO TRAP VECTOR
2117    020622  012737  000340  000006          MOV     #340,ERRVEC+2   ;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
2118    020630  012737  016142  000250          MOV     #MGMERR,MMVEC   ;LOAD MEMORY MANAGENT ROUTINE INTO VECTOR
2119    020636  012737  000340  000252          MOV     #340,MMVEC+2    ;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
2120    020644  012700  177777                  MOV     #-1,R0          ;PUT -1 INTO R0 TO INITIALIZE FLAGS
2121    020650  010037  016072                  MOV     R0,TIMFLG       ;INITIALIZE CPU ERROR FLAG
2122    020654  010037  016144                  MOV     R0,MGMFLG       ;INITIALIZE MEMORY MANAGEMENT ERROR FLAG
2123    020660  012737  000340  001274          MOV     #340,TBITPS     ;INITIALIZE LOG THAT HOLDS T-BIT PSW
2124    020666  005037  177572          INIT:   CLR     MMR0            ;BE SURE MEM. MGMT IS OFF TO START WITH,
2125    020672  012737  000020  172516          MOV     #BIT4,MMR3      ;BUT TURN ON 22-BIT ADDRESSING MODE
2126    020700  004737  002060                  JSR     PC,APRINIT      ;JUMP TO PAR/PDR INIT ROUTINE
```

```
2128                            .SBTTL  GROUP 1   ACCESS CONTROL FIELD/SPECIAL ABORTS TESTS
2129                            ;/.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\
2130                            ;;*****************************************************************
2131                            ;*
2132                            ;*      THE NEXT THREE (3) TESTS CAUSE MEMORY MANAGEMENT ERRORS
2133                            ;*      TO CHECK THE ABILITY OF STATUS REGISTER 0 TO RECORD KT
2134                            ;*      ERRORS AND THE ABILITY OF STATUS REGISTER 2 TO LOCK UP THE
2135                            ;*      VIRTUAL ADDR. OF THE INSTRUCTION THAT CAUSED THE ERROR.
2136                            ;*      THE BITS OF SR2 ARE CHECKED AND BITS <15:13>, <6:5>, AND <3:0>
2137                            ;*      ARE CHECKED IN SR0.  SO THE SR0 AND SR2 LOGIC AND THE
2138                            ;*      KT ERROR LOGIC ARE CHECKED.
2139                            ;*
2140                            ;;*****************************************************************
2141                            ;/.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\
```

```
                                        .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.
                                        ;*
                                        ;;*****************************************************************
    020704  000004                      TST1:   SCOPE
2151 020706  012700  000600             1$:     MOV     #600,R0         ;LOAD DATA FOR PAR'S INTO R0
2152 020712  010037  172346                     MOV     R0,KIPAR3       ;MAP KERNEL PAR'S 3&4 TO 12-16K
2153 020716  010037  172350                     MOV     R0,KIPAR4
2154 020722  010037  172246                     MOV     R0,SIPAR3       ;MAP SUPERVISOR PAR'S 3&4 TO 12-16K
2155 020726  010037  172250                     MOV     R0,SIPAR4
2156 020732  010037  177646                     MOV     R0,UIPAR3       ;MAP USER PAR'S 3&4 TO 12-16K
2157 020736  010037  177650                     MOV     R0,UIPAR4
2158 020742  012700  060000                     MOV     #60000,R0       ;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
2159 020746  012701  100000                     MOV     #100000,R1      ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
2160 020752  012703  100011                     MOV     #100011,R3      ;LOAD R3 WITH WHAT SR0 SHOULD READ - N.R., KERNEL, PG.4
2161 020756  012702  077400                     MOV     #77400,R2       ;LOAD ACF=0 (NON-RESIDENT) PDR VALUE IN R2
2162 020762  012737  021036  000250     2$:     MOV     #5$,MMVEC       ;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
2163 020770  010237  172310                     MOV     R2,KIPDR4       ;LOAD ACF TEST VALUE INTO KIPDR4
2164 020774  010237  172210                     MOV     R2,SIPDR4       ;LOAD ACF TEST VALUE INTO SIPDR4
2165 021000  010237  177610                     MOV     R2,UIPDR4       ;LOAD ACF TEST VALUE INTO UIPDR4
2166 021004  012737  021020  001110             MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2167 021012  012737  000001  177572             MOV     #1,MMR0         ;TURN ON MEMORY MANAGEMENT
2168 021020  005010                     3$:     CLR     (R0)            ;CLEAR PHYS. LOC. 60000 USING PDR3
2169 021022  013737  177776  001176             MOV     PSW,$TMP0       ;SAVE PSW IN CASE OF ERROR
2170 021030  005211                     4$:     INC     (R1)            ;TRY TO REF. IT USING PDR4 - SHOULD TRAP TO 5$
2171 021032  104003                             ERROR   +3              ;MEM. MGMT. ABORT DID NOT OCCUR
2172                                                                     ;FOR TIGHTER SCOPE LOOP
2173                                                                     ;REPLACE ERROR CALL WITH
2174                                                                     ;'BR 3$" = 000772
2175 021034  000425                             BR      8$              ;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
2176 021036  062706  000004             5$:     ADD     #4,SP           ;RESTORE STACK POINTER
2177 021042  005710                             TST     (R0)            ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
2178 021044  001401                             BEQ     6$              ;BRANCH IF YES
2179 021046  104004                             ERROR   +4              ;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
2180                                                                     ;FOR TIGHTER SCOPE LOOP
2181                                                                     ;REPLACE ERROR CALL WITH
2182                                                                     ;'BR 3$" = 000764
2183 021050  013737  177572  001264     6$:     MOV     SR0,WASSR0      ;READ STATUS REGISTER 0
2184 021056  013737  177576  001270             MOV     SR2,WASSR2      ;READ STATUS REGISTER 2
2185 021064  020337  001264                     CMP     R3,WASSR0       ;DID SR0 REPORT NON-RESIDENT ERROR CORRECTLY?
2186 021070  001401                             BEQ     7$              ;BRANCH IF YES
2187 021072  104005                             ERROR   +5              ;SR0 DID NOT REPORT NON-RES. ERROR CORRECTLY
2188                                                                     ;FOR TIGHTER SCOPE LOOP
2189                                                                     ;REPLACE ERROR CALL WITH
2190                                                                     ;'BR 3$" = 000752
2191 021074  012704  021030             7$:     MOV     #4$,R4          ;LOAD R4 WITH WHAT SR2 SHOULD READ
2192 021100  020437  001270                     CMP     R4,WASSR2       ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4$)?
2193 021104  001401                             BEQ     8$              ;BRANCH IF YES
2194 021106  104006                             ERROR   +6              ;SR2 DID NOT LOCK VIRTUAL ADDR. OF NON-RES. ERROR
2195                                                                     ;FOR TIGHTER SCOPE LOOP
2196                                                                     ;REPLACE ERROR CALL WITH
```

```
2197                                                                    ;'BR 3$'' = 000744
2198 021110  005037  177572          8$:    CLR     MMR0               ;TURN OFF MEMORY MANAGEMENT
2199 021114  032737  140000  001176         BIT     #140000,$TMP0      ;HAS ACF=0&4 BEEN TESTED IN USER YET
2200 021122  001006                         BNE     11$                ;BRANCH IF YES
2201 021124  012703  100151                MOV     #100151,R3         ;LOAD R3 WITH WHAT SR0 SHOULD READ - N.R., USER, PG.4
2202 021130  012737  140000  177776         MOV     #140000,PSW        ;GO TO USER MODE
2203 021136  000711                         BR      2$                 ;REPEAT TEST IN USER MODE
2204 021140  022737  040000  001176  11$:    CMP     #40000,$TMP0       ;HAS ACF=0&4 BEEN TESTED IN SUPERVISOR YET
2205 021146  001006                         BNE     9$                 ;BRANCH IF YES
2206 021150  012703  100051                MOV     #100051,R3         ;LOAD R3 WITH WHAT SR0 SHOULD READ - N.R., SUPERVISOR, PG.4
2207 021154  012737  040000  177776         MOV     #40000,PSW         ;GO TO SUPERVISOR MODE
2208 021162  000677                         BR      2$                 ;REPEAT TEST IN SUPERVISOR MODE
2209 021164  022702  077404          9$:    CMP     #77404,R2          ;HAS ACF=4 BEEN TESTED YET?
2210 021170  001407                         BEQ     10$                ;BRANCH IF YES
2211 021172  012702  077404                MOV     #77404,R2          ;THEN LOAD ACF=4 (NON-RES) PDR VALUE IN R2
2212 021176  012703  100011                MOV     #100011,R3         ;LOAD R3 WITH WHAT SR0 SHOULD READ-N.R.,KERNEL,PG. 4
2213 021202  005037  177776                CLR     PSW                ;GO BACK TO KERNEL MODE
2214 021206  000665                         BR      2$                 ;GO BACK & TEST ACF=4 IN SAME MODE
2215 021210  005037  177776          10$:   CLR     PSW                ;GO BACK TO KERNEL MODE BEFORE LEAVING
2216 021214  012737  016142  000250         MOV     #MMGMERR,MMVEC     ;RESTORE ADDRESS OF NORMAL MEMORY
2217                                                                    ;MANAGEMENT ERROR ROUTINE TO MMVEC
```

```
2226                                    .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.
                                        ;*
                                        ;*****************************************************************
    021222  000004                      TST2:   SCOPE
2227 021224                             1$:                                     ;KERNEL, SUPERVISOR AND USER PAR'S 3 & 4,
2228                                                                            ;AND PDR 3 ARE SETUP FROM LAST TEST
2229 021224  012700  060000                    MOV     #60000,R0               ;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
2230 021230  012701  100000                    MOV     #100000,R1              ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
2231 021234  012703  020011                    MOV     #20011,R3               ;LOAD R3 WITH WHAT SR0 SHOULD READ - R/O, KERNEL, PG.4
2232 021240  012702  077402                    MOV     #77402,R2               ;LOAD ACF=2 (READ-ONLY) PDR VALUE IN R2
2233 021244  012737  021316  000250  2$:      MOV     #5$,MMVEC               ;POINT MEM. MGMT. TRAP VECTOR TO 5$ BELOW
2234 021252  010237  172310                    MOV     R2,KIPDR4               ;LOAD ACF=2 INTO KIPDR4
2235 021256  010237  172210                    MOV     R2,SIPDR4               ;LOAD ACF=2 INTO SIPDR4
2236 021262  010237  177610                    MOV     R2,UIPDR4               ;LOAD ACF=2 INTO UIPDR4
2237 021266  012737  021300  001110            MOV     #3$,$LPERR              ;SET LOOP ON ERROR POINTER TO 3$
2238 021274  005237  177572                    INC     MMR0                    ;TURN ON MEMORY MANAGEMENT
2239 021300  005010                    3$:     CLR     (R0)                    ;CLEAR PHYS. LOC. 60000 USING PDR3
2240 021302  013737  177776  001176            MOV     PSW,$TMP0               ;SAVE PSW IN CASE OF ERROR
2241 021310  005211                    4$:     INC     (R1)                    ;TRY TO WRITE USING PDR4 - SHOULD TRAP TO 5$
2242 021312  104003                            ERROR   +3                      ;MEM. MGMT. ABORT DID NOT OCCUR
2243                                                                            ;FOR TIGHTER SCOPE LOOP
2244                                                                            ;REPLACE ERROR CALL WITH
2245                                                                            ;'BR 3$" = 000772
2246 021314  000425                            BR      8$                      ;BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
2247 021316  062706  000004            5$:     ADD     #4,SP                   ;RESTORE STACK POINTER
2248 021322  005710                            TST     (R0)                    ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
2249 021324  001401                            BEQ     6$                      ;BRANCH IF YES
2250 021326  104004                            ERROR   +4                      ;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
2251                                                                            ;FOR TIGHTER SCOPE LOOP
2252                                                                            ;REPLACE ERROR CALL WITH
2253                                                                            ;'BR 3$" = 000764
2254 021330  013737  177572  001264  6$:      MOV     SR0,WASSR0              ;READ STATUS REG. 0
2255 021336  013737  177576  001270            MOV     SR2,WASSR2              ;READ STATUS REG. 2
2256 021344  020337  001264                    CMP     R3,WASSR0               ;DID SR0 REPORT READ-ONLY ERROR CORRECTLY?
2257 021350  001401                            BEQ     7$                      ;BRANCH IF YES
2258 021352  104005                            ERROR   +5                      ;SR0 DID NOT REPORT R/O ERROR CORRECTLY
2259                                                                            ;FOR TIGHTER SCOPE LOOP
2260                                                                            ;REPLACE ERROR CALL WITH
2261                                                                            ;'BR 3$" = 000752
2262 021354  012704  021310            7$:     MOV     #4$,R4                  ;LOAD R4 WITH WHAT SR2 SHOULD READ
2263 021360  020437  001270                    CMP     R4,WASSR2               ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4$)?
2264 021364  001401                            BEQ     8$                      ;BRANCH IF YES
2265 021366  104006                            ERROR   +6                      ;SR2 DID NOT LOCKUP VIRTUAL ADDR. OF R/O ERROR
2266                                                                            ;FOR TIGHTER SCOPE LOOP
2267                                                                            ;REPLACE ERROR CALL WITH
2268                                                                            ;'BR 3$" = 000744
2269 021370  005037  177572            8$:     CLR     MMR0                    ;TURN OFF MEMORY MANAGEMENT
2270 021374  032737  140000  001176            BIT     #140000,$TMP0           ;HAS ACF=2 BEEN TESTED IN USER MODE?
2271 021402  001006                            BNE     11$                     ;BRANCH IF YES
2272 021404  012703  020151                    MOV     #20151,R3               ;LOAD R3 WITH WHAT SR0 SHOULD READ-R/O, USER, PG.4
```

TEST # 2 - READ-ONLY ABORT TEST (ACF=2)

```
2273 021410  012737  140000  177776          MOV   #140000,PSW    ;GO TO USER MODE
2274 021416  000712                          BR    2$            ;REPEAT TEST IN USER MODE
2275 021420  022737  040000  001176  11$:    CMP   #40000,$TMP0  ;HAS ACF=2 BEEN TESTED IN SUPERVISOR MODE?
2276 021426  001006                          BNE   9$            ;BRANCH IF YES
2277 021430  012703  020051                  MOV   #20051,R3     ;LOAD R3 WITH WHAT SR0 SHOULD READ-R/O, SUPERVISOR, PG.4
2278 021434  012737  040000  177776          MOV   #40000,PSW    ;GO TO SUPERVISOR MODE
2279 021442  000700                          BR    2$            ;REPEAT TEST IN SUPERVISOR MODE
2280 021444  005037  177776          9$:     CLR   PSW           ;GO BACK TO KERNEL MODE BEFORE LEAVING
2281 021450  012737  016142  000250          MOV   #MGMERR,MMVEC ;RESTORE ADDRESS OF NORMAL MEMORY
2282                                                              ;MANAGEMENT ERROR ROUTINE TO MMVEC.
```

```
2294                                    .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 IF M7095 ECO #8 IS MISSIGN.  IF THE ECO IS
                                        ;*      INSTALLED, THE ADDRESS LOCKED UP WILL BE THAT OF THE UPDATED PC.
                                        ;*
                                        ;;*****************************************************************
    021456  000004                     TST3:   SCOPE
2295 021460  012737  021502  001110    1$:     MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
2296 021466  012737  021522  000250            MOV     #3$,MMVEC       ;LOAD MEM. MGMT. TRAP VECTOR WITH 3$
2297 021474  012737  000001  177572            MOV     #1,MMR0         ;TURN ON MEMORY MANAGEMENT
2298 021502  012737  100000  177776    10$:    MOV     #100000,PSW     ;SET 10 IN PSW CURRENT MODE BITS
2299 021510  104030                    2$:     ERROR   +30             ;ILLEGAL MODE 10 NOT ABORTED
2300                                            ;FOR A TIGHTER SCOPE LOOP, REPLACE ERROR CALL WITH 'BR 10$" = 000774
2301 021512  012737  016142  000250            MOV     #MGMERR,MMVEC   ;RESTORE MEM. MGMT. ABORT VECTOR
2302 021520  000441                            BR      5$              ;BRANCH AROUND SR0 & SR2 CHECKS
2303 021522  013737  177576  001270    3$:     MOV     SR2,WASSR2      ;READ CONTENTS OF SR2
2304 021530  012737  016142  000250            MOV     #MGMERR,MMVEC   ;RESTORE MEM. MGMT. ABORT VECTOR
2305 021536  012706  001100                    MOV     #KERSTK,SP      ;RESTORE STACK POINTER
2306 021542  022737  100103  177572            CMP     #100103,SR0     ;DID SR0 REPORT ILLEGAL MODE CORRECTLY?
2307 021550  001406                            BEQ     4$              ;BRANCH IF YES
2308 021552  012701  100103                    MOV     #100103,R1      ;LOAD EXPECTED CONTENTS OR SR0 INTO R1
2309 021556  013737  177572  001264            MOV     SR0,WASSR0      ;READ CONTENTS OF SR0
2310 021564  104031                            ERROR   +31             ;SR0 DID NOT REPORT NR ABORT, PG=1, MODE=10
2311                                            ;FOR TIGHTER SCOPE LOOP, REPLACE ERROR CALL WITH 'BR 10$" = 000746
2312 021566  022737  021502  001270    4$:     CMP     #10$,WASSR2     ;DID SR2 LOCKUP VIRT. ADDR OF ABORT     ;DPM002
2313 021574  001410                            BEQ     45$             ;BRANCH IF OK BUT NOT FOR ECO #8        ;DPM002
2314 021576  022737  021510  001270            CMP     #2$,WASSR2      ;DID SR2 LOCKUP VIRT. ADDR OF ABORT     ;DPM002
2315 021604  001407                            BEQ     5$              ;BRANCH IF ANSWER IS OK                 ;DPM002
2316 021606  012701  021510                    MOV     #2$,R1          ;MOVE ERROR ADDRESS TO R1 FOR ERROR CALL;DPM002
2317 021612  104013                            ERROR   +13             ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ILL. MODE INST.
2318                                            ;FOR TIGHTER SCOPE LOOP, REPLACE ERROR CALL WITH 'BR 10$" = 000733
2319 021614  000403                            BR      5$              ;BRANCH OVER 2ND ERROR CALL
2320 021616  012701  021502            45$:    MOV     #10$,R1         ;MOVE ADDRESS TO R1 FOR ERROR CALL      ;DPM002
2321 021622  104054                            ERROR   +54             ;SR2 LOCKED UP AN 11/34 COMPATIBLE      ;DPM002
2322                                            ;ADDRESS THAT DOES NOT MAKE IT COMPA-   ;DPM002
2323                                            ;TIBLE WITH AN 11/70 SINCE THE OPTIONAL ;DPM002
2324                                            ;M7095 ECO #8 IS MISSING                ;DPM002
2325                                            ;FOR TIGHTER SCOPE LOOP, REPLACE ERROR CALL WITH 'BR 10$" - 000727
2326 021624  042737  160000  177572    5$:     BIC     #160000,SR0     ;CLEAR POSSIBLE ERROR BITS IN SR0
```

TEST # 3 - TEST ILLEGAL MODE ''10''

```
2327                    ;;*****************************************************************
2328                    ;*
2329                    ;*      THE NEXT TWO (2) TESTS WILL BE CHECKING THE PAGE LENGTH
2330                    ;*      COMPARATORS AND SOME MORE OF THE KT ERROR DETECTION
2331                    ;*      AND STATUS LOGIC.  THE PAGE LENGTH FIELD (PLF) IN KERNEL
2332                    ;*      PDR 4 IS VARIED AND FOR EVERY PLF, THREE (3) VIRTUAL
2333                    ;*      ADDRESSES ARE READ.  WHILE USING BOTH UPWARD & DOWNWARD PAGE
2334                    ;*      EXPANSION, ONE OF THOSE THREE VIRTUAL ADDRESSES WILL CAUSE A
2335                    ;*      ''PAGE LENGTH ABORT'' WHILE THE OTHER TWO WON'T.
2336                    ;*
2337                    ;*      STATUS REGISTER 0 & 2 ARE CHECKED WHEN THE PAGE LENGTH
2338                    ;*      ABORT DOES OCCUR TO SEE THAT THE ABORT IS REPORTED AND THAT
2339                    ;*      THE VIRTUAL ADDRESS OF THE INSTRUCTION THAT CAUSED THE ABORT
2340                    ;*      IS LOCKED UP.
2341                    ;*
2342                    ;*****************************************************************
```

```
2354                                    .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).
                                        ;*
                                        ;;*****************************************************************
    021632  000004                      TST4:   SCOPE
2355 021634  012737  077406  172306     1$:     MOV     #77406,KIPDR3   ;MAKE SURE PDR3 IS DESCRIBED AS R/W
2356 021642  012737  077406  172312             MOV     #77406,KIPDR5   ;MAKE SURE PDR5 IS DESCRIBED AS R/W
2357 021650  012700  100000                     MOV     #100000,R0      ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
2358 021654  012704  000406                     MOV     #406,R4         ;LOAD FIRST PDR VALUE IN R4 (PLF=1, ACF=6)
2359 021660  012737  022052  000250     2$:     MOV     #9$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
2360 021666  010437  172310                     MOV     R4,KIPDR4       ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
2361 021672  012737  021700  001110             MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2362 021700  012706  001100             3$:     MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
2363 021704  011001                             MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA < PLF - NO ABORT)
2364 021706  062700  000100                     ADD     #100,R0         ;FORM NEXT VIRTUAL ADDRESS IN R0
2365 021712  012737  021720  001110             MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
2366 021720  012706  001100             4$:     MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
2367 021724  011001                             MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
2368 021726  062700  000100                     ADD     #100,R0         ;FORM NEXT VIRTUAL ADDR IN R0
2369 021732  020027  117700                     CMP     R0,#117700      ;HAVE ALL PLF'S BEEN TESTED YET?
2370 021736  001470                             BEQ     10$             ;BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED
2371 021740  012737  021754  001110             MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
2372 021746  012737  021762  000250             MOV     #6$,MMVEC       ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
2373 021754  011001             5$:     MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA > PLF - ABORT TO 6$)
2374 021756  104010                             ERROR   +10             ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
2375                                                                    ;FOR TIGHTER SCOPE LOOP
2376                                                                    ;REPLACE ERROR CALL WITH
2377                                                                    ;'BR 5$'' = 000776
2378 021760  000424                             BR      8$              ;BRANCH AROUND ABORT CHECKS
2379 021762  012706  001100             6$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER FOLLOWING ABORT
2380 021766  013737  177572  001264             MOV     SR0,WASSR0      ;READ M.M. STATUS REG. 0
2381 021774  013737  177576  001270             MOV     SR2,WASSR2      ;READ M.M. STATUS REG. 2
2382 022002  012702  040011                     MOV     #40011,R2       ;PUT EXPECTED SR0 CONTENTS IN R2
2383 022006  020237  001264                     CMP     R2,WASSR0       ;DID SR0 REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
2384 022012  001401                             BEQ     7$              ;BRANCH IF YES
2385 022014  104011                             ERROR   +11             ;SR0 DID NOT REPORT PG. LENGTH ABORT CORRECTLY
2386                                                                    ;FOR TIGHTER SCOPE LOOP
2387                                                                    ;REPLACE ERROR CALL WITH
2388                                                                    ;'BR 5$'' = 000757
2389 022016  012703  021754             7$:     MOV     #5$,R3          ;PUT EXPECTED SR2 CONTENTS IN R3
2390 022022  020337  001270                     CMP     R3,WASSR2       ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
2391 022026  001401                             BEQ     8$              ;BRANCH IF YES
2392 022030  104012                             ERROR   +12             ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
2393                                                                    ;FOR TIGHTER SCOPE LOOP
2394                                                                    ;REPLACE ERROR CALL WITH
2395                                                                    ;'BR 5$'' = 000751
2396 022032  042737  160000  177572     8$:     BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
2397 022040  062704  000400                     ADD     #400,R4         ;FORM NEXT PLF VALUE FOR KIPDR4
```

TEST # 4 - PAGE LENGTH FAULTS-UPWARD EXPANSION

```
2398 022044  162700  000100              SUB     #100,R0         ;FORM FIRST VIRT. ADDR FOR THAT PLF VALUE
2399 022050  000703                      BR      2$              ;BRANCH BACK AND ACCESS 3 VBA'S FOR
2400                                                             ;THE PLF VALUE JUST FORMED
2401 022052  012637  001260        9$:   MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
2402 022056  012637  001262              MOV     (KSP)+,TRAPPS
2403 022062  013737  177572  001264      MOV     SR0,WASSR0      ;SAVE CONTENTS OF SR0 FOR ERROR
2404 022070  013737  177576  001270      MOV     SR2,WASSR2      ;SAVE CONTENTS OF SR2 FOR ERROR
2405 022076  042737  160000  177572      BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
2406 022104  104007                      ERROR   +7              ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
2407                                                             ;FOR TIGHTER SCOPE LOOP
2408                                                             ;REPLACE ERROR CALL WITH
2409                                                             ;A 'NOP' = 000240
2410 022106  013746  001262              MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
2411 022 12  013746  001260              MOV     TRAPPC,-(KSP)
2412 022116  000002                      RTI                     ;RETURN FROM UNEXPECTED ABORT
2413
2414 022120  012737  016142  000250 10$:  MOV     #MGMERR,MMVEC   ;RESTORE NORMAL M.M. TRAP HANDLER
2415                                                             ;ADDRESS TO M.M. TRAP VECTOR
```

```
2427                                    .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 REPORTED BY SR0 & SR2.
                                        ;*      THE PAGE EXPANSION DIRECTION IN THIS TEST IS DOWNWARD, (THE ED BIT
                                        ;*      (BIT 3) OF PDR4=1).
                                        ;*
                                        ;*****************************************************************
  022126  000004                        TST5:   SCOPE
2428 022130  012700  117700           1$:     MOV     #117700,R0      ;LOAD VIRTUAL ADDR. TO SELECT PDR4 INTO R0
2429 022134  012704  077016                   MOV     #77016,R4       ;LOAD FIRST PDR VALUE IN R4 (PLF=176,ACF=6)
2430 022140  012737  022332  000250   2$:     MOV     #9$,@#MMVEC     ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
2431 022146  010437  172310                   MOV     R4,@#KIPDR4     ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
2432 022152  012737  022160  001110           MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
2433 022160  012706  001100           3$:     MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
2434 022164  011001                           MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA > PLF - NO ABORT)
2435 022166  162700  000100                   SUB     #100,R0         ;FORM NEXT VIRTUAL ADDRESS IN R0
2436 022172  012737  022200  001110           MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
2437 022200  012706  001100           4$:     MOV     #KERSTK,KSP     ;MAKE SURE STACK POINTER IS ALL SET UP
2438 022204  011001                           MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA=PLF - NO ABORT)
2439 022206  162700  000100                   SUB     #100,R0         ;FORM NEXT VIRTUAL ADDR. IN R0
2440 022212  020027  100000                   CMP     R0,#100000      ;HAVE ALL PLF'S BEEN TESTED YET?
2441 022216  001470                           BEQ     10$             ;BRANCH IF ALL VBA'S & PLF'S HAVE BEEN USED
2442 022220  012737  022234  001110           MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
2443 022226  012737  022242  000250           MOV     #6$,@#MMVEC     ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
2444 022234  011001                   5$:     MOV     (R0),R1         ;ACCESS VIRTUAL ADDR. (VBA < PLF - ABORT TO 6$)
2445 022236  104010                           ERROR   +10             ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
2446                                                                   ;FOR TIGHTER SCOPE LOOP
2447                                                                   ;REPLACE ERROR CALL WITH
2448                                                                   ;"BR 5$" = 000776
2449 022240  000424                           BR      8$              ;BRANCH AROUND ABORT CHECKS
2450 022242  012706  001100           6$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER FOLLOWING ABORT
2451 022246  013737  177572  001264           MOV     @#SR0,WASSR0    ;READ M.M. STATUS REG. 0
2452 022254  013737  177576  001270           MOV     @#SR2,WASSR2    ;READ M.M. STATUS REG. 2
2453 022262  012702  040011                   MOV     #40011,R2       ;PUT EXPECTED SR0 CONTENTS IN R2
2454 022266  020237  001264                   CMP     R2,WASSR0       ;DID SR0 REPORT PG. LENGTH ABORT, PG. 4, KERNEL?
2455 022272  001401                           BEQ     7$              ;BRANCH IF YES
2456 022274  104011                           ERROR   +11             ;SR0 DID NOT REPORT PG. LENGTH ABORT CORRECTLY
2457                                                                   ;FOR TIGHTER SCOPE LOOP
2458                                                                   ;REPLACE ERROR CALL WITH
2459                                                                   ;"BR 5$" = 000757
2460 022276  012703  022234           7$:     MOV     #5$,R3          ;PUT EXPECTED SR2 CONTENTS IN R3
2461 022302  020337  001270                   CMP     R3,WASSR2       ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
2462 022306  001401                           BEQ     8$              ;BRANCH IF YES
2463 022310  104012                           ERROR   +12             ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
2464                                                                   ;FOR TIGHTER SCOPE LOOP
2465                                                                   ;REPLACE ERROR CALL WITH
2466                                                                   ;"BR 5$" = 000751
2467 022312  042737  160000  177572   8$:     BIC     #160000,@#SR0   ;CLEAR ERROR BITS IN SR0
2468 022320  162704  000400                   SUB     #400,R4         ;FORM NEXT PLF VALUE FOR KIPDR4
2469 022324  062700  000100                   ADD     #100,R0         ;FORM FIRST VIRT. ADDR. FOR THAT PLF VALUE
2470 022330  000703                           BR      2$              ;BRANCH BACK AND ACCESS 3 VBA'S FOR
```

```
2471                                                                    ;THE PLF VALUE JUST FORMED
2472 022332  012637  001260          9$:    MOV     (KSP)+,TRAPPC       ;SAVE PC & PS OF TRAP
2473 022336  012637  001262                 MOV     (KSP)+,TRAPPS
2474 022342  013737  177572  001264         MOV     SR0,WASSR0          ;SAVE CONTENTS OF SR0 FOR ERROR
2475 022350  013737  177576  001270         MOV     SR2,WASSR2          ;SAVE CONTENTS OF SR2 FOR ERROR
2476 022356  042737  160000  177572         BIC     #160000,SR0         ;CLEAR ERROR BITS IN SR0
2477 022364  104007                        ERROR   +7                  ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
2478                                                                    ;FOR TIGHTER SCOPE LOOP
2479                                                                    ;REPLACE ERROR CALL WITH
2480                                                                    ;A "NOP" = 000240
2481 022366  013746  001262                 MOV     TRAPPS,-(KSP)       ;PUT PC & PS OF TRAP ON STACK
2482 022372  013746  001260                 MOV     TRAPPC,-(KSP)
2483 022376  000002                         RTI                         ;RETURN FROM UNEXPECTED ABORT
2484
2485 022400  012737  016142  000250  10$:    MOV     #MGMERR,MMVEC       ;RESTORE NORMAL M.M. TRAP HANDLER
2486                                                                    ;ADDRESS TO M.M. TRAP VECTOR
```

```
2500                                    .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$)             ;IN CASE A R/O ABORT DOES NOT OCCUR,
                                        ;*                                ;IN WHICH CASE SR2 WILL NOT CONTAIN CORRECT ADDR.
                                        ;*
                                        ;;*****************************************************************
     022406  000004                     TST6:   SCOPE
2501 022410  012737  000600  172346     1$:     MOV     #600,KIPAR3       ;BE SURE PAR3 IS MAPPED TO 12-16K
2502 022416  012737  000600  172350             MOV     #600,KIPAR4       ;BE SURE PAR4 IS MAPPED TO 12-16K
2503 022424  012737  077406  172306             MOV     #77406,KIPDR3     ;MAP PAGE 3  128 BLOCKS, R/W
2504 022432  012737  077402  172310             MOV     #77402,KIPDR4     ;MAP PAGE 4  128 BLOCKS, READ-ONLY
2505 022440  012700  060000                     MOV     #60000,R0         ;LOAD R0 WITH VIRTUAL ADDR. WHICH USES PDR3
2506 022444  012701  100000                     MOV     #100000,R1        ;LOAD R1 WITH VIRTUAL ADDR. WHICH USES PDR4
2507 022450  012737  022504  000250             MOV     #3$,MMVEC         ;SET M.M. TRAP VECTOR TO 3$
2508 022456  012737  022464  001110             MOV     #2$,$LPERR        ;SET LOOP ON ERROR POINTER TO 2$
2509 022464  012720  010727             2$:     MOV     #010727,(R0)+     ;LOAD "MOV PC,(PC)+" INSTRUCTION AT ADDR.
2510 022470  005020                             CLR     (R0)+             ;       REACHED THRU PDR/PAR 4.
2511 022472  012720  000137                     MOV     #000137,(R0)+     ;LOAD "JMP @#3$" INSTRUCTION AT VIRT. ADDR.
2512 022476  012710  022504                     MOV     #3$,(R0)          ;       IN CASE R/O VIOL. DOES NOT ABORT
2513 022502  010107                             MOV     R1,PC             ;TRANSFER PROGRAM EXECUTION TO "PAGE 4 INSTRUCTIONS"
2514 022504  012706  001100             3$:     MOV     #KERSTK,KSP       ;RESTORE STACK POINTER
2515 022510  013737  177576  001270             MOV     SR2,WASSR2        ;READ CONTENTS OF STATUS REG 2
2516 022516  020137  001270                     CMP     R1,WASSR2         ;WAS ADDR. OF "RELOCATED - R/O ABORT" LOCKED UP?
2517 022522  001401                             BEQ     4$                ;BRANCH IF YES
2518 022524  104013                             ERROR   +13               ;SR2 DID NOT LOCK UP VIRTUAL ADDR. OF R/O VIOL.
2519                                                                       ;FOR TIGHTER SCOPE LOOP
2520                                                                       ;REPLACE ERROR CALL WITH
2521                                                                       ;"BR 2$" = 000757
2522 022526  042737  160000  177572     4$:     BIC     #160000,SR0       ;CLEAR THE ERROR BITS IN SR0
2523 022534  162700  000004                     SUB     #4,R0             ;RESET R0 TO POINT TO NEXT VIRT. ADDR. TO LOAD
2524 022540  062701  000002                     ADD     #2,R1             ;FORM VIRTUAL ADDR. THAT SHOULD BE LOCKED UP NEXT
2525 022544  020127  111002                     CMP     R1,#111002        ;HAVE ALL VBA'S 100000-111000 BEEN TESTED?
2526 022550  103745                             BLO     2$                ;BRANCH IF NO
2527
2528 022552  012737  077406  172310     5$:     MOV     #77406,KIPDR4     ;RESTORE PDR4 TO R/W ACCESS
2529 022560  012737  016142  000250             MOV     #MGMERR,MMVEC     ;RESTORE ADDRESS OF NORMAL M.M.
2530                                                                       ;TRAP HANDLER TO M.M. VECTOR
```

```
2544                                    .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.
                                        ;*
                                        ;*****************************************************************
    022566  000004                      TST7:   SCOPE
2545 022570  012737  000600  172352     1$:     MOV     #600,KIPAR5     ;MAP KERNEL PAGE 5 TO 12-16K
2546 022576  012737  000406  172310             MOV     #406,KIPDR4     ;SETUP PDR4 FOR PAGE LENGTH ABORT
2547 022604  012737  077402  172312             MOV     #77402,KIPDR5   ;SETUP PDR5 FOR R/O ABORT
2548 022612  012737  022620  001110             MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2549 022620  013737  177576  001270     2$:     MOV     SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING
2550 022626  012701  022620                     MOV     #2$,R1          ;PUT EXPECTED VIRTUAL PC IN R1
2551 022632  020137  001270                     CMP     R1,WASSR2       ;DID SR2 CONTAIN VIRTUAL PC AT 2$?
2552 022636  001401                             BEQ     3$              ;BRANCH IF YES
2553 022640  104016                             ERROR   +16             ;SR2 NOT TRACKING CORRECTLY
2554                                                                     ;FOR TIGHTER SCOPE LOOP
2555                                                                     ;REPLACE ERROR CALL WITH
2556                                                                     ;"BR 2$" = 000767
2557 022642  012737  022650  001110     3$:     MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
2558 022650  013737  177576  001270     4$:     MOV     SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING
2559 022656  012701  022650                     MOV     #4$,R1          ;PUT EXPECTED VIRTUAL PC IN R1
2560 022662  020137  001270                     CMP     R1,WASSR2       ;DID SR2 CONTAIN VIRTUAL PC AT 4$
2561 022666  001401                             BEQ     5$              ;BRANCH IF YES
2562 022670  104016                             ERROR   +16             ;SR2 NOT TRACKING CORRECTLY
2563                                                                     ;FOR TIGHTER SCOPE LOOP
2564                                                                     ;REPLACE ERROR CALL WITH
2565                                                                     ;"BR 4$" = 000767
2566 022672  012737  022700  001110     5$:     MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
2567 022700  012737  022716  000250     6$:     MOV     #7$,MMVEC       ;PUT ADDRESS OF 7$ IN M.M. TRAP VECTOR
2568 022706  005037  001200                     CLR     $TMP1           ;CLEAR ERROR INDICATOR
2569 022712  005237  100500                     INC     @#100500        ;CAUSE PAGE LENGTH ABORT - TRAP TO 7$
2570 022716  012706  001100             7$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER AFTER ABORT
2571 022722  013737  177572  001176             MOV     SR0,$TMP0       ;SAVE SR0'S INFORMATION ON PG. LGTH. ABORT
2572 022730  013737  177576  001202             MOV     SR2,$TMP2       ;SAVE SR2'S INFORMATION ON PG. LGTH. ABORT
2573 022736  012737  022750  000250             MOV     #8$,MMVEC       ;PUT ADDRESS OF 8$ IN M.M. TRAP VECTOR
2574 022744  005237  120000                     INC     @#120000        ;CAUSE R/O ABORT - TRAP TO 8$
2575 022750  012706  001100             8$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER AFTER ABORT
2576 022754  013737  177572  001264             MOV     SR0,WASSR0      ;READ SR0 FOLLWOING SECOND KT ABORT
2577 022762  013737  177576  001270             MOV     SR2,WASSR2      ;READ SR2 FOLLOWING SECOND KT ABORT
2578 022770  023737  001176  001264             CMP     $TMP0,WASSR0    ;IS SR0 STILL HOLDING INFO ON FIRST ABORT?
2579 022776  001402                             BEQ     9$              ;BRANCH IF YES
2580 023000  005237  001200                     INC     $TMP1           ;SET ERROR INDICATOR
2581 023004  023737  001202  001270     9$:     CMP     $TMP2,WASSR2    ;DOES SR2 STILL HOLD PC OF FIRST ABORT?
2582 023012  001402                             BEQ     10$             ;BRANCH IF YES
2583 023014  005237  001200                     INC     $TMP1           ;SET ERROR INDICATOR
2584 023020  005737  001200             10$:    TST     $TMP1           ;WERE SR0 OR SR2 CHANGED BY A SECOND ABORT?
2585 023024  001401                             BEQ     11$             ;BRANCH IF NO
```

TEST # 7 - MORE CHECKS OF SR0 & SR2

```
2586 023026  104014                          ERROR   +14             ;ONE OF STATUS REGS. CHANGED BY SECOND ABORT
2587                                                                 ;FOR TIGHTER SCOPE LOOP
2588                                                                 ;REPLACE ERROR CALL WITH
2589                                                                 ;"BR 6$" = 000726
2590 023030  005037  001200          11$:   CLR     $TMP1           ;CLEAR ERROR INDICATOR
2591 023034  000005                         RESET                   ;EXECUTE A RESET, APPLYING AN "INIT"
2592 023036  013737  177572  001264         MOV     SR0,WASSR0      ;READ SR0
2593 023044  005737  001264                 TST     WASSR0          ;WAS SR0 CLEARED BY THE RESET?
2594 023050  001402                         BEQ     12$             ;BRANCH IF YES
2595 023052  005237  001200                 INC     $TMP1           ;SR0 NOT CLEARED BY A RESET
2596 023056  013737  177576  001270  12$:   MOV     SR2,WASSR2      ;READ SR2
2597 023064  022737  023056  001270         CMP     #12$,WASSR2     ;WAS SR2 UNLOCKED BY A RESET?
2598 023072  001402                         BEQ     13$             ;BRANCH IF YES
2599 023074  005237  001200                 INC     $TMP1           ;SR2 NOT UNLOCKED BY A RESET
2600 023100  005737  001200          13$:   TST     $TMP1           ;WERE SR0 & SR2 BOTH "RESET" BY A RESET?
2601 023104  001401                         BEQ     14$             ;BRANCH IF YES
2602 023106  104015                         ERROR   +15             ;SR0 OR SR2 NOT "RESET" BY A RESET
2603                                                                 ;FOR TIGHTER SCOPE LOOP
2604                                                                 ;REPLACE ERROR CALL WITH
2605                                                                 ;"BR 6$" = 000676
2606 023110  005237  177572          14$:   INC     SR0             ;TURN MEMORY MANAGEMENT BACK ON
2607 023114  013737  177576  001270  15$:   MOV     SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING AGAIN
2608 023122  012701  023114                 MOV     #15$,R1         ;PUT EXPECTED VIRTUAL PC IN R1
2609 023126  020137  001270                 CMP     R1,WASSR2       ;DID SR2 CONTAIN VIRTUAL PC AT 15$
2610 023132  001401                         BEQ     16$             ;BRANCH IF YES
2611 023134  104016                         ERROR   +16             ;SR2 NOT TRACKING CORRECTLY
2612                                                                 ;FOR TIGHTER SCOPE LOOP
2613                                                                 ;REPLACE ERROR CALL WITH
2614                                                                 ;"BR 6$" = 000663
2615 023136  012737  077406  172310  16$:   MOV     #77406,KIPDR4   ;RESET PDR4 TO 128 BLKS, R/W
2616 023144  012737  077406  172312         MOV     #77406,KIPDR5   ;RESET PDR5 TO 128 BLKS, R/W
2617 023152  012737  016142  000250         MOV     #MGMERR,MMVEC   ;RESTORE ADDRESS OF NORMAL MEMORY
2618                                                                 ;MANAGEMENT TRAP ROUTINE TO M.M. VECTOR
```

```
2632                                    .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 ADDRESS ERROR IS
                                        ;*      USED TO CAUSE A TRAP TO "4".
                                        ;*
                                        ;;*****************************************************************
     023160  000004                     TST10:  SCOPE
2633 023162  004737  002356             1$:     JSR     PC,TOFF         ;TURN OFF T-BIT TRAPPING FOR THIS TEST
2634 023166  012737  023174  001110             MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2635 023174  005037  177776             2$:     CLR     PSW             ;GO TO KERNEL MODE
2636 023200  012706  001100                     MOV     #KERSTK,KSP     ;SETUP KERNEL STACK PTR.
2637                                    ;*
2638                                    ;*      TEST USER MODE ABORT
2639                                    ;*
2640 023204  012737  000600  177640             MOV     #600,UIPAR0     ;MAP USER PAGE 0 TO 12K
2641 023212  012737  023274  000004             MOV     #4$,@#4         ;LOAD KERNEL VECTOR 4 (LOC.4) WITH 4$
2642 023220  012737  000340  000006             MOV     #340,@#6        ;LOAD VECTOR+2 WITH NEW PSW
2643 023226  012737  140000  177776             MOV     #140000,PSW     ;GO TO USER MODE
2644 023234  012706  000600                     MOV     #USESTK,USP     ;SETUP USER STACK PTR.
2645 023240  012737  023260  000004             MOV     #3$,@#4         ;LOAD USER VECTOR 4 (LOC. 60004) WITH 3$
2646 023246  012737  000340  000006             MOV     #340,@#6        ;LOAD VECTOR+2 WITH NEW PSW
2647 023254  005737  023261                     TST     3$+1            ;CAUSE ODD ADDR. ERROR TRAP TO "4"
2648                                                                     ;SHOULD PICK UP NEW PC=4$ FROM KERNEL
2649                                                                     ;LOC. 4, NOT PC=3$ FROM USER LOC. 4 (=60004)
2650 023260  013701  177776             3$:     MOV     PSW,R1          ;SAVE PSW FOR ERROR
2651 023264  010602                             MOV     SP,R2           ;SAVE VALUE OF STACK POINTER FOR ERROR
2652 023266  005037  177776                     CLR     PSW             ;BE SURE BACK IN KERNEL MODE
2653 023272  104017                             ERROR   +17             ;DID NOT TRAP THRU KERNEL SPACE
2654                                                                     ;FOR TIGHTER SCOPE LOOP
2655                                                                     ;REPLACE ERROR CALL WITH
2656                                                                     ;"BR 2$" = 000740
2657 023274  005037  177776             4$:     CLR     PSW             ;BE SURE BACK IN KERNEL MODE
2658 023300  012706  001100                     MOV     #KERSTK,KSP     ;RESTORE KERNEL S.P. IN CASE IT CHANGED
2659 023304  005037  177640                     CLR     UIPAR0          ;REMAP USER PAGE 0 TO 0-4K
2660 023310  012737  140000  177776             MOV     #140000,PSW     ;GO TO USER MODE
2661 023316  012706  000600                     MOV     #USESTK,USP     ;RESTORE USER STACK POINTER
2662 023322  005037  177776                     CLR     PSW             ;GO BACK TO KERNAL MODE
2663                                    ;*
2664                                    ;*      NOW TEST THE SUPERVISOR MODE ABORT
2665                                    ;*
2666 023326  012737  000600  172240             MOV     #600,SIPAR0     ;MAP SUPERVISOR PAGE 0 TO 12K
2667 023334  012737  023416  000004             MOV     #6$,@#4         ;LOAD KERNAL VECTOR 4 WITH 6$
2668 023342  012737  000340  000006             MOV     #340,@#6        ;LOAD VECTOR+2 WITH NEW PSW
2669 023350  012737  040000  177776             MOV     #40000,PSW      ;GO TO SUPERVISOR MODE
2670 023356  012706  000700                     MOV     #SUPSTK,SSP     ;SETUP SUPERVISOR STACK PTR.
2671 023362  012737  023402  000004             MOV     #5$,@#4         ;LOAD SUPERVISOR VECTOR 4 (LOC. 60004) WITH 5$
2672 023370  012737  000340  000006             MOV     #340,@#6        ;LOAD VECTOR+2 WITH NEW PSW
2673 023376  005737  023403                     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)
023402  013701  177776          5$:   MOV     PSW,R1            ;SAVE PSW FOR ERROR
023406  010602                        MOV     SP,R2             ;SAVE VALUE OF STACK POINTER FOR ERROR
023410  005037  177776                CLR     PSW               ;BE SURE BACK IN KERNEL MODE
023414  104017                        ERROR   +17               ;DID NOT TRAP THRU KERNEL SPACE
                                                                ;FOR TIGHTER SCOPE LOOP
                                                                ;REPLACE ERROR CALL WITH
                                                                ;'BR 2$'' = 000740
023416  005037  177776          6$:   CLR     PSW               ;BE SURE BACK IN KERNEL MODE
023422  012706  001100                MOV     #KERSTK,KSP       ;RESTORE KERNEL S.P. IN CASE IT CHANGED
023426  005037  172240                CLR     SIPAR0            ;REMAP SUPERVISOR PAGE 0 TO 0-4K
023432  012737  040000  177776        MOV     #40000,PSW        ;GO TO SUPERVISOR MODE
023440  012706  000700                MOV     #SUPSTK,SSP       ;RESTORE SUPERVISOR STACK POINTER
023444  005037  177776                CLR     PSW               ;GO BACK TO KERNEL MODE
023450  012737  016070  000004        MOV     #TIMERR,@#4       ;RESTORE ADDR. OF NORMAL CPU TRAP HANDLER TO 4
023456  004737  002412                JSR     PC,TON            ;TURN T-BIT TRAPPING BACK ON
```

```
2697                                    .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.
                                        ;*
                                        ;;*****************************************************************
     023462  000004                     TST11:  SCOPE
2698
2699 023464  012737  023476  001110     1$:     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2700 023472  012702  170000             1$:     MOV     #170000,R2      ;LOAD "PRESENT & EXPECTED" PSW VALUE INTO R2
2701 023476  010237  177776             2$:     MOV     R2,PSW          ;GO TO PRESENT MODE-PRIORITY 0
2702 023502  012746  000340                     MOV     #340,-(SP)      ;PUT A NEW PSW (PRIORITY=7) ON STACK
2703 023506  012746  023514                     MOV     #3$,-(SP)       ;PUT NEW PC ON THE STACK
2704 023512  000002                             RTI                     ;DO AN RTI FROM PRESENT MODE
2705 023514  013701  177776             3$:     MOV     PSW,R1          ;READ NEW PSW INTO R1
2706 023520  042701  007437                     BIC     #7437,R1        ;MASK OFF COND. CODE, T-BIT, AND UNUSED BITS
2707 023524  005037  177776                     CLR     PSW             ;GO BACK TO KERNEL MODE
2708 023530  020201                             CMP     R2,R1           ;DID PSW STAY IN PRESENT MODE, PRIORITY=0?
2709 023532  001401                             BEQ     4$              ;BRANCH IF YES
2710 023534  104032                             ERROR   +32             ;PSW CHANGED BY AN RTI FROM USER
2711                                                                     ;FOR A TIGHTER SCOPE LOOP
2712                                                                     ;REPLACE ERROR CALL WITH
2713                                                                     ;"BR=2$" = 000760
2714 023536  022702  050000             4$:     CMP     #50000,R2       ;IF SUPERVISOR MODE HAS BEEN CHECKED,
2715 023542  001403                             BEQ     TST12           ;;GO TO NEXT TEST
2716 023544  012702  050000                     MOV     #50000,R2       ;ELSE,SET SUPERVISOR MODE,
2717 023550  000752                             BR      2$              ;AND BRANCH BACK TO TEST IT.
```

```
2729                                    .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
                                        ;*
                                        ;;*****************************************************************
  023552  000004                       TST12:  SCOPE
2730 023554  012737  000600  172350    1$:     MOV     #600,KIPAR4     ;MAP KERNEL PAGE 4 TO 12-16K
2731 023562  012705  077402             MOV     #77402,R5       ;LOAD PDR4 DATA INTO R5
2732 023566  010537  172310             MOV     R5,KIPDR4       ;MAP PAGE 4 READ-ONLY
2733 023572  012737  023622  000004     MOV     #4$,@#4         ;SET CPU TRAP VECTOR TO ADDRESS OF 4$
2734 023600  012737  023620  000250     MOV     #3$,@#250       ;SET M.M. TRAP VECTOR TO ADDRESS OF 3$
2735 023606  012737  023614  001110     MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2736 023614  005237  060001             2$:     INC     60001           ;CAUSE ODD ADDR. ERROR & POTENTIAL R/O ABORT
2737 023620  104020                     3$:     ERROR   +20             ;TRAPPED THRU M.M. VECTOR BUT SHOULDN'T HAVE
2738                                                                     ;FOR TIGHTER SCOPE LOOP
2739                                                                     ;REPLACE ERROR CALL WITH
2740                                                                     ;'BR 2$'' = 000776
2741 023622  012706  001100             4$:     MOV     #KERSTK,KSP     ;RESTORE STACK POINTER AFTER TRAPPING
2742 023626  005037  001200                     CLR     $TMP1           ;CLEAR ERROR INDICATOR
2743 023632  013737  177572  001264             MOV     SR0,WASSR0      ;READ STATUS REG. 0
2744 023640  013737  177576  001270     5$:     MOV     SR2,WASSR2      ;READ STATUS REG. 2
2745 023646  012700  000017                     MOV     #17,R0          ;LOAD EXPECTED SR0 CONTENTS INTO R0
2746 023652  020037  001264                     CMP     R0,WASSR0       ;SR0  ERROR BITS LEFT CLEAR BY TRAPPING?
2747 023656  001402                             BEQ     6$              ;BRANCH IF YES
2748 023660  005237  001200                     INC     $TMP1           ;SR0 ERROR BITS SET WHEN ODD ADDR. SERVICED
2749 023664  012701  023640             6$:     MOV     #5$,R1          ;LOAD EXPECTED SR2 CONTENTS INTO R1
2750 023670  020137  001270                     CMP     R1,WASSR2       ;WAS SR2 LEFT UNLOCKED BY TRAPPING?
2751 023674  001402                             BEQ     7$              ;BRANCH IF YES
2752 023676  005237  001200                     INC     $TMP1           ;SR2 LOCKED UP BY ODD ADDR. ERROR
2753 023702  005737  001200             7$:     TST     $TMP1           ;WHERE SR0 OR SR2 EFFECTED?
2754 023706  001404                             BEQ     8$              ;BRANCH IF NO
2755 023710  104021                             ERROR   +21             ;SR0 OR SR2 CHANGED BY ODD ADDR. ERROR
2756                                                                     ;FOR TIGHTER SCOPE LOOP
2757                                                                     ;REPLACE ERROR CALL WITH
2758                                                                     ;'BR 2$'' = 000741
2759 023712  042737  160000  177572             BIC     #160000,SR0     ;CLEAR ERROR BITS THAT MAY BE SET IN SR0
2760 023720  012737  016070  000004     8$:     MOV     #TIMERR,@#4     ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
2761 023726  012737  016142  000250             MOV     #MGMERR,@#250   ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
2762 023734  012737  077406  172310             MOV     #77406,KIPDR4   ;REMAP PAGE 4 TO READ/WRITE
```

```
2777                                    .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.
                                        ;*
                                        ;*********************************************************
    023742  000004                      TST13:  SCOPE
2778 023744  004737  002356             1$:     JSR     PC,TOFF         ;TURN T-BIT TRAPPING OFF FOR THIS TEST
2779 023750  012737  000600  177646             MOV     #600,UIPAR3     ;MAP USER PAGE 3 TO 12-16K
2780 023756  012737  000600  177650             MOV     #600,UIPAR4     ;MAP USER PAGE 4 TO 12-16K
2781 023764  012737  077402  177606             MOV     #77402,UIPDR3   ;MAP USER PAGE 3 READ-ONLY
2782 023772  012737  077406  177610             MOV     #77406,UIPDR4   ;MAP USER PAGE 4 READ/WRITE
2783 024000  012737  024054  000004             MOV     #4$,@#4         ;LOAD ADDRESS OF 4$ IN CPU (ODD ADDR.) VECTOR
2784 024006  012737  140017  000006             MOV     #140017,@#6     ;LOAD PSW THAT SHOULD BE PUT ON STACK IN VECTOR+2
2785 024014  012737  024054  000250             MOV     #4$,@#250       ;LOAD ADDRESS OF 4$ IN M.M. TRAP VECTOR
2786 024022  012737  000340  000252             MOV     #340,@#252      ;LOAD A KERNEL PSW IN MMVEC+2
2787 024030  012737  024036  001110             MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2788 024036  012737  140000  177776  2$:        MOV     #140000,PSW     ;GO TO USER MODE
2789 024044  012706  100002                     MOV     #100002,USP     ;SET USER STACK PTR. SO SECOND PUSH IS IN PG. 3
2790 024050  005237  100005             3$:     INC     100005          ;CAUSE ODD ADDRESS ERROR THAT WILL CAUSE
2791                                                                    ;R/O ERROR WHEN TRY TO SAVE OLD PC
2792 024054  016601  000002             4$:     MOV     2(KSP),R1       ;PUT PSW SAVED ON KERNEL STACK INTO R1
2793 024060  011603                             MOV     (KSP),R3        ;PUT PC SAVED ON KERNEL STACK INTO R3
2794 024062  013737  177572  001264             MOV     SR0,WASSR0      ;READ THE CONTENTS OF M.M. STATUS REG. 0
2795 024070  013737  177576  001270             MOV     SR2,WASSR2      ;READ THE CONTENTS OF M.M. STATUS REG. 2
2796 024076  042737  160000  177572             BIC     #160000,SR0     ;CLEAR THE ERROR BITS IN SR0
2797 024104  005037  177776                     CLR     PSW             ;BE SURE IN KERNEL MODE
2798 024110  012706  001100                     MOV     #KERSTK,KSP     ;RESTORE KERNEL STACK POINTER
2799 024114  012737  140000  177776             MOV     #140000,PSW     ;GO TO USER MODE
2800 024122  012706  000600                     MOV     #USESTK,USP     ;RESTORE USER STACK POINTER
2801 024126  005037  177776                     CLR     PSW             ;GO BACK TO KERNEL MODE
2802 024132  005037  001176                     CLR     $TMP0           ;CLEAR ERROR INDICATOR
2803 024136  020127  170017                     CMP     R1,#170017      ;WAS THE PSW SAVED THE ONE PICKED UP BY THE
2804                                                                    ;ODD ADDR. TRAP FROM ERRVEC+2?
2805                                                                    ;VALUE 170017 = PSW FROM LOC. 6 WITH
2806                                                                    ;PREVIOUS MODE BITS = USER
2807 024142  001402                             BEQ     5$              ;BRANCH IF YES
2808 024144  005237  001176                     INC     $TMP0           ;WRONG PSW SAVED DURING "DOUBLE ERROR" SEQUENCE
2809 024150  020327  024054             5$:     CMP     R3,#3$+4        ;WAS THE PC AT THE TIME OF THE ODD ADDR. ERROR
2810                                                                    ;SAVED ON THE STACK?
2811 024154  001402                             BEQ     6$              ;BRANCH IF YES
2812 024156  005237  001176                     INC     $TMP0           ;WRONG PC SAVED DURING TRAP SEQUENCE
2813 024162  023727  001264  020147  6$:        CMP     WASSR0,#20147   ;DID SR0 REPORT - USER, PAGE 3, R/O ABORT?
2814 024170  001402                             BEQ     7$              ;BRANCH IF YES
2815 024172  005237  001176                     INC     $TMP0           ;SR0 DID NOT REPORT R/O ABORT
2816 024176  023727  001270  024050  7$:        CMP     WASSR2,#3$      ;DID SR2 LOCK UP VIRTUAL ADDR. OF LAST
2817                                                                    ;INSTRUCTION SUCCESSFULLY FETCHED?
```

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

```
2818 024204  001402                        BEQ     8$              ;BRANCH IF YES
2819 024206  005237  001176                INC     $TMP0           ;SR2 DID NOT LOCK UP ADDR. OF ODD ADDR. INST.
2820 024212  005737  001176          8$:   TST     $TMP0           ;ANY 'ERRORS' DURING TRAP SEQUENCE?
2821 024216  001401                        BEQ     9$              ;BRANCH IF NO
2822 024220  104022                        ERROR   +22             ;THE WRONG PC OR PSW WERE SAVED
2823                                                               ;OR SR0 OR SR2 DID NOT REPORT R/O
2824                                                               ;ERROR DURING ODD ADDR. - KT TRAP
2825                                                               ;SEQUENCE
2826                                                               ;FOR TIGHTER SCOPE LOOP
2827                                                               ;REPLACE ERROR CALL WITH
2828                                                               ;'BR 2$' = 000710
2829 024222  012737  016070  000004  9$:   MOV     #TIMERR,@#4     ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
2830 024230  012737  000340  000006        MOV     #340,@#6        ;RELOAD ERRVEC+2 WITH KERNEL PSW
2831 024236  012737  016142  000250        MOV     #MGMERR,@#250   ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
2832 024244  012737  077406  177606        MOV     #77406,UIPDR3   ;REMAP USER PAGE 3 READ/WRITE
2833 024252  004737  002412                JSR     PC,TON          ;TURN T-BIT TRAPPING BACK ON
```

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

```
                                        .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.
                                        ;*
                                        ;;*****************************************************************
     024256  000004                     TST14:  SCOPE
2853                                            ;INITIALIZE KERNAL I/D SPACE PAR'S AND PDR'S
2854 024260  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2855 024264  012700  077400                     MOV     #77400,R0       ;MAKE KERNAL D-SPACE PGS. 1,2&3 NON-RESIDENT
2856 024270  010037  172310                     MOV     R0,KIPDR4       ;AND KERNAL I-SPACE PDR4 NON-RESIDENT
2857 024274  010037  172322                     MOV     R0,KDPDR1
2858 024300  010037  172324                     MOV     R0,KDPDR2
2859 024304  010037  172326                     MOV     R0,KDPDR3
2860 024310  012737  000600  172370            MOV     #600,KDPAR4     ;MAP KDPAR4 TO 12->16K
2861 024316  012737  002232  000250            MOV     #NODSPAC,MMVEC  ;SET M.M. VECTOR TO D-SPACE SERVICE ROUTINE
2862 024324  012737  024344  001110            MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
2863 024332  005237  177572                     INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2864 024336  012737  000024  172516            MOV     #24,MMR3        ;ENABLE 22-BIT,KERNAL D-SPACE MAPPING
2865                                    ;*      TEST THAT INSTRUCTION FETCHES FORCE I-SPACE
2866                                    ;*
2867 024344  000400                     10$:    BR      4$              ;BRANCH
2868 024346  000244                     4$:     CLZ                     ;CLEAR ZERO BIT IN PSW
2869 024350  001776                             BEQ     4$              ;NO BRANCH
2870 024352  000264                     5$:     SEZ                     ;SET ZERO BIT IN PSW
2871 024354  001376                             BNE     5$              ;NO BRANCH
2872 024356  000270                     6$:     SEN                     ;SET NEGATIVE BIT IN PSW
2873 024360  100376                             BPL     6$              ;NO BRANCH
2874                                    ;*      TRY SOME SOP INSTRUCTIONS WITH SRCM=DSTM=0
2875                                    ;*      THESE SHOULD NEVER INVOKE D-SPACE
2876                                    ;*
2877 024362  005700                             TST     R0
2878 024364  005200                             INC     R0
2879 024366  005300                             DEC     R0
2880 024370  006000                             ROR     R0
2881 024372  006100                             ROL     R0
2882 024374  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2883                                    ;*      TEST SOB NON-MOD WITH DSTM=2,3,6,7; DSTF=7
2884                                    ;*
2885 024400  012737  100002  060000            MOV     #100002,@#60000 ;SET UP TEST LOCATION
2886 024406  012737  024420  001110            MOV     #11$,$LPERR     ;SET LOOP ON ERROR POINTER TO 11$
2887 024414  005237  177572                     INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2888 024420  005727  060000              11$:   TST     #60000          ;DSTM=2 SOP NON-MOD
2889 024424  005737  100000                     TST     @#100000        ;DSTM=3 SOP NON-MOD
2890 024430  005737  100000                     TST     100000          ;DSTM=6 SOP NON-MOD
2891 024434  005777  053340                     TST     @100000         ;DSTM=7 SOP NON-MOD
```

```
2892 024440  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2893                                 ;*         TEST SOB MOD WITH DSTM=2,3,6,7; DSTF=7
2894                                 ;*
2895 024444  012737  024464  001110            MOV     #12$,$LPERR     ;SET LOOP ON ERROR POINTER TO 12$
2896 024452  012737  100000  060002            MOV     #100000,@#60002 ;SET UP TEST VALUES
2897 024460  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2898 024464  005027  060000          12$:      CLR     #60000          ;DSTM=2 SOP MOD
2899 024470  005037  100000                    CLR     @#100000        ;DSTM=3 SOP MOD
2900 024474  005037  100000                    CLR     100000          ;DSTM=6 SOP MOD
2901 024500  005077  053276                    CLR     @100002         ;DSTM=7 SOP MOD
2902 024504  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2903                                 ;*
2904                                 ;*         THE NEXT THREE TESTS ARE CONCERNED WITH TESTING
2905                                 ;*         DOP AND NOT(SRCM=DSTM=0)
2906                                 ;*
2907                                 ;*         TEST DOP DEST NON-MOD SRCM=DSTM=2,3,6,7; SRCF=DSTF=7
2908                                 ;*
2909 024510  012737  024530  001110            MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
2910 024516  012737  100000  060002            MOV     #100000,@#60002 ;SET UP TEST VALUE
2911 024524  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2912 024530  022727  060000  060000  13$:      CMP     #60000,#60000   ;SRCM=2 DSTM=2 DOP NON-MOD
2913 024536  023737  100000  100000            CMP     @#100000,@#100000 ;SRCM=3 DSTM=3 DOP NON-MOD
2914 024544  023737  100000  100000            CMP     100000,100000   ;SRCM=6 DSTM=6 DOP NON-MOD
2915 024552  027777  053224  053222            CMP     @100002,@100002 ;SRCM=7 DSTM=7 DOP NON-MOD
2916 024560  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2917                                 ;*         TEST MOV DEST AND NOT(SRCM=DSTM=0)
2918                                 ;*         SRCM=DSTM=2,3,6,7; SRCF=DSTF=7
2919                                 ;*
2920 024564  012737  024604  001110            MOV     #14$,$LPERR     ;SET LOOP ON ERROR POINTER TO 14$
2921 024572  012737  100000  060002            MOV     #100000,@#60002 ;SET UP TEST VALUE
2922 024600  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2923 024604  012727  060002  060002  14$:      MOV     #60002,#60002   ;SRCM=2 DSTM=2 MOV
2924 024612  012737  060000  100000            MOV     #60000,@#100000 ;SRCM=2 DSTM=3 MOV
2925 024620  012737  060000  100000            MOV     #60000,100000   ;SRCM=2 DSTM=6 MOV
2926 024626  012777  060000  053146            MOV     #60000,@100002  ;SRCM=2 DSTM=7 MOV
2927 024634  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2928                                 ;*         TEST DOP DEST MOD AND NOT SUB
2929                                 ;*         SRCM=DSTM=2,3,6,7; SRCF=DSTF=7
2930                                 ;*
2931 024640  012737  024652  001110            MOV     #15$,$LPERR     ;SET LOOP ON ERROR POINTER TO 15$
2932 024646  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2933 024652  052727  060000  060000  15$:      BIS     #60000,#60000   ;SRCM=2 DSTM=2 DOP MOD
2934 024660  052737  000000  100000            BIS     #00000,@#100000 ;SRCM=2 DSTM=3 DOP MOD
2935 024666  052737  000000  100000            BIS     #00000,100000   ;SRCM=2 DSTM=6 DOP MOD
2936 024674  052777  000000  053100            BIS     #00000,@100002  ;SRCM=2 DSTM=7 DOP MOD
2937 024702  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2938                                 ;*         TEST SWAB WITH DSTM=2,3,6,7; DSTF=7
2939                                 ;*
2940 024706  012737  024726  001110            MOV     #16$,$LPERR     ;SET LOOP ON ERROR POINTER TO 16$
2941 024714  012737  100002  060000            MOV     #100002,@#60000 ;SET UP TEST VALUES
2942 024722  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2943 024726  000327  060000          16$:      SWAB    #60000          ;DSTM=2 SWAB
2944 024732  000337  100002                    SWAB    @#100002        ;DSTM=3 SWAB
2945 024736  000337  100002                    SWAB    100002          ;DSTM=6 SWAB
2946 024742  000377  053032                    SWAB    @100000         ;DSTM=7 SWAB
2947 024746  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2948                                 ;*         TEST ROT/SHFT WITH DSTM=2,3,6,7; DSTF=7
```

```
2949                                        ;*
2950 024752  012737  024772  001110         MOV     #17$,$LPERR     ;SET LOOP ON ERROR POINTER TO 17$
2951 024760  012737  100000  060002         MOV     #100000,@#60002 ;SET UP TEST VALUES
2952 024766  005237  177572                 INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2953 024772  006127  060000         17$:    ROL     #60000          ;DSTM=2 ROT/SHFT
2954 024776  006137  100000                 ROL     @#100000        ;DSTM=3 ROT/SHFT
2955 025002  006137  100000                 ROL     100000          ;DSTM=6 ROT/SHFT
2956 025006  006177  052770                 ROL     @100002         ;DSTM=7 ROT/SHFT
2957 025012  005037  177572                 CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2958                                        ;*      TEST ASH/ASHC WITH DSTM=2,3,6,7; DSTF=7
2959                                        ;*
2960 025016  012737  025044  001110         MOV     #18$,$LPERR     ;SET LOOP ON ERROR POINTER TO 18$
2961 025024  012737  000001  060000         MOV     #1,@#60000      ;SET UP TEST VALUES
2962 025032  012737  100000  060002         MOV     #100000,@#60002
2963 025040  005237  177572                 INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2964 025044  072027  000001         18$:    ASH     #1,R0           ;DSTM=2 ASH/ASHC
2965 025050  072037  100000                 ASH     @#100000,R0     ;DSTM=3 ASH/ASHC
2966 025054  072037  100000                 ASH     100000,R0       ;DSTM=6 ASH/ASHC
2967 025060  072077  052716                 ASH     @100002,R0      ;DSTM=7 ASH/ASHC
2968 025064  005037  177572                 CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2969                                        ;*      TEST MUL/DIV WITH DSTM=2,3,6,7; DSTF=7
2970                                        ;*
2971 025070  012737  025102  001110         MOV     #19$,$LPERR     ;SET LOOP ON ERROR POINTER TO 19$
2972 025076  005237  177572                 INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2973 025102  070027  000002         19$:    MUL     #2,R0           ;DSTM=2 MUL/DIV
2974 025106  070037  100000                 MUL     @#100000,R0     ;DSTM=3 MUL/DIV
2975 025112  070037  100000                 MUL     100000,R0       ;DSTM=6 MUL/DIV
2976 025116  070077  052660                 MUL     @100002,R0      ;DSTM=7 MUL/DIV
2977 025122  005037  177572                 CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2978                                        ;*      TEST JMP WITH DSTM=3,6,7; DSTF=7
2979                                        ;*
2980 025126  012737  025146  001110         MOV     #20$,$LPERR     ;SET LOOP ON ERROR POINTER TO 20$
2981 025134  012737  025162  060000         MOV     #23$,@#60000    ;SET UP TEST VALUES
2982 025142  005237  177572                 INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2983 025146  000137  025152         20$:    JMP     @#21$           ;DSTM=3 JMP
2984 025152  000137  025156         21$:    JMP     22$             ;DSTM=6 JMP
2985 025156  000177  052616         22$:    JMP     @100000         ;DSTM=7 JMP
2986 025162  005037  177572         23$:    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
2987                                        ;*      TEST SUB WITH DSTM=2,3,6,7; DSTF=7
2988                                        ;*
2989 025166  012737  025210  001110         MOV     #28$,$LPERR     ;SET LOOP ON ERROR POINTER TO 28$
2990 025174  005000                         CLR     R0              ;SET UP TEST VALUES
2991 025176  012737  100000  060002         MOV     #100000,@#60002
2992 025204  005237  177572                 INC     MMR0            ;TURN ON MEMORY MANAGEMENT
2993 025210  160027  060002         28$:    SUB     R0,#60002       ;DSTM=2 SUB
2994 025214  160037  100000                 SUB     R0,@#100000     ;DSTM=3 SUB
2995 025220  160037  100000                 SUB     R0,100000       ;DSTM=6 SUB
2996 025224  160077  052552                 SUB     R0,@100002      ;DSTM=7 SUB
2997 025230  005037  177572                 CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
```

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

```
3008                                    .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.
                                        ;*
                                        ;*********************************************************************
     025234  000004                     TST15:  SCOPE
3009 025236  012700  077406                     MOV     #77406,R0
3010 025242  010037  172310                     MOV     R0,KIPDR4       ;MAKE KIPDR4 R/W,4K,200 BLOCKS
3011 025246  010037  172322                     MOV     R0,KDPDR1       ;MAKE KDPDR1 R/W,4K,200 BLOCKS
3012 025252  010037  172324                     MOV     R0,KDPDR2       ;MAKE KDPDR2 R/W,4K,200 BLOCKS
3013 025256  010037  172326                     MOV     R0,KDPDR3       ;MAKE KDPDR3 R/W,4K,200 BLOCKS
3014 025262  105037  172306                     CLRB    KIPDR3          ;MAKE KIPDR3 NON-RESIDENT
3015                                    ;*      TEST SOP NON-MOD; DSTM=1,2,3,5,7
3016                                    ;*
3017 025266  012700  060000            20$:     MOV     #60000,R0       ;SET UP CONSTANTS FOR TEST
3018 025272  010037  060000                     MOV     R0,@#60000
3019 025276  012737  060002  060002            MOV     #60002,@#60002
3020 025304  012737  060004  060004            MOV     #60004,@#60004
3021 025312  012737  060006  060006            MOV     #60006,@#60006
3022 025320  012737  025362  001110            MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
3023 025326  012737  060000  060000            MOV     #60000,@#60000
3024 025334  012737  060002  060002            MOV     #60002,@#60002
3025 025342  012737  060004  060004            MOV     #60004,@#60004
3026 025350  012737  060006  060006            MOV     #60006,@#60006
3027 025356  005237  177572                     INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3028 025362  005710                    1$:     TST     (R0)            ;DSTM=1 SOP NON-MOD
3029 025364  005720                            TST     (R0)+           ;DSTM=2 SOP NON-MOD
3030 025366  005730                            TST     @(R0)+          ;DSTM=3 SOP NON-MOD
3031 025370  005750                            TST     @-(R0)          ;DSTM=5 SOP NON-MOD
3032 025372  005770  000000                    TST     @0(R0)          ;DSTM=7 SOP NON-MOD
3033 025376  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3034                                    ;*      TEST SOP MOD; DSTM=1,2,3,5,7
3035                                    ;*
3036 025402  012737  025414  001110            MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3037 025410  005237  177572                     INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3038 025414  005010                    2$:     CLR     (R0)            ;DSTM=1 SOP MOD
3039 025416  005020                            CLR     (R0)+           ;DSTM=2 SOP MOD
3040 025420  005030                            CLR     @(R0)+          ;DSTM=3 SOP MOD
3041 025422  005050                            CLR     @-(R0)          ;DSTM=5 SOP MOD
3042 025424  005070  000000                    CLR     @0(R0)          ;DSTM=7 SOP MOD
3043 025430  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3044                                    ;*      TEST DOP DEST NON-MOD WITH SRCM=1,2,3,5,7 AND DSTM=1,2,3,5,7
3045                                    ;*      ALL SOURCE MODES TO BE TESTED ARE TESTED HERE
3046                                    ;*
3047 025434  012737  025476  001110            MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
3048 025442  012702  000032                     MOV     #32,R2          ;SET UP ADDRESSES 60000-60064 FOR TEST
3049 025446  012700  060000                     MOV     #60000,R0
3050 025452  012701  060000                     MOV     #60000,R1
3051 025456  010021                    21$:     MOV     R0,(R1)+
3052 025460  062700  000002                     ADD     #2,R0
```

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

```
3053 025464  077204                        SOB     R2,21$
3054 025466  012700  060000                MOV     #60000,R0
3055 025472  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3056 025476  021010                3$:     CMP     (R0),(R0)       ;SRCM=1 DSTM=1 DOP DEST NON-MOD
3057 025500  021020                        CMP     (R0),(R0)+      ;SRCM=1 DSTM=2 DOP DEST NON-MOD
3058 025502  021030                        CMP     (R0),@(R0)+     ;SRCM=1 DSTM=3 DOP DEST NON-MOD
3059 025504  021050                        CMP     (R0),@-(R0)     ;SRCM=1 DSTM=5 DOP DEST NON-MOD
3060 025506  021070  000000                CMP     (R0),@0(R0)     ;SRCM=1 DSTM=7 DOP DEST NON-MOD
3061 025512  022010                        CMP     (R0)+,(R0)      ;SRCM=2 DSTM=1 DOP DEST NON-MOD
3062 025514  022020                        CMP     (R0)+,(R0)+     ;SRCM=2 DSTM=2 DOP DEST NON-MOD
3063 025516  022030                        CMP     (R0)+,@(R0)+    ;SRCM=2 DSTM=3 DOP DEST NON-MOD
3064 025520  022050                        CMP     (R0)+,@-(R0)    ;SRCM=2 DSTM=5 DOP DEST NON-MOD
3065 025522  022070  000000                CMP     (R0)+,@0(R0)    ;SRCM=2 DSTM=7 DOP DEST NON-MOD
3066 025526  023010                        CMP     @(R0)+,(R0)     ;SRCM=3 DSTM=1 DOP DEST NON-MOD
3067 025530  023020                        CMP     @(R0)+,(R0)+    ;SRCM=3 DSTM=2 DOP DEST NON-MOD
3068 025532  023030                        CMP     @(R0)+,@(R0)+   ;SRCM=3 DSTM=3 DOP DEST NON-MOD
3069 025534  023050                        CMP     @(R0)+,@-(R0)   ;SRCM=3 DSTM=5 DOP DEST NON-MOD
3070 025536  023070  000000                CMP     @(R0)+,@0(R0)   ;SRCM=3 DSTM=7 DOP DEST NON-MOD
3071 025542  025010                        CMP     @-(R0),(R0)     ;SRCM=5 DSTM=1 DOP DEST NON-MOD
3072 025544  025020                        CMP     @-(R0),(R0)+    ;SRCM=5 DSTM=2 DOP DEST NON-MOD
3073 025546  025030                        CMP     @-(R0),@(R0)+   ;SRCM=5 DSTM=3 DOP DEST NON-MOD
3074 025550  025050                        CMP     @-(R0),@-(R0)   ;SRCM=5 DSTM=5 DOP DEST NON-MOD
3075 025552  025070  000000                CMP     @-(R0),@0(R0)   ;SRCM=5 DSTM=7 DOP DEST NON-MOD
3076 025556  027010  000000                CMP     @0(R0),(R0)     ;SRCM=7 DSTM=1 DOP DEST NON-MOD
3077 025562  027020  000000                CMP     @0(R0),(R0)+    ;SRCM=7 DSTM=2 DOP DEST NON-MOD
3078 025566  027030  000000                CMP     @0(R0),@(R0)+   ;SRCM=7 DSTM=3 DOP DEST NON-MOD
3079 025572  027050  000000                CMP     @0(R0),@-(R0)   ;SRCM=7 DSTM=5 DOP DEST NON-MOD
3080 025576  027070  000000  000000        CMP     @0(R0),@0(R0)   ;SRCM=7 DSTM=7 DOP DEST NON-MOD
3081 025604  005037  177572                CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3082                               ;*       TEST DOP DEST MOD AND NOT SUB; DSTM=1,2,3,5,7
3083                               ;*
3084 025610  005000                        CLR     R0              ;SET UP CONSTANTS FOR TEST
3085 025612  012701  060000                MOV     #60000,R1
3086 025616  012737  025630  001110        MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
3087 025624  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3088 025630  050011                4$:     BIS     R0,(R1)         ;DSTM=1 DOP DEST MOD
3089 025632  050021                        BIS     R0,(R1)+        ;DSTM=2 DOP DEST MOD
3090 025634  050031                        BIS     R0,@(R1)+       ;DSTM=3 DOP DEST MOD
3091 025636  050051                        BIS     R0,@-(R1)       ;DSTM=5 DOP DEST MOD
3092 025640  050071  000000                BIS     R0,@0(R1)       ;DSTM=7 DOP DEST MOD
3093 025644  005037  177572                CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3094                               ;*       TEST MOV DEST AND NOT(SM0 AND DM0); DSTM=3,5,7
3095                               ;*
3096 025650  012701  060000                MOV     #60000,R1       ;SET UP CONSTANTS FOR TEST
3097 025654  012737  060002  060000        MOV     #60002,@#60000
3098 025662  012737  025674  001110        MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
3099 025670  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3100 025674  010031                5$:     MOV     R0,@(R1)+       ;DSTM=3 MOV DEST
3101 025676  010051                        MOV     R0,@-(R1)       ;DSTM=5 MOV DEST
3102 025700  010071  000000                MOV     R0,@0(R1)       ;DSTM=7 MOV DEST
3103 025704  005037  177572                CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3104                               ;*       TEST SWAB; DSTM=1,2,3,5,7
3105                               ;*
3106 025710  012701  060000                MOV     #60000,R1       ;SET UP CONSTANTS FOR TEST
3107 025714  012737  025726  001110        MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
3108 025722  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3109 025726  000311                6$:     SWAB    (R1)            ;DSTM=1 SWAB
```

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

```
3110 025730  000321                        SWAB    (R1)+           ;DSTM=2 SWAB
3111 025732  000331                        SWAB    @(R1)+          ;DSTM=3 SWAB
3112 025734  000351                        SWAB    @-(R1)          ;DSTM=5 SWAB
3113 025736  000371  000000                SWAB    @0(R1)          ;DSTM=7 SWAB
3114 025742  005037  177572                CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3115                              ;*       TEST ROT/SHFT; DSTM=1,2,3,5,7
3116                              ;*
3117 025746  012701  060000                MOV     #60000,R1       ;SET UP CONSTANTS FOR TEST
3118 025752  012702  060006                MOV     #60006,R2
3119 025756  010203                        MOV     R2,R3
3120 025760  012737  060000  060000        MOV     #60000,@#60000
3121 025766  012737  060002  060002        MOV     #60002,@#60002
3122 025774  012737  060004  060004        MOV     #60004,@#60004
3123 026002  012737  060006  060006        MOV     #60006,@#60006
3124 026010  012737  026022  001110        MOV     #7$,$LPERR      ;SET LOOP ON ERROR POINTER TO 7$
3125 026016  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3126 026022  006111                7$:     ROL     (R1)            ;DSTM=1 ROT/SHFT
3127 026024  006121                        ROL     (R1)+           ;DSTM=2 ROT/SHFT
3128 026026  006131                        ROL     @(R1)+          ;DSTM=3 ROT/SHFT
3129 026030  006152                        ROL     @-(R2)          ;DSTM=5 ROT/SHFT
3130 026032  006173  000000                ROL     @0(R3)          ;DSTM=7 ROT/SHFT
3131 026036  005037  177572                CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3132                              ;*       TEST MUL/DIV; DSTM=1,2,3,5,7
3133                              ;*
3134 026042  012737  026056  001110        MOV     #8$,$LPERR      ;SET LOOP ON ERROR POINTER TO 8$
3135 026050  005003                        CLR     R3              ;SET UP CONSTANT FOR TEST
3136 026052  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3137 026056  070310                8$:     MUL     (R0),R3         ;DSTM=1 MUL/DIV
3138 026060  070320                        MUL     (R0)+,R3        ;DSTM=2 MUL/DIV
3139 026062  070330                        MUL     @(R0)+,R3       ;DSTM=3 MUL/DIV
3140 026064  070350                        MUL     @-(R0),R3       ;DSTM=5 MUL/DIV
3141 026066  070370  000000                MUL     @0(R0),R3       ;DSTM=7 MUL/DIV
3142 026072  005037  177572                CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3143                              ;*       TEST ASH/ASHC; DSTM=1,2,3,5,7
3144                              ;*
3145 026076  012737  000001  060000        MOV     #1,@#60000      ;SET UP CONSTANTS FOR THE TEST
3146 026104  012737  060000  060002        MOV     #60000,@#60002
3147 026112  012700  060000                MOV     #60000,R0
3148 026116  012737  026130  001110        MOV     #9$,$LPERR      ;SET LOOP ON ERROR POINTER TO 9$
3149 026124  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3150 026130  072310                9$:     ASH     (R0),R3         ;DSTM=1 ASH/ASHC
3151 026132  072320                        ASH     (R0)+,R3        ;DSTM=2 ASH/ASHC
3152 026134  072330                        ASH     @(R0)+,R3       ;DSTM=3 ASH/ASHC
3153 026136  072350                        ASH     @-(R0),R3       ;DSTM=5 ASH/ASHC
3154 026140  072370  000000                ASH     @0(R0),R3       ;DSTM=7 ASH/ASHC
3155 026144  005037  177572                CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3156                              ;*       TEST JMP; DSTM=3,7
3157                              ;*
3158 026150  012737  026202  001110        MOV     #10$,$LPERR     ;SET LOOP ON ERROR POINTER TO 10$
3159 026156  012737  026204  060000        MOV     #11$,@#60000    ;SET UP CONSTANTS FOR THE TEST
3160 026164  012737  026210  060002        MOV     #12$,@#60002
3161 026172  012701  060000                MOV     #60000,R1
3162 026176  005237  177572                INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3163 026202  000131                10$:    JMP     @(R1)+          ;DSTM=3 JMP
3164 026204  000171  000000        11$:    JMP     @0(R1)          ;DSTM=7 JMP
3165 026210  005037  177572        12$:    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3166                              ;*       TEST JSR; DSTM=3,7
```

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

```
3167                                    ;*
3168 026214  012737  026246  001110          MOV     #13$,$LPERR     ;SET LOOP ON ERROR POINTER TO 13$
3169 026222  012737  026250  060000          MOV     #14$,@#60000    ;SET UP CONSTANTS FOR THE TEST
3170 026230  012737  026254  060002          MOV     #15$,@#60002
3171 026236  012701  060000                  MOV     #60000,R1
3172 026242  005237  177572                  INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3173 026246  004731                  13$:    JSR     PC,@(R1)+       ;DSTM=3 JSR
3174 026250  004771  000000          14$:    JSR     PC,@0(R1)       ;DSTM=7 JSR
3175 026254  005037  177572          15$:    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3176 026260  112737  000006  172306          MOVB    #6,KIPDR3       ;MAKE KIPDR3 RESIDENT
3177 026266  012706  001100                  MOV     #KERSTK,KSP     ;RESET STACK POINTER
```

```
3186                                    .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'.
                                        ;*
                                        ;;*****************************************************************
    026272  000004                      TST16:  SCOPE
3187 026274  105037  172222             20$:    CLRB    SDPDR1          ;MAKE SDPDR1 NON-RESIDENT
3188 026300  012737  026326  001110             MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
3189 026306  012737  000022  172516             MOV     #22,MMR3        ;ENABLE 22-BIT SUPERVISOR D-SPACE
3190 026314  052737  040000  177776             BIS     #40000,PSW      ;ENABLE SUPERVISOR MODE
3191 026322  005237  177572                     INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3192                                    ;*      THE NEXT INSTRUCTIONS SHOULD NEVER INVOKE D-SPACE
3193                                    ;*
3194 026326  000400                     1$:     BR      2$
3195 026330  005700                     2$:     TST     R0
3196 026332  005200                             INC     R0
3197 026334  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3198 026340  112737  000006  172222             MOVB    #6,SDPDR1       ;MAKE SDPDR1 RESIDENT
3199 026346  105037  172206                     CLRB    SIPDR3          ;MAKE SIPDR3 NON-RESIDENT
3200                                    ;*      TEST SOP INSTRUCTIONS
3201                                    ;*
3202 026352  012737  026364  001110             MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
3203 026360  005237  177572                     INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3204 026364  005737  060000             3$:     TST     @#60000         ;DSTM=3 DSTF=7 SOP NON-MOD
3205 026370  005037  060000                     CLR     @#60000         ;DSTM=3 DSTF=7 SOP MOD
3206 026374  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3207                                    ;*      TEST DOP INSTUCTIONS
3208                                    ;*
3209 026400  012737  026416  001110             MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
3210 026406  012700  060000                     MOV     #60000,R0       ;SET UP CONSTANT FOR TEST
3211 026412  005237  177572                     INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3212 026416  023710  060000             4$:     CMP     @#60000,(R0)    ;SRCM=3 DSTM=1 DOP DEST NON-MOD
3213 026422  052730  000000                     BIS     #0,@(R0)+       ;SRCM=2 DSTM=3 DOP DEST MOD
3214 026426  013737  060002  060002             MOV     @#60002,@#60002 ;SRCM=3 DSTM=3 MOV DEST
3215 026434  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3216                                    ;*      TEST ROT/SHFT AND ASH/ASHC INSTRUCTIONS
3217                                    ;*
3218 026440  012737  026464  001110             MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
3219 026446  012737  000001  060000             MOV     #1,@#60000      ;SET CONSTANT FOR TEST
3220 026454  012700  000001                     MOV     #1,R0           ;SET UP CONSTANT FOR TEST
3221 026460  005237  177572                     INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3222 026464  006137  060000             5$:     ROL     @#60000         ;DSTM=3 ROT/SHFT
3223 026470  072337  060000                     ASH     @#60000,R3      ;DSTM=3 ASH/ASHC
3224 026474  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3225                                    ;*      TEST MUL/DIV AND SWAB INSTRUCTIONS
3226                                    ;*
3227 026500  012737  026512  001110             MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
3228 026506  005237  177572                     INC     MMR0            ;TURN ON MEMORY MANAGEMENT
3229 026512  070037  060000             6$:     MUL     @#60000,R0      ;DSTM=3 MUL/DIV
3230 026516  000337  060004                     SWAB    @#60004         ;DSTM=3 SWAB
3231 026522  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
3232 026526  042737  000002  172516             BIC     #2,MMR3         ;DISABLE SUPERVISOR D-SPACE
```

```
3233 026534  112737  000006  172206          MOVB    #6,SIPDR3       ;MAKE SIPDR3 RESIDENT
```

```
 3242                                   .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'.
                                        ;*
                                        ;;*****************************************************************
     026542  000004                     TST17:  SCOPE
 3243 026544  105037  177622            20$:    CLRB    UDPDR1          ;MAKE UDPDR1 NON-RESIDENT
 3244 026550  012737  026576  001110            MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
 3245 026556  012737  000021  172516            MOV     #21,MMR3        ;ENABLE 22-BIT USER D-SPACE
 3246 026564  052737  140000  177776            BIS     #140000,PSW     ;ENABLE USER MODE
 3247 026572  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
 3248                                   ;*       THE NEXT INSTRUCTIONS SHOULD NEVER INVOKE D-SPACE
 3249                                   ;*
 3250 026576  000400                    1$:     BR      2$
 3251 026600  005700                    2$:     TST     R0
 3252 026602  005200                            INC     R0
 3253 026604  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
 3254 026610  112737  000006  177622            MOVB    #6,UDPDR1       ;MAKE UDPDR1 RESIDENT
 3255 026616  105037  177606                    CLRB    UIPDR3          ;MAKE UIPDR3 NON-RESIDENT
 3256                                   ;*       TEST SOP INSTRUCTIONS
 3257                                   ;*
 3258 026622  012737  026634  001110            MOV     #3$,$LPERR      ;SET LOOP ON ERROR POINTER TO 3$
 3259 026630  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
 3260 026634  005737  060000            3$:     TST     @#60000         ;DSTM=3 DSTF=7 SOP NON-MOD
 3261 026640  005037  060000                    CLR     @#60000         ;DSTM=3 DSTF=7 SOP MOD
 3262 026644  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
 3263                                   ;*       TEST DOP INSTUCTIONS
 3264                                   ;*
 3265 026650  012737  026666  001110            MOV     #4$,$LPERR      ;SET LOOP ON ERROR POINTER TO 4$
 3266 026656  012700  060000                    MOV     #60000,R0       ;SET UP CONSTANT FOR TEST
 3267 026662  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
 3268 026666  023710  060000            4$:     CMP     @#60000,(R0)    ;SRCM=3 DSTM=1 DOP DEST NON-MOD
 3269 026672  052730  000000                    BIS     #0,@(R0)+       ;SRCM=2 DSTM=3 DOP DEST MOD
 3270 026676  013737  060002  060002            MOV     @#60002,@#60002 ;SRCM=3 DSTM=3 MOV DEST
 3271 026704  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
 3272                                   ;*       TEST ROT/SHFT AND ASH/ASHC INSTRUCTIONS
 3273                                   ;*
 3274 026710  012737  026730  001110            MOV     #5$,$LPERR      ;SET LOOP ON ERROR POINTER TO 5$
 3275 026716  012737  000001  060000            MOV     #1,@#60000      ;SET CONSTANT FOR TEST
 3276 026724  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
 3277 026730  006137  060000            5$:     ROL     @#60000         ;DSTM=3 ROT/SHFT
 3278 026734  072337  060000                    ASH     @#60000,R3      ;DSTM=3 ASH/ASHC
 3279 026740  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
 3280                                   ;*       TEST MUL/DIV AND SWAB INSTRUCTIONS
 3281                                   ;*
 3282 026744  012737  026762  001110            MOV     #6$,$LPERR      ;SET LOOP ON ERROR POINTER TO 6$
 3283 026752  012700  000001                    MOV     #1,R0           ;SET UP CONSTANT FOR TEST
 3284 026756  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
 3285 026762  070037  060000            6$:     MUL     @#60000,R0      ;DSTM=3 MUL/DIV
 3286 026766  000337  060004                    SWAB    @#60004         ;DSTM=3 SWAB
 3287 026772  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
 3288 026776  042737  000001  172516            BIC     #1,MMR3         ;DISABLE USER D-SPACE
```

```
 3289 027004  112737  000006  177606          MOVB    #6,UIPDR3       ;MAKE UIPDR3 RESIDENT
 3290 027012  005037  177776                  CLR     PSW             ;RESET TO KERNAL SPACE
```

```
 3299                                   .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$. EACH PSW IN
                                        ;*      VIRTUAL 252 IS DIFFERENT SO THE PROGRAM CAN TELL WHICH
                                        ;*      AREA IT IS PICKED UP FROM.
                                        ;;*****************************************************************
     027016  000004                     TST20:  SCOPE
 3300 027020  004737  002356                     JSR     PC,TOFF         ;TURN OF T-BIT FOR THIS TEST
 3301 027024  012737  027060  001110  20$:     MOV     #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
 3302 027032  012737  027126  000250          MOV     #10$,@#MMVEC    ;SET M.M. VEC. TO HOLD BAD VECTOR
 3303 027040  005037  000252                  CLR     @#MMVEC+2       ;PSW IN 252 HAS PRIORITY OF ZERO
 3304 027044  012737  027134  000350          MOV     #15$,@#350      ;SET D-SPACE M.M VECTOR TO 350
 3305 027052  012737  000340  000352          MOV     #340,@#352      ;SET PSW PRIORITY TO 7
 3306 027060  012706  001000          1$:     MOV     #1000,KSP       ;SET UP KERNAL VECTOR
 3307 027064  005237  177572                  INC     MMR0            ;TURN ON MEMORY MANAGEMENT
 3308                                   ;NOW SET UP FOR AN ABORT IN KERNAL MODE WITH D-SPACE ENABLED
 3309                                   ;
 3310 027070  012737  077402  172330          MOV     #77402,KDPDR4   ;KERNAL D-SPACE PAGE 4 IS READ ONLY
 3311 027076  012737  000600  172370          MOV     #600,KDPAR4     ;MAP D-SPACE PAGE 4 TO 12K
 3312 027104  052737  000004  172516          BIS     #BIT2,MMR3      ;ENABLE KERNAL D-SPACE MAPPING
 3313 027112  012737  000001  172360          MOV     #1,KDPAR0       ;MAP KERNAL D PAGE 0 TO 000100
 3314 027120  012737  177777  100000          MOV     #-1,@#100000    ;TRY TO WRITE TO PAGE 4
 3315 027126  013700  177776          10$:    MOV     PSW,R0          ;SAVE PSW FOR COMPARE
 3316 027132  000402                          BR      16$             ;BRANCH TO D-SPACE READ CODE
 3317 027134  013700  177776          15$:    MOV     PSW,R0          ;SAVE PSW FOR COMPARE
 3318 027140  005037  172360          16$:    CLR     KDPAR0          ;REMAP KERNAL D PAGE 0 TO PHYSICAL 0
 3319 027144  012706  001100                  MOV     #KERSTK,KSP     ;RESET STACK POINTER AFTER D-SPACE ABORT
 3320 027150  042737  000004  172516          BIC     #BIT2,MMR3      ;TURN OFF KERNAL D-SPACE ENABLE
 3321 027156  013701  177572                  MOV     MMR0,R1         ;SAVE MMR0 FOR COMPARE
 3322 027162  013702  177574                  MOV     MMR1,R2         ;SAVE MMR1 FOR COMPARE
 3323 027166  013703  177576                  MOV     MMR2,R3         ;SAVE MMR2 FOR COMPARE
 3324 027172  122700  000340                  CMPB    #340,R0         ;DID YOU PICK CORRECT PSW
 3325 027176  001401                          BEQ     2$              ;BRANCH IF PSW IS 340
 3326 027200  104033                          ERROR   +33             ;WRONG PSW PICKED IN ABORT SEQUENCE
 3327 027202  022701  020031          2$:     CMP     #020031,R1      ;EXPECTING READ ONLY ABORT
 3328                                                                 ;KERNAL MODE D-SPACE PAGE 4
 3329 027206  001401                          BEQ     3$              ;BRANCH IF CONDITION IS CORRECT
 3330 027210  104027                          ERROR   +27             ;WRONG M.M. ABORT CONDITION
 3331 027212  005037  177572          3$:     CLR     MMR0            ;CLEAR OFF MMR0 FOR EXIT OF TEST
 3332 027216  012737  016142  000250          MOV     #MGMERR,MMVEC   ;RESTORE NORMAL M.M. TRAP VECTOR
 3333 027224  012737  000340  000252          MOV     #340,MMVEC+2    ;RESTORE TRAP PSW (PRIORITY=7)
 3334
```

```
                        .SBTTL  GROUP 3  MOVE TO/FROM PREVIOUS INSTRUCTIONS TESTS
                        ;/.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\
                        ;;*****************************************************************
                        ;*
                        ;*      THIS GROUP OF TESTS WILL TEST ALL THE LOGIC ASSOCIATED WITH
                        ;*      THE 'MFPI', 'MTPI' AND 'MFPD' INSTRUCTIONS.
                        ;*
                        ;;*****************************************************************
                        ;/.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\./.\
```

```
 3359                                   .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.
                                        ;*
                                        ;;*****************************************************************
    027232  000004                      TST21:  SCOPE
 3360 027234  004737  002060                    JSR     PC,APRINIT      ;INIT ALL PAR/PDR'S
 3361 027240  012737  000001  177572            MOV     #1,MMR0         ;TURN ON MEMORY MANAGEMENT
 3362 027246  012737  000600  172350  1$.       MOV     #600,KIPAR4     ;MAP KIPAR4 TO 12K
 3363 027254  012737  000600  172250            MOV     #600,SIPAR4     ;MAP SIPAR4 TO 12K
 3364 027262  012700  036514                    MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
 3365 027266  010037  100000                    MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 60000
 3366 027272  105037  172310                    CLRB    KIPDR4          ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
 3367                                   ;THE FOLLOWING WILL TEST DSTM=0 MFPI
 3368 027276  012737  027304  001110            MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
 3369 027304  012737  010340  177776  2$:       MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
 3370 027312  006506                          MFPI    SSP             ;PUT SUPERVISOR STACK POINTER ON KERNEL STACK
 3371 027314  022706  001100                    CMP     #KERSTK,KSP     ;WAS SOMETHING PUSHED ON STACK AT 6$
 3372 027320  001407                          BEQ     3$              ;BRANCH IF NOTHING WAS PUSHED
 3373 027322  012600                          MOV     (KSP)+,R0       ;POP KERNEL STACK INTO R0
 3374 027324  012701  000700                    MOV     #SUPSTK,R1      ;EXPECTING TO GET 700 AS SSP
 3375 027330  020001                          CMP     R0,R1           ;DID YOU GET THE RIGHT POINTER?
 3376 027332  001403                          BEQ     4$              ;BRANCH IF YOU DID
 3377 027334  104023                          ERROR   +23             ;WRONG THING WAS PUSHED ON STACK
 3378                                   ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 4$' WITH 'BR 2$' = 000764
 3379 027336  000401                          BR      4$              ;BRANCH TO NEXT TRY
 3380 027340  104025                    3$:     ERROR   +25             ;NOTHING PUSHED ON STACK
 3381                                   ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 3$' ABOVE WITH 'BR 2$' = 000771
 3382
 3383                                   ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
 3384 027342  012737  002516  001110  4$:       MOV     #MFPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MFPILP IN SUBROUTINE
 3385 027350  012700  036514                    MOV     #36514,R0       ;RELOAD DATA PATTERN IN R0
 3386 027354  012737  010340  002520            MOV     #010340,MFPIPS  ;MAKE PREVIOUS MODE SUPERVISOR IN SUBROUTINE
 3387 027362  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
 3388 027366  012737  027566  002622            MOV     #MFPIV1,MFPIVC  ;LOAD ADDRESS OF THIS TEST'S TRAP CATCHER TO MFPIVC
 3389 027374  004737  002464                    JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
 3390 027400  006512                          MFPI    (R2)            ;<HERE - READ FROM PHYSICAL 60000
 3391 027402  000240                          NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
 3392 027404  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
 3393 027406  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
 3394                                   ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
 3395 027410  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
 3396 027414  004737  002464                    JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
 3397 027420  006522                          MFPI    (R2)+           ;<HERE - READ FROM PHYSICAL 60000
 3398 027422  000240                          NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
 3399 027424  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
```

```
3400 027426  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3401                              ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
3402 027430  004737  002464                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3403 027434  006537  100000                  MFPI    @#100000        ;<HERE - READ FROM PHYSICAL 60000
3404 027440  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3405 027442  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3406                              ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3407 027444  012702  100002                  MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3408 027450  004737  002464                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3409 027454  006542                          MFPI    -(R2)           ;<HERE - READ FROM PHYSICAL 60000
3410 027456  000240                          NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3411 027460  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3412 027462  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3413                              ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3414 027464  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3415 027472  012702  001204                  MOV     #<$TMP2+2>,R2   ;LOAD ADDR. OF $TMP2+2 INTO R2
3416 027476  004737  002464                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3417 027502  006552                          MFPI    @-(R2)          ;<HERE - READ FROM PHYSICAL 60000
3418 027504  000240                          NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3419 027506  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3420 027510  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3421                              ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3422 027512  005002                          CLR     R2              ;MAKE REGISTER 2 A ZERO
3423 027514  004737  002464                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3424 027520  006562  100000                  MFPI    100000(R2)      ;<HERE - READ FROM PHYSICAL 60000
3425 027524  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3426 027526  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3427                              ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3428 027530  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD TEST LOC. V.A. INTO $TMP2
3429 027536  012702  001202                  MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3430 027542  004737  002464                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3431 027546  006572  000000                  MFPI    @0(R2)          ;USE $TMP2 TO FETCH VIRTUAL ADDRESS OF 60000
3432 027552  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3433 027554  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3434 027556  112737  000006  172310          MOVB    #6,KIPDR4       ;MAKE KIPDR4 RESIDENT
3435 027564  000423                          BR      TST22           ;;BRANCH TO NEXT TEST
```

```
                                        .SBTTL  MM TRAP CATCHER FOR ABOVE TEST
027566  012637  001260          MFPIV1: MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
027572  012637  001262                  MOV     (KSP)+,TRAPPS
027576  013737  177572  001264          MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
027604  013737  177576  001270          MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
027612  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0 AND LEAVE
027620  104026                          ERROR   +26             ;TRIED TO READ NON-RESIDENT PAGE
                        ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI' = 000002
027622  013746  001262                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
027626  013746  001260                  MOV     TRAPPC,-(KSP)
027632  000002                          RTI
```

```
3462                                    .SBTTL  TEST # 22 - MOVE FROM PREVIOUS (USER) I-SPACE
                                        ;;*****************************************************************
                                        ;*TEST 22       MO'  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.
                                        ;*
                                        ;;*****************************************************************
    027634  000004                      TST22:  SCOPE
3463 027636  012700  036514             1$:     MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
3464 027642  012737  000600  177650             MOV     #600,UIPAR4     ;MAP UIPAR4 TO 12K
3465 027650  010037  100000                     MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 60000
3466 027654  105037  172310                     CLRB    KIPDR4          ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
3467                                    ;THE FOLLOWING WILL TEST DSTM=0 MFPI
3468 027660  012737  027666  001110             MOV     #2$,$LPERR      ;SET LOOP ON ERROR POINTER TO 2$
3469 027666  012737  030340  177776  2$:        MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3470 027674  012737  030164  000250             MOV     #MFPIV2,MMVEC   ;LOAD ADDRESS OF THIS TEST'S TRAP CATCHER TO MMVEC
3471 027702  006506                             MFPI    USP             ;PUT USER STACK POINTER ON KERNEL STACK
3472 027704  012737  016142  000250             MOV     #MGMERR,MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
3473 027712  022706  001076                     CMP     #KERSTK-2,KSP   ;WAS SOMETHING PUSHED ON STACK BY THE MFPI?
3474 027716  001007                             BNE     3$              ;BRANCH TO ERROR IF POINTER IS WRONG
3475 027720  012600                             MOV     (KSP)+,R0       ;POP KERNEL STACK INTO R0
3476 027722  012701  000600                     MOV     #USESTK,R1      ;EXPECTING TO GET 600 AS USP
3477 027726  020001                             CMP     R0,R1           ;DID YOU GET THE RIGHT POINTER?
3478 027730  001403                             BEQ     4$              ;BRANCH IF YOU DID
3479 027732  104023                             ERROR   +23             ;WRONG THING WAS PUSHED ON STACK
3480                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 4$" WITH 'BR 2$" = 000764
3481 027734  000401                             BR      4$              ;BRANCH TO NEXT TRY
3482 027736  104025                     3$:     ERROR   +25             ;NOTHING PUSHED ON STACK
3483                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BNE 3$" ABOVE WITH 'BR 2$" = 000771
3484                                            ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3485 027740  012737  002516  001110     4$:     MOV     #MFPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MFPILP IN SUBROUTINE
3486 027746  012737  030340  002520             MOV     #030340,MFPIPS  ;MAKE PREVIOUS MODE USER IN SUBROUTINE LOCATION
3487 027754  012700  036514                     MOV     #36514,R0       ;RELOAD DATA PATTERN IN R0
3488 027760  012737  030164  002622             MOV     #MFPIV2,MFPIVC  ;LOAD ADDRESS OF THIS TEST'S TRAP CATCHER TO MFPIVC
3489 027766  012702  100000                     MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3490 027772  004737  002464                     JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3491 027776  006512                             MFPI    (R2)            ;<HERE - READ FROM PHYSICAL 60000
3492 030000  000240                             NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3493 030002  104023                             ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3494 030004  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3495                                    ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
3496 030006  012702  100000                     MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3497 030012  004737  002464                     JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3498 030016  006522                             MFPI    (R2)+           ;<HERE - READ FROM PHYSICAL 60000
3499 030020  000240                             NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3500 030022  104023                             ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3501 030024  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3502                                    ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
```

```
3503 030026  004737  002464                JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3504 030032  006537  100000                MFPI    @#100000        ;<HERE - READ FROM PHYSICAL 60000
3505 030036  104023                        ERROR   +23             ;RETURN IS EHRE FOR ERROR - WRONG DATA WAS FETCHED
3506 030040  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3507                              ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3508 030042  012702  100002                MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3509 030046  004737  002464                JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3510 030052  006542                        MFPI    -(R2)           ;<HERE - READ FROM PHYSICAL 60000
3511 030054  000240                        NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3512 030056  104023                        ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3513 030060  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3514                              ;THE FOLLOWING WILL TEST DSTM-5 MFPI.
3515 030062  012737  100000  001202        MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3516 030070  012702  001204                MOV     #<$TMP2+2>,R2   ;LOAD ADDR. OF $TMP2+2 INTO R2
3517 030074  004737  002464                JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3518 030100  006552                        MFPI    @-(R2)          ;<HERE - READ FROM PHYSICAL 60000
3519 030102  000240                        NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3520 030104  104023                        ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3521 030106  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3522                              ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3523 030110  005002                        CLR     R2              ;MAKE REGISTER 2 A ZERO
3524 030112  004737  002464                JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3525 030116  006562  100000                MFPI    100000(R2)      ;<HERE - READ FROM PHYSICAL 60000
3526 030122  104023                        ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3527 030124  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3528                              ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3529 030126  012737  100000  001202        MOV     #100000,$TMP2   ;LOAD TEST LOC. V.A. INTO $TMP2
3530 030134  012702  001202                MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3531 030140  004737  002464                JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION FOUND
3532 030144  006572  0000( )               MFPI    @0(R2)          ;USE $TMP2 TO FETCH VIRTUAL ADDRESS OF 60000
3533 030150  104023                        ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3534 030152  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3535 030154  112737  000006  172310        MOVB    #6,KIPDR4       ;MAKE KIPDR4 RESIDENT
3536 030162  000423                        BR      TST23           ;;BRANCH TO NEXT TEST
```

```
                                                .SBTTL  MM TRAP CATCHER FOR PREVIOUS TEST
030164  012637  001260            MFPIV2: MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
030170  012637  001262                    MOV     (KSP)+,TRAPPS
030174  013737  177572  001264            MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
030202  013737  177576  001270            MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
030210  042737  160000  177572            BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0 AND LEAVE
030216  104026                            ERROR   +26             ;TRIED TO READ NON-RESIDENT PAGE
                                        ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN "RTI" = 000002
030220  013746  001262                    MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
030224  013746  001260                    MOV     TRAPPC,-(KSP)
030230  000002                            RTI
```

```
3563                                    .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.
                                        ;*
                                        ;;*****************************************************************
    030232  000004                      TST23:  SCOPE
3564 030234  012737  077406  172210     1$:     MOV     #77406,SIPDR4   ;SUPERVISOR I-SPACE PAGE 4 READ/WRITE
3565                                    ;THE FOLLOWING WILL TEST DSTM=0 MTPI
3566 030242  012737  010340  177776     2$:     MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
3567 030250  012746  007777             MOV     #7777,-(KSP)    ;PUSH DATA ON KERNEL STACK
3568 030254  012737  030614  000250     MOV     #MTPIV1,MMVEC   ;LOAD ADDRESS OF THIS TEST'S TRAP CATCHER TO MMVEC
3569 030262  006606                     MTPI    SSP             ;LOAD SUPERVISOR STACK POINTER
3570 030264  006506                     MFPI    SSP             ;READ SUPERVISOR STACK POINTER
3571 030266  012737  016142  000250     MOV     #MGMERR,MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
3572 030274  012601                     MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3573 030276  022701  007777             CMP     #7777,R1        ;WAS SUPERVISOR STACK POINTER CHANGED
3574 030302  001401                     BEQ     3$              ;BRANCH IF IT WAS
3575 030304  104025                     ERROR   +25             ;SUPERVISOR STACK POINTER NOT CHANGED
3576                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 3$' WITH 'BR 2$' = 000765
3577 030306  012737  010340  177776     3$:     MOV     #010340,PSW     ;MAKE PREVIOUS MODE SUPERVISOR
3578 030314  012746  000700             MOV     #SUPSTK,-(KSP)  ;GET READY TO RESTORE SUPERVISOR S. POINT
3579 030320  012737  030614  000250     MOV     #MTPIV1,MMVEC   ;LOAD ADDRESS OF THIS TEST'S TRAP CATCHER TO MMVEC
3580 030326  006606                     MTPI    SSP             ;RESTORE SUPERVISOR STACK POINTER
3581 030330  012737  016142  000250     MOV     #MGMERR,MMVEC   ;SET M.M. VECTOR TO NORMAL ROUTINE
3582 030336                             4$:     ;THIS WILL TEST DSTM = 1 MTPI.
3583 030336  012737  002654  001110     MOV     #MTPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MTPILP IN SUBROUTINE
3584 030344  012737  010340  002656     MOV     #010340,MTPIPM  ;MAKE PREVIOUS MODE SUPER IN LOCATION IN SUBROUTINE
3585 030352  012702  100000             MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3586 030356  012700  125252             MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3587 030362  004737  002626             JSR     PC,MTPITS       ;GO DO THE TEST USING THE MTPI INSTRUCTION FOUND
3588 030366  006612                     MTPI    (R2)            ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3589 030370  000240                     NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3590 030372  000000                     .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3591 030374  104024                     ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3592 030376  005726                     TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3593                                    ;THE FOLLOWING WILL TEST DSTM=2 MTPI.
3594 030400  012700  125252             MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3595 030404  012702  100000             MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3596 030410  004737  002626             JSR     PC,MTPITS       ;GO DO THE TEST USING THE MTPI INSTRUCTION FOUND
3597 030414  006622                     MTPI    (R2)+           ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3598 030416  000240                     NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3599 030420  177776                     .WORD   -2              ;ADD -2 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3600 030422  104024                     ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3601 030424  005726                     TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3602                                    ;THIS WILL TEST DSTM = 3 MTPI.
3603 030426  012700  052525             MOV     #52525,R0       ;LOAD TEST DATA INTO R0
```

```
3604 030432  004737  002626                JSR     PC,MTPITS       ;GO DO THE TEST USING THE MTPI INSTRUCTION FOUND
3605 030436  006637  100000                MTPI    @#100000        ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3606 030442  000000                        .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3607 030444  104024                        ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3608 030446  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3609                              ;THIS WILL TEST DSTM = 4 MTPI.
3610 030450  012700  125252                MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3611 030454  012702  100002                MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3612 030460  004737  002626                JSR     PC,MTPITS       ;GO DO THE TEST USING THE MTPI INSTRUCTION FOUND
3613 030464  006642                        MTPI    -(R2)           ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3614 030466  000240                        NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3615 030470  000000                        .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3616 030472  104024                        ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3617 030474  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3618                              ;THE FOLLOWING WILL TEST DSTM=5 MTPI.
3619 030476  012700  052525                MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3620 030502  012702  001204                MOV     #<$TMP2+2>,R2   ;LOAD ADDR. OF LOC. $TMP2+2 INTO R2
3621 030506  012737  100000  001202        MOV     #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOC. INTO $TMP2
3622 030514  004737  002626                JSR     PC,MTPITS       ;GO DO THE TEST USING THE MTPI INSTRUCTION FOUND
3623 030520  006652                        MTPI    @-(R2)          ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3624 030522  000240                        NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3625 030524  076576                        .WORD   100000-$TMP2    ;ADD 100000-$TMP2 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3626 030526  104024                        ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3627 030530  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3628                              ;THIS WILL TEST DSTM = 6 MTPI.
3629 030532  012700  052525                MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3630 030536  005002                        CLR     R2              ;MAKE REGISTER 2 ZERO
3631 030540  004737  002626                JSR     PC,MTPITS       ;GO DO THE TEST USING THE MTPI INSTRUCTION FOUND
3632 030544  006662  100000                MTPI    100000(R2)      ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3633 030550  100000                        .WORD   100000          ;ADD 100000 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3634 030552  104024                        ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3635 030554  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3636                              ;THE FOLLOWING WILL TEST DSTM=7 MTPI.
3637 030556  012700  125252                MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3638 030562  012737  100000  001202        MOV     #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOCATION INTO LOCATION $TMP2
3639 030570  012702  001202                MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3640 030574  004737  002626                JSR     PC,MTPITS       ;GO DO THE TEST USING THE MTPI INSTRUCTION FOUND
3641 030600  006672  000000                MTPI    @0(R2)          ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3642 030604  076576                        .WORD   100000-$TMP2    ;ADD 100000-$TMP2 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3643 030606  104024                        ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3644 030610  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3645 030612  000423                        BR      TST24           ;;BRANCH TO NEXT TEST
3646
3647 030614  012637  001260        MTPIV1: MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3648 030620  012637  001262                MOV     (KSP)+,TRAPPS
3649 030624  013737  177572  001264        MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
3650 030632  013737  177576  001270        MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3651 030640  042737  160000  177572        BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3652 030646  104024                        ERROR   +24             ;TRIED TO LOAD A N.R. PAGE 4
3653                              ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN "RTI" = 000002
3654 030650  013746  001262                MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3655 030654  013746  001260                MOV     TRAPPC,-(KSP)
3656 030660  000002                        RTI                     ;RETURN TO TEST
```

```
3657                                    .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.
                                        ;*
                                        ;;*****************************************************************
    030662  000004                      TST24:  SCOPE
3658 030664  012737  077406  177610     1$:     MOV     #77406,UIPDR4   ;USER I-SPACE PAGE 4 READ/WRITE
3659 030672  012737  000600  177650             MOV     #600,UIPAR4     ;MAP USER I PAGE 4 TO 12K
3660                                    ;THE FOLLOWING WILL TEST DSTM=0 MTPI
3661 030700  012737  030340  177776     2$:     MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3662 030706  012746  007777                     MOV     #7777,-(KSP)    ;PUSH DATA ON KERNEL STACK
3663 030712  012737  031260  000250             MOV     #MTPIV2,MMVEC   ;SET M.M. VECTOR TO 20$
3664 030720  006606                             MTPI    USP             ;LOAD USER STACK POINTER
3665 030722  006506                             MFPI    USP             ;READ USER STACK POINTER
3666 030724  012737  016142  000250             MOV     #MGMERR,MMVEC   ;RESTORE MM VECTOR TO NORMAL ROUTINE
3667 030732  012601                             MOV     (KSP)+,R1       ;POP KERNEL STACK INTO R1
3668 030734  022701  007777                     CMP     #7777,R1        ;WAS USER STACK POINTER CHANGED
3669 030740  001401                             BEQ     3$              ;BRANCH IF IT WAS
3670 030742  104025                             ERROR   +25             ;USER STACK POINTER NOT CHANGED
3671                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 3$' WITH 'BR 2$' = 000765
3672 030744  012737  030340  177776     3$:     MOV     #030340,PSW     ;MAKE PREVIOUS MODE USER
3673 030752  012746  000600                     MOV     #USESTK,-(KSP)  ;GET READY TO RESTORE USER S. POINT
3674 030756  012737  031260  000250             MOV     #MTPIV2,MMVEC   ;SET M.M. VECTOR TO 20$
3675 030764  006606                             MTPI    USP             ;RESTORE USER STACK POINTER
3676 030766  012737  016142  000250             MOV     #MGMERR,MMVEC   ;RESTORE MM VECTOR TO NORMAL ROUTINE
3677                                    ;THIS WILL TEST DSTM = 1 MTPI.
3678 030774  012737  002654  001110             MOV     #MTPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MTPILP IN SUBROUTINE
3679 031002  012737  030340  002656             MOV     #030340,MTPIPM  ;SET PREVIOUS MODE = USER IN SUBROUTINE LOCATION
3680 031010  012737  031260  002744             MOV     #MTPIV2,MTPIVC  ;SET THIS TEST'S MM TRAP HANDLER IN MTPIVC
3681 031016  012702  100000                     MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3682 031022  012700  125252                     MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3683 031026  004737  002626                     JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION LOCATED
3684 031032  006612                             MTPI    (R2)            ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3685 031034  000240                             NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3686 031036  000000                             .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3687 031040  104024                             ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3688 031042  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3689                                    ;THE FOLLOWING WILL TEST DSTM=2 MTPI.
3690 031044  012700  125252                     MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3691 031050  012702  100000                     MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3692 031054  004737  002626                     JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION LOCATED
3693 031060  006622                             MTPI    (R2)+           ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3694 031062  000240                             NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3695 031064  177776                             .WORD   -2              ;ADD -2 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3696 031066  104024                             ERROR   +24             ;RETUR' IS HERE FOR ERROR - INCORRECT STORE
3697 031070  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
```

```
3698                                    ;THIS WILL TEST DSTM = 3 MTPI.
3699 031072  012700  052525                MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3700 031076  004737  002626                JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION LOCATED
3701 031102  006637  100000                MTPI    @#100000        ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3702 031106  000000                        .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3703 031110  104024                        ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3704 03  12  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3705                                    ;THIS WILL TEST DSTM = 4 MTPI.
3706 031114  012700  125252                MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3707 031120  012702  100002                MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3708 031124  004737  002626                JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION LOCATED
3709 031130  006642                        MTPI    -(R2)           ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3710 031132  000240                        NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3711 031134  000000                        .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3712 031136  104024                        ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3713 031140  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3714                                    ;THE FOLLOWING WILL TEST DSTM=5 MTPI.
3715 031142  012700  052525                MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3716 031146  012702  001204                MOV     #<$TMP2+2>,R2   ;LOAD ADDR. OF LOC. $TMP2+2 INTO R2
3717 031152  012737  100000  001202        MOV     #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOC. INTO $TMP2
3718 031160  004737  002626                JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION LOCATED
3719 031164  006652                        MTPI    @-(R2)          ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3720 031166  000240                        NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3721 031170  076576                        .WORD   100000-$TMP2    ;ADD 100000-$TMP2 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3722 031172  104024                        ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3723 031174  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3724                                    ;THIS WILL TEST DSTM = 6 MTPI.
3725 031176  012700  052525                MOV     #52525,R0       ;LOAD TEST DATA INTO R0
3726 031202  005002                        CLR     R2              ;MAKE REGISTER 2 ZERO
3727 031204  004737  002626                JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION LOCATED
3728 031210  006662  100000                MTPI    100000(R2)      ;<HERE - LOAD TEST DATA INTO PHYSICAL 60000
3729 031214  100000                        .WORD   100000          ;ADD 100000 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3730 031216  104024                        ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3731 031220  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3732                                    ;THE FOLLOWING WILL TEST DSTM=7 MTPI.
3733 031222  012700  125252                MOV     #125252,R0      ;LOAD TEST DATA INTO R0
3734 031226  012737  100000  001202        MOV     #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOCATION
3735                                                                ;INTO LOCATION $TMP2
3736 031234  012702  001202                MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3737 031240  004737  002626                JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION LOCATED
3738 031244  006672  000000                MTPI    @0(R2)          ;<HERE LOAD TEST DATA INTO PHYSICAL 60000
3739 031250  076576                        .WORD   100000-$TMP2    ;ADD 100000-$TMP2 TO R2 AFTER MTPI INSTRUCTION EXECUTE
3740 031252  104024                        ERROR   +24             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
3741 031254  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3742 031256  000423                        BR      TST25           ;;BRANCH TO NEXT TEST
3743
3744 031260  012637  001260        MTPIV2: MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3745 031264  012637  001262                MOV     (KSP)+,TRAPPS
3746 031270  013737  177572  001264        MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
3747 031276  013737  177576  001270        MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3748 031304  042737  160000  177572        BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3749 031312  104024                        ERROR   +24             ;TRIED TO LOAD A N.R. PAGE 4
3750                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN "RTI" = 000002
3751 031314  013746  001262                MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3752 031320  013746  001260                MOV     TRAPPC,-(KSP)
3753 031324  000002                        RTI                     ;RETURN TO TEST
```

```
3768                                    .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.
                                        ;*
                                        ;;*****************************************************************
     031326  000004                     TST25:  SCOPE
3769 031330  012737  031364  001110             MOV     #1$,$LPERR      ;SET LOOP ON ERROR TO 1$
3770 031336  012737  040340  177776             MOV     #040340,PSW     ;GO TO SUPERVISOR MODE FOR THIS TEST
3771 031344  012700  036514                     MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
3772 031350  010037  100000                     MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 60000
3773 031354  012702  100000                     MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3774                                    ;THE FOLLOWING WILL TEST DSTM=0 MFPI
3775 031360  105037  172210                     CLRB    SIPDR4          ;MAKE SUPERVISOR I-SPACE PAGE 4 NON-RESIDENT
3776 031364  012737  040340  177776  1$:      MOV     #040340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT SUPERVISOR
3777 031372  012737  031674  000250             MOV     #MFPIV3,MMVEC   ;SET M.M. VECTOR TO MFPIV3
3778 031400  006506                     MFPI    KSP             ;PUT KERNEL STACK POINTER ON SUPERVISOR STACK
3779 031402  012737  016142  000250             MOV     #MGMERR,MMVEC   ;RESTORE MM VECTOR TO NORMAL ROUTINE
3780 031410  022706  000700                     CMP     #SUPSTK,SSP     ;WAS SOMETHING PUSHED ON STACK AT THE MFPI
3781 031414  001407                     BEQ     2$              ;BRANCH TO ERROR IF NOTHING WAS PUSHED
3782 031416  012600                     MOV     (SSP)+,R0       ;POP SUPERVISOR STACK INTO R0
3783 031420  012701  001100                     MOV     #KERSTK,R1      ;EXPECTING 1100 AS KSP
3784 031424  020001                     CMP     R0,R1           ;DID YOU GET THE RIGHT POINTER?
3785 031426  001403                     BEQ     3$              ;BRANCH IF YOU DID
3786 031430  104023                     ERROR   +23             ;WRONG THING WAS PUSHED ON STACK
3787                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 3$" WITH 'BR 1$" = 000756
3788 031432  000401                     BR      3$              ;BRANCH TO NEXT TRY
3789 031434  104025                     2$:     ERROR   +25             ;NOTHING PUSHED ON STACK
3790                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 2$" ABOVE WITH 'BR 1$" = 000762
3791                                    ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3792 031436  012737  002516  001110  3$:      MOV     #MFPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MFPILP IN SUBROUTINE
3793 031444  012737  040340  002520             MOV     #040340,MFPIPS  ;MAKE PREVIOUS MODE KERNEL PRESENT SUPERVISOR
3794 031452  012737  031674  002622             MOV     #MFPIV3,MFPIVC  ;MOVE THIS TEST'S MM TRAP VECTOR TO MFPIVC
3795 031460  012700  036514                     MOV     #36514,R0       ;LOAD DATA EXPECTED INTO R0
3796 031464  012702  100000                     MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3797 031470  004737  002464                     JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3798 031474  006512                     MFPI    (R2)            ;<HERE - READ FROM PHYSICAL 60000
3799 031476  000240                     NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3800 031500  104023                     ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3801 031502  005726                     TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3802                                    ;THE FOLLOWING WILL TEST DSM=2 MFPI.
3803 031504  012702  100000                     MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3804 031510  004737  002464                     JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3805 031514  006522                     MFPI    (R2)+           ;<HERE - READ FROM PHYSICAL 60000
3806 031516  000240                     NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3807 031520  104023                     ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3808 031522  005726                     TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
```

```
3809                                        ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
3810 031524  004737  002464                      JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3811 031530  006537  100000                      MFPI    @#100000        ;<HERE - READ FROM PHYSICAL 60000
3812 031534  104023                             ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3813 031536  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3814                                        ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3815 031540  012702  100002                      MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3816 031544  004737  002464                      JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3817 031550  006542                             MFPI    -(R2)           ;<HERE - READ FROM PHYSICAL 60000
3818 031552  000240                             NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3819 031554  104023                             ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3820 031556  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3821                                        ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3822 031560  012737  100000  001202              MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3823 031566  012702  001204                      MOV     #<$TMP2+2>,R2   ;LOAD ADDRESS OF $TMP2+2 INTO R2
3824 031572  004737  002464                      JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3825 031576  006552                             MFPI    @-(R2)          ;<HERE - READ FROM PHYSICAL 60000
3826 031600  000240                             NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3827 031602  104023                             ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3828 031604  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3829                                        ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3830 031606  005002                             CLR     R2              ;MAKE REGISTER 2 A ZERO
3831 031610  004737  002464                      JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3832 031614  006562  100000                      MFPI    100000(R2)      ;<HERE - READ FROM PHYSICAL 60000
3833 031620  104023                             ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3834 031622  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3835                                        ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3836 031624  012737  100000  001202              MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR. INTO $TMP2
3837 031632  012702  001202                      MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3838 031636  004737  002464                      JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3839 031642  006572  000000                      MFPI    @0(R2)          ;<HERE - READ FROM PHYSICAL 60000
3840 031646  104023                             ERROR   +23             ;WRONG DATA WAS FETCHED
3841 031650  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3842 031652  012737  000340  177776              MOV     #00340,PSW      ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
3843 031660  112737  000006  172210              MOVB    #6,SIPDR4       ;MAKE SIPDR4 RESIDENT
3844 031666  012706  001100                      MOV     #KERSTK,KSP     ;RESET KERNEL STACK POINTER
3845 031672  000423                             BR      TS726           ;;BRANCH TO NEXT TEXT
```

MM TRAP ROUTINE FOR ABOVE TEST

```
3847                                            .SBTTL  MM TRAP ROUTINE FOR ABOVE TEST
3848 031674  012637  001260          MFPIV3: MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3849 031700  012637  001262                  MOV     (KSP)+,TRAPPS
3850 031704  013737  177572  001264          MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
3851 031712  013737  177576  001270          MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3852 031720  042737  160000  177572          BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3853 031726  104026                          ERROR   +26             ;TRIED TO READ NON-RESIDENT PAGE
3854                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCITON WITH AN 'RTI'' = 000002
3855 031730  013746  001262                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3856 031734  013746  001260                  MOV     TRAPPC,-(KSP)
3857 031740  000002                          RTI                     ;RETURN TO TEST
```

```
3858                                    .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.
                                        ;*
                                        ;;*****************************************************************
    031742  000004                      TST26:  SCOPE
3859 031744  012737  031752  001110            MOV     #1$,$LPERR      ;SET LOOP ON ERROR TO 1$
3860 031752  012737  140340  177776    1$:     MOV     #140340,PSW     ;GO TO USER MODE FOR THIS TEST
3861 031760  012700  036514                    MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
3862 031764  010037  100000                    MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHY 60000
3863 031770  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3864                                    ;THE FOLLOWING WILL TEST DSTM=0 MFPI
3865 031774  012737  032304  000250            MOV     #MFPIV4,MMVEC   ;SET M.M. VECTOR TO MFPIV4
3866 032002  105037  177610                    CLRB    UIPDR4          ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT
3867 032006  012737  140340  177776            MOV     #140340,PSW     ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3868 032014  006506                            MFPI    KSP             ;PUT KERNEL STACK POINTER ON USER STACK
3869 032016  012737  016142  000250            MOV     #MGMERR,MMVEC   ;RESTORE MM VECTOR TO NORMAL ROUTINE
3870 032024  022706  000600                    CMP     #USESTK,USP     ;WAS SOMETHING PUSHED ON STACK AT THE MFPI
3871 032030  001407                            BEQ     2$              ;BRANCH IF NOTHING WAS PUSHED
3872 032032  012600                            MOV     (USP)+,R0       ;POP USER STACK INTO R0
3873 032034  012701  001100                    MOV     #KERSTK,R1      ;EXPECTING 1100 AS KSP
3874 032040  020001                            CMP     R0,R1           ;DID YOU GET THE RIGHT POINTER?
3875 032042  001403                            BEQ     3$              ;BRANCH IF YOU DID
3876 032044  104023                            ERROR   +23             ;WRONG THING WAS PUSHED ON STACK
3877                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 3$' WITH 'BR 1$' = 000763
3878 032046  000401                            BR      3$              ;BRANCH TO NEXT TRY
3879 032050  104025                    2$:     ERROR   +25             ;NOTHING PUSHED ON STACK
3880                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 'BEQ 2$' WITH 'BR 1$' = 000763
3881
3882                                    ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3883 032052  012737  002516  001110    3$:     MOV     #MFPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MFPILP IN SUBROUTINE
3884 032060  012737  140340  002520            MOV     #140340,MFPIPS  ;MAKE PREVIOUS MODE KERNEL PRESENT USER
3885 032066  012737  032304  002622            MOV     #MFPIV4,MFPIVC  ;MOVE THIS TEST'S MM TRAP HANDLER TO MFPIVC
3886 032074  012700  036514                    MOV     #36514,R0       ;LOAD DATA EXPECTED INTO R0
3887 032100  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3888 032104  004737  002464                    JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3889 032110  006512                            MFPI    (R2)            ;<HERE - READ FROM PHYSICAL 60000
3890 032112  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3891 032114  104023                            ERROR   +23             ;WRONG DATA WAS FETCHED
3892 032116  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3893                                    ;THE FOLLOWING WILL TEST DSM=2 MFPI.
3894 032120  012702  100000                    MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3895 032124  004737  002464                    JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3896 032130  006522                            MFPI    (R2)+           ;<HERE - READ FROM PHYSICAL 60000
3897 032132  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3898 032134  104023                            ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
```

```
3899 032136  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3900                              ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
3901 032140  004737  002464                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3902 032144  006537  100000                  MFPI    @#100000        ;<HERE - READ FROM PHYSICAL 60000
3903 032150  104023                          ERROR   +23             ;WRONG DATA WAS FETCHED
3904 032152  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3905                              ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3906 032154  012702  100002                  MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3907 032160  004737  002464                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3908 032164  006542                          MFPI    -(R2)           ;<HERE - READ FROM PHYSICAL 60000
3909 032166  000240                          NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3910 032170  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3911 032172  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3912                              ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
3913 032174  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
3914 032202  012702  001204                  MOV     #<$TMP2+2>,R2   ;LOAD ADDRESS OF $TMP2+2 INTO R2
3915 032206  004737  002464                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3916 032212  006552                          MFPI    @-(R2)          ;<HERE - READ FROM PHYSICAL 60000
3917 032214  000240                          NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3918 032216  104023                          ERROR   +23             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
3919 032220  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3920                              ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
3921 032222  005002                          CLR     R2              ;MAKE REGISTER 2 A ZERO
3922 032224  004737  002464                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3923 032230  006562  100000                  MFPI    100000(R2)      ;<HERE - READ FROM PHYSICAL 60000
3924 032234  104023                          ERROR   +23             ;WRONG DATA WAS FETCHED
3925 032236  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3926                              ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
3927 032240  012737  100000  001202          MOV     #100000,$TMP2   ;LOAD TEST LOC. VIRT. ADDR. INTO $TMP2
3928 032246  012702  001202                  MOV     #$TMP2,R2       ;LOAD ADDRESS OF $TMP2 INTO R2
3929 032252  004737  002464                  JSR     PC,MFPITS       ;GO DO TEST USING MFPI INSTRUCTION LOCATED
3930 032256  006572  000000                  MFPI    @0(R2)          ;<HERE - READ FROM PHYSICAL 60000
3931 032262  104023                          ERROR   +23             ;WRONG DATA WAS FETCHED
3932 032264  005726                          TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3933 032266  012737  000340  177776          MOV     #00340,PSW      ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
3934 032274  112737  000006  177610          MOVB    #6,UIPDR4       ;MAKE UIPDR4 RESIDENT
3935 032302  000423                          BR      TST27           ;;BRANCH TO NEXT TEXT
```

```
;** TRAP HANDLER FOR ABOVE TEST

3937                                            .SBTTL  ;** TRAP HANDLER FOR ABOVE TEST
3938 032304  012637  001260          MFPIV4:    MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
3939 032310  012637  001262                     MOV     (KSP)+,TRAPPS
3940 032314  013737  177572  001264             MOV     SR0,WASSR0      ;SAVE SR0 FOR ERROR TYPEOUT
3941 032322  013737  177576  001270             MOV     SR2,WASSR2      ;SAVE SR2 FOR ERROR TYPEOUT
3942 032330  042737  160000  177572             BIC     #160000,SR0     ;CLEAR ERROR BITS IN SR0
3943 032336  104026                             ERROR   +26             ;TRIED TO READ NON-RESIDENT PAGE
3944                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI' = 000002
3945 032340  013746  001262                     MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
3946 032344  013746  001260                     MOV     TRAPPC,-(KSP)
3947 032350  000002                             RTI                     ;RETURN TO TEST
```

```
3962                                    .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 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 MFPIV5, WHERE THE ERRORS ARE REPORTED.
                                        ;*
                                        ;*****************************************************************
    032352  000004                      TST27:  SCOPE
3963 032354  012700  077400                     MOV     #77400,R0       ;MAKE PAGE 4 IN ALL BUT SUPERVISOR I
3964                                                                    ;AND KERNAL I NON-RESIDENT
3965 032360  010037  172330                     MOV     R0,KDPDR4       ;KERNAL D-SPACE PAGE 4
3966 032364  010037  172230                     MOV     R0,SDPDR4       ;SUPERVISOR D-SPACE PAGE 4
3967 032370  010037  177630                     MOV     R0,UDPDR4       ;USER D-SPACE PAGE 4
3968 032374  010037  177610                     MOV     R0,UIPDR4       ;USER I-SPACE PAGE 4
3969 032400  012700  036514                     MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
3970 032404  010037  060000                     MOV     R0,@#60000      ;LOAD DATA PATTERN INTO PHYS. 60000
3971 032410  052737  000002  172516             BIS     #BIT1,MMR3      ;ENABLE SUPERVISOR D-SPACE
3972 032416  105037  172310                     CLRB    KIPDR4          ;MAKE KERNAL I-SPACE PAGE 4 NON-RESIDENT
3973                                    ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
3974 032422  012737  002516  001110             MOV     #MFPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MFPILP IN SUBROUTINE
3975 032430  012737  010340  002520             MOV     #010340,MFPIPS  ;MAKE PREVIOUS MODE SUPERVISOR IN SUBRTN LOCATION
3976 032436  012737  032562  002622             MOV     #MFPIV5,MFPIVC  ;MOVE THIS TEST'S MM TRAP HANDLER TO MFPIVC
3977 032444  012702  100000                     MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3978 032450  004737  002464                     JSR     PC,MFPITS       ;GO DO TEST USING THE MFPI INSTRUCTION FOUND
3979 032454  006512                            MFPI    (R2)            ;<HERE - READ FROM PHYSICAL 60000
3980 032456  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3981 032460  104037                            ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA FETCHED
3982 032462  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3983                                    ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
3984 032464  012702  100000                     MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3985 032470  004737  002464                     JSR     PC,MFPITS       ;GO DO TEST USING THE MFPI INSTRUCTION FOUND
3986 032474  006522                            MFPI    (R2)+           ;<HERE - READ FROM PHYSICAL 100000
3987 032476  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
3988 032500  104037                            ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA FETCHED
3989 032502  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3990                                    ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
3991 032504  012702  100000                     MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3992 032510  004737  002464                     JSR     PC,MFPITS       ;GO DO TEST USING THE MFPI INSTRUCTION FOUND
3993 032514  006537  100000                     MFPI    @#100000        ;<HERE - READ FROM PHYSICAL 100000
3994 032520  104037                            ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA FETCHED
3995 032522  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
3996                                    ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
3997 032524  012702  100002                     MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
3998 032530  004737  002464                     JSR     PC,MFPITS       ;GO DO TEST USING THE MFPI INSTRUCTION FOUND
3999 032534  006542                            MFPI    -(R2)           ;<HERE - READ FROM PHYSICAL 100000
4000 032536  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
4001 032540  104037                            ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA FETCHED
4002 032542  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
```

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

```
4003 032544  112737  000006  172310          MOVB    #6,KIPDR4       ;MAKE KIPDR4 RESIDENT
4004 032552  042737  000002  172516          BIC     #BIT1,MMR3      ;DISABLE SUPERVISOR D-SPACE
4005 032560  000426                          BR      TST30           ;;BRANCH TO NEXT TEST
```

```
4007                                                    .SBTTL  MM TRAP HANDLER FOR ABOVE TEST
4008 032562  013737  177572  001264  MFPIV5: MOV     MMR0,WASSR0     ;SAVE MMR0 FOR ERROR TYPEOUT
4009 032570  013737  177574  001266          MOV     MMR1,WASSR1     ;SAVE MMR1 FOR ERROR TYPEOUT
4010 032576  013737  177576  001270          MOV     MMR2,WASSR2     ;SAVE MMR2 FOR ERROR TYPEOUT
4011 032604  013737  172516  001272          MOV     MMR3,WASSR3     ;SAVE MMR3 FOR ERROR TYPEOUT
4012 032612  012637  001260                  MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
4013 032616  012637  001262                  MOV     (KSP)+,TRAPPS
4014 032622  104040                          ERROR   +40             ;TRIED TO READ NON-RESIDENT PAGE
4015                            ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI' = 000002
4016 032624  013746  001262                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
4017 032630  013746  001260                  MOV     TRAPPC,-(KSP)
4018 032634  000002                          RTI
```

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

```
4033                                    .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.
                                        ;*
                                        ;*****************************************************************
    032636  000004                      TST30:  SCOPE
4034 032640  052737  000002  172516             BIS     #BIT1,MMR3      ;ENABLE SUPERVISOR D-SPACE
4035                                    ;THIS WILL TEST DSTM = 1 MTPI
4036 032646  012737  002654  001110             MOV     #MTPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MTPILP IN SUBROUTINE
4037 032654  012737  010340  002656             MOV     #010340,MTPIPM  ;MAKE PREVIOUS MODE SUPERVISOR IN LOCATION IN SUBRTN
4038 032662  012737  033022  002744             MOV     #MTPIV3,MTPIVC  ;LOAD THIS TEST'S MM TRAP HANDLER TO MTPIVC
4039 032670  012702  100000                     MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4040 032674  012700  125252                     MOV     #125252,R0      ;LOAD TEST DATA INTO R0
4041 032700  004737  002626                     JSR     PC,MTPITS       ;GO DO THE TEST USING MTPI INSTRUCTION FOUND
4042 032704  006612                            MTPI    (R2)            ;LOAD TEST DATA INTO PHYSICAL 100000
4043 032706  000240                            NOP                     ;<HERE - NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
4044 032710  000000                            .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
4045 032712  104035                            ERROR   +35             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
4046 032714  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4047                                    ;THIS WILL TEST DSTM = 3 MTPI
4048 032716  012700  052525                     MOV     #52525,R0       ;LOAD TEST DATA INTO R0
4049 032722  004737  002626                     JSR     PC,MTPITS       ;GO DO THE TEST USING MTPI INSTRUCTION FOUND
4050 032726  006637  100000                     MTPI    @#100000        ;<HERE - LOAD TEST DATA INTO PHYSICAL 100000
4051 032732  000000                            .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
4052 032734  104035                            ERROR   +35             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
4053 032736  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4054                                    ;THIS WILL TEST DSTM = 4 MTPI
4055 032740  012700  125252                     MOV     #125252,R0      ;LOAD TEST DATA INTO R0
4056 032744  012702  100002                     MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4057 032750  004737  002626                     JSR     PC,MTPITS       ;GO DO THE TEST USING MTPI INSTRUCTION FOUND
4058 032754  006642                            MTPI    -(R2)           ;<HERE - LOAD TEST DATA INTO PHYSICAL 100000
4059 032756  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
4060 032760  000000                            .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
4061 032762  104035                            ERROR   +35             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
4062 032764  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4063                                    ;THIS WILL TEST DSTM = 6 MTPI
4064 032766  012700  052525                     MOV     #52525,R0       ;LOAD TEST DATA INTO R0
4065 032772  005002                            CLR     R2              ;MAKE R2 ZERO
4066 032774  004737  002626                     JSR     PC,MTPITS       ;GO DO THE TEST USING MTPI INSTRUCTION FOUND
4067 033000  006662  100000                     MTPI    100000(R2)      ;<HERE - LOAD TEST DATA INTO PHYSICAL 100000
4068 033004  100000                            .WORD   100000          ;ADD 100000 TO R2 AFTER MTPI INSTRUCTION EXECUTE
4069 033006  104035                            ERROR   +35             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
4070 033010  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4071 033012  042737  000002  172516             BIC     #BIT1,MMR3      ;DISABLE SUPERVISOR D-SPACE
4072 033020  000410                            BR      TST31           ;;BRANCH TO NEXT TEST
```

```
                                .SBTTL  MM TRAP HANDLER FOR ABOVE TEST
033022  013700  177572  MTPIV3: MOV     MMR0,R0         ;SAVE MMR0 FOR ERROR TYPEOUT
033026  013701  177574          MOV     MMR1,R1         ;SAVE MMR1 FOR ERROR TYPEOUT
033032  013702  177576          MOV     MMR2,R2         ;SAVE MMR2 FOR ERROR TYPEOUT
033036  104036                  ERROR   +36             ;TRIED TO LOAD A NON-RESIDENT PAGE 4
                        ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI' = 000002
033040  000002                  RTI                     ;RETURN TO TEST
```

```
4081                                    .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.
                                        ;*
                                        ;;*****************************************************************
    033042  000004                      TST31:  SCOPE
4082 033044  012737  077400  172210             MOV     #77400,SIPDR4   ;MAKE SIPDR4 NON-RESIDENT
4083 033052  012737  077406  177610             MOV     #77406,UIPDR4   ;MAKE UIPDR4 RESIDENT
4084 033060  052737  000001  172516             BIS     #BIT0,MMR3      ;ENABLE USER D-SPACE
4085                                    ;THIS WILL TEST DSTM = 1 MTPI
4086 033066  012737  002654  001110             MOV     #MTPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MTPILP IN SUBROUTINE
4087 033074  012737  030340  002656             MOV     #030340,MTPIPM  ;MAKE LOCATION IN SUBROUTINE PREVIOUS MODE USER
4088 033102  012737  033242  002744             MOV     #MTPIV4,MTPIVC  ;PUT THIS TEST'S MM TRAP HANDLER ADDRESS IN LOC MTPIVC
4089 033110  012702  100000                     MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4090 033114  012700  125252                     MOV     #125252,R0      ;LOAD TEST DATA INTO R0
4091 033120  004737  002626                     JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION FOUND
4092 033124  006612                             MTPI    (R2)            ;<HERE - LOAD TEST DATA INTO PHYSICAL 100000
4093 033126  000240                             NOP                     ;MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
4094 033130  000000                             .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
4095 033132  104035                             ERROR   +35             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
4096 033134  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4097                                    ;THIS WILL TEST DSTM = 3 MTPI
4098 033136  012700  052525                     MOV     #52525,R0       ;LOAD TEST DATA INTO R0
4099 033142  004737  002626                     JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION FOUND
4100 033146  006637  100000                     MTPI    @#100000        ;<HERE - LOAD TEST DATA INTO PHYSICAL 100000
4101 033152  000000                             .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
4102 033154  104035                             ERROR   +35             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
4103 033156  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4104                                    ;THIS WILL TEST DSTM = 4 MTPI
4105 033160  012700  125252                     MOV     #125252,R0      ;LOAD TEST DATA INTO R0
4106 033164  012702  100002                     MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4107 033170  004737  002626                     JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION FOUND
4108 033174  006642                             MTPI    -(R2)           ;LOAD TEST DATA INTO PHYSICAL 100000
4109 033176  000240                             NOP                     ;<HERE - MODE NOT 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
4110 033200  000000                             .WORD   0               ;ADD 0 TO R2 AFTER MTPI INSTRUCTION EXECUTE
4111 033202  104035                             ERROR   +35             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
4112 033204  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4113                                    ;THIS WILL TEST DSTM = 6 MTPI
4114 033206  012700  052525                     MOV     #52525,R0       ;LOAD TEST DATA INTO R0
4115 033212  005002                             CLR     R2              ;MAKE R2 ZERO
4116 033214  004737  002626                     JSR     PC,MTPITS       ;GO DO TEST USING MTPI INSTRUCTION FOUND
4117 033220  006662  100000                     MTPI    100000(R2)      ;<HERE - LOAD TEST DATA INTO PHYSICAL 100000
4118 033224  100000                             .WORD   100000          ;ADD 100000 TO R2 AFTER MTPI INSTRUCTION EXECUTE
4119 033226  104035                             ERROR   +35             ;RETURN IS HERE FOR ERROR - INCORRECT STORE
4120 033230  005726                             TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4121 033232  042737  000001  172516             BIC     #BIT0,MMR3      ;DISABLE USER D-SPACE
```

```
   4122 033240  000410                    BR      TST32          ::BRANCH TO NEXT TEST
```

MM TRAP CATCHER FOR THE ABOVE TEST

```
                                        .SBTTL  MM TRAP CATCHER FOR THE ABOVE TEST
033242  013700  177572          MTPIV4: MOV     MMR0,R0         ;SAVE MMR0 FOR ERROR TYPEOUT
033246  013701  177574                  MOV     MMR1,R1         ;SAVE MMR1 FOR ERROR TYPEOUT
033252  013702  177576                  MOV     MMR2,R2         ;SAVE MMR2 FOR ERROR TYPEOUT
033256  104036                          ERROR   +36             ;TRIED TO LOAD A NON-RESIDENT PAGE 4
                                ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI' = 000002
033260  000002                          RTI                     ;RETURN TO TEST
```

```
 4139                                   .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).
                                        ;*
                                        ;;*****************************************************************
     033262  000004                     TST32:  SCOPE
 4140 033264  112737  000006  172330    1$:     MOVB    #6,KDPDR4       ;MAKE KDPDR4 RESIDENT
 4141 033272  005037  177776            1$:     CLR     @#PSW           ;SET PREVIOUS = CURRENT = KERNEL
 4142 033276  012700  001100                    MOV     #STACK,R0       ;SETUP VALUE FOR STACK POINTER
 4143 033302  010006                            MOV     R0,KSP          ;LOAD STACK POINTER
 4144 033304  006506                            MFPI    KSP             ;THE VALUE "STACK" SHOULD BE PUSHED BEFORE BEING DEC'D
 4145 033306  01   ?1                           MOV     (KSP),R1        ;READ DATA WHICH WAS PUSHED
 4146 033310  020(  1                           CMP     R0,R1           ;WAS THE ORIGINAL VALUE OF THE STACK POINTER PUSHED?
 4147 033312  0014  1                           BEQ     2$              ;BRANCH IF YES
 4148 033314  104037                            ERROR   +37             ;MFPI FETCHED WRONG DATA
 4149                                   ;FOR TIGHTER SCOPE LOOP, REPLACE "BEQ 2$" WITH "BR 1$" = 000767
 4150 033316  005740                    2$:     TST     -(R0)           ;SETUP EXPECTED STACK POINTER VALUE
 4151 033320  020600                            CMP     KSP,R0          ;WAS THE STACK POINTER DECREMENTED?
 4152 033322  001401                            BEQ     3$              ;BRANCH IF YES
 4153 033324  104025                            ERROR   +25             ;STACK NOT PUSHED BY THE MFPI
 4154                                   ;FOR TIGHTER SCOPE LOOP, REPLACE "BEQ 3$" WITH "BR 1$" = 000760
 4155 033326  012706  001100            3$:     MOV     #STACK,KSP      ;RESTORE STACK POINTER
```

```
4169                                    .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.
                                        ;*
                                        ;;*****************************************************************
    033332  000004                      TST33:  SCOPE
4170 033334  012737  000600  172270             MOV     #600,SDPAR4     ;MAP SDPAR4 TO 12K
4171 033342  012700  077400             20$:    MOV     #77400,R0       ;MAKE PAGE 4 IN ALL BUT SUPERVISOR D
4172                                                                    ;AND KERNAL I NON-RESIDENT
4173 033346  010037  172330                     MOV     R0,KDPDR4       ;KERNAL D-SPACE PAGE 4
4174 033352  010037  172210                     MOV     R0,SIPDR4       ;SUPERVISOR I-SPACE PAGE 4
4175 033356  010037  177630                     MOV     R0,UDPDR4       ;USER D-SPACE PAGE 4
4176 033362  010037  177610                     MOV     R0,UIPDR4       ;USER I-SPACE PAGE 4
4177 033366  012737  077406  172230             MOV     #77406,SDPDR4   ;MAKE SDPDR4 RESIDENT
4178 033374  012700  036514                     MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
4179 033400  010037  100000                     MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHYS. 100000
4180 033404  052737  000002  172516             BIS     #BIT1,MMR3      ;ENABLE SUPERVISOR D-SPACE
4181 033412  105037  172310                     CLRB    KIPDR4          ;MAKE KERNAL I-SPACE PAGE 4 NON-RESIDENT
4182                                    ;THE FOLLOWING WILL TEST DSTM=1 MFPD
4183 033416  012737  002770  001110             MOV     #MFPDLP,$LPERR  ;SET LOOP ON ERROR POINTER TO MFPDLP IN SUBROUTINE
4184 033424  012737  010340  002772             MOV     #010340,MFPDPS  ;MOVE PREVIOUS MODE=SUPERVISOR TO LOCATION IN SUBRTN
4185 033432  012737  033564  003040             MOV     #MFPDV1,MFPDVC  ;PUT ADDRESS OF THIS TEST'S MM TRAP CATCHER IN MFPDVC
4186 033440  012702  100000                     MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4187 033444  004737  002746                     JSR     PC,MFPDTS       ;GO DO TEST USING THE MFPD INSTRUCTION LOCATED
4188 033450  106512                            MFPD    (R2)            ;<HERE - READ FROM PHYSICAL 100000
4189 033452  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
4190 033454  104037                            ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
4191 033456  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4192                                    ;THE FOLLOWING WILL TEST DSTM=2 MFPD
4193 033460  012702  100000                     MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4194 033464  004737  002746                     JSR     PC,MFPDTS       ;GO DO TEST USING THE MFPD INSTRUCTION LOCATED
4195 033470  106522                            MFPD    (R2)+           ;<HERE - READ FROM PHYSICAL 100000
4196 033472  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
4197 033474  104037                            ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
4198 033476  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4199                                    ;THE FOLLOWING WILL TEST DSTM=3 MFPD
4200 033500  012702  100000                     MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4201 033504  004737  002746                     JSR     PC,MFPDTS       ;GO DO TEST USING THE MFPD INSTRUCTION LOCATED
4202 033510  106537  100000                     MFPD    @#100000        ;<HERE - READ FROM PHYSICAL 100000
4203 033514  104037                            ERROR   +37             ;WRONG DATA WAS FETCHED
4204 033516  005726                            TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4205                                    ;THE FOLLOWING WILL TEST DSTM=4 MFPD
4206 033520  012702  100002                     MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4207 033524  004737  002746                     JSR     PC,MFPDTS       ;GO DO TEST USING THE MFPD INSTRUCTION LOCATED
4208 033530  106542                            MFPD    -(R2)           ;<HERE - READ FROM PHYSICAL 100000
4209 033532  000240                            NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
4210 033534  104037                            ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
```

```
4211 033534  005726                        TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4212 033540  042737  000002  172516        BIC     #BIT1,MMR3      ;DISABLE SUPERVISOR D-SPACE
4213 033546  012700  077406                MOV     #77406,R0       ;SET UP R0 FOR 4K RESIDENT R/W
4214 033552  010037  172310                MOV     R0,KIPDR4       ;MAKE KIPAR4 RESIDENT
4215 033556  010037  177630                MOV     R0,UDPDR4       ;MAKE UDPDR4 RESIDENT
4216 033562  000423                        BR      TST34           ;;BRANCH TO NEXT TEST
```

```
                                                .SBTTL  MM TRAP HANDLER FOR THE ABOVE TEST
4218
4219 033564  013737  177572  001264  MFPDV1: MOV     MMR0,WASSR0     ;SAVE MMR0 FOR ERROR TYPEOUT
4220 033572  013737  177574  001266          MOV     MMR1,WASSR1     ;SAVE MMR1 FOR ERROR TYPEOUT
4221 033600  013737  177576  001270          MOV     MMR2,WASSR2     ;SAVE MMR2 FOR ERROR TYPEOUT
4222 033606  012637  001260                  MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
4223 033612  012637  001262                  MOV     (KSP)+,TRAPPS
4224 033616  104040                          ERROR   +40             ;TRIED TO READ NON-RESIDNT PAGE
4225                                    ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI' = 000002
4226 033620  013746  001262                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
4227 033624  013746  001260                  MOV     TRAPPC,-(KSP)
4228 033630  000002                          RTI                     ;RETURN TO TEST
```

```
4240                                    .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.
                                        ;*
                                        ;;*****************************************************************
    033632  000004                      TST34:  SCOPE
4241 033634  012737  000600  177670             MOV     #600,UDPAR4     ;MAP UDPAR4 TO 12K
4242 033642  012700  036514             MOV     #36514,R0       ;LOAD DATA PATTERN INTO R0
4243 033646  010037  100000             MOV     R0,@#100000     ;LOAD DATA PATTERN INTO PHYS. 100000
4244 033652  052737  000001  172516     BIS     #BIT0,MMR3      ;ENABLE USER D-SPACE
4245 033660  105037  172230             CLRB    SDPDR4          ;MAKE SDPDR4 NON-RESIDENT
4246 033664  105037  172310             CLRB    KIPDR4          ;MAKE KERNAL I-SPACE PAGE 4 NON-RESIDENT
4247                                    ;THE FOLLOWING WILL TEST DSTM=1 MFPI
4248 033670  012737  002516  001110     MOV     #MFPILP,$LPERR  ;SET LOOP ON ERROR POINTER TO MFPILP IN SUBROUTINE
4249 033676  012737  170340  002520     MOV     #170340,MFPIPS  ;SET PREVIOUS AND CURRENT MODE TO LOCATION IN SUBRTN
4250 033704  012737  034036  002622     MOV     #MFPIV6,MFPIVC  ;PUT ADDRESS OF THIS TEST'S TRAP CATCHER IN MFPIVC
4251 033712  012702  100000             MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4252 033716  004737  002464             JSR     PC,MFPITS       ;GO DO TEST USING THE MFPI INSTRUCTION LOCATED
4253 033722  006512                     MFPI    (R2)            ;<HERE - READ FROM PHYSICAL 100000
4254 033724  000240                     NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
4255 033726  104037                     ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
4256 033730  005726                     TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4257                                    ;THE FOLLOWING WILL TEST DSTM=2 MFPI
4258 033732  012702  100000             MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4259 033736  004737  002464             JSR     PC,MFPITS       ;GO DO TEST USING THE MFPI INSTRUCTION LOCATED
4260 033742  006522                     MFPI    (R2)+           ;<HERE - READ FROM PHYSICAL 100000
4261 033744  000240                     NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
4262 033746  104037                     ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
4263 033750  005726                     TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4264                                    ;THE FOLLOWING WILL TEST DSTM=3 MFPI
4265 033752  012702  100000             MOV     #100000,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4266 033756  004737  002464             JSR     PC,MFPITS       ;GO DO TEST USING THE MFPI INSTRUCTION LOCATED
4267 033762  006537  100000             MFPI    @#100000        ;<HERE - READ FROM PHYSICAL 100000
4268 033766  104037                     ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
4269 033770  005726                     TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4270                                    ;THE FOLLOWING WILL TEST DSTM=4 MFPI
4271 033772  012702  100002             MOV     #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
4272 033776  004737  002464             JSR     PC,MFPITS       ;GO DO TEST USING THE MFPI INSTRUCTION LOCATED
4273 034002  006542                     MFPI    -(R2)           ;<HERE - READ FROM PHYSICAL 100000
4274 034004  000240                     NOP                     ;NOT MODE 3, 6 OR 7 - NEEDED FOR SUBROUTINE LOAD
4275 034006  104037                     ERROR   +37             ;RETURN IS HERE FOR ERROR - WRONG DATA WAS FETCHED
4276 034010  005726                     TST     (SP)+           ;POP EXCESS RETURN OFF STACK - LOOPING NOT DONE
4277 034012  042737  000001  172516     BIC     #BIT0,MMR3      ;DISABLE USER D-SPACE
4278 034020  012737  000340  177776     MOV     #340,PSW        ;MAKE PRESENT MODE KERNAL
4279 034026  112737  000006  172310     MOVB    #6,KIPDR4       ;MAKE KIPDR4 RESIDENT
4280 034034  000423                     BR      TST35           ;;BRANCH TO NEXT TEST
```

```
                                                .SBTTL  MM TRAP CATCHER FOR ABOVE TEST
4283 034036  013737  177572  001264  MFPIV6: MOV     MMR0,WASSR0     ;SAVE MMR0 FOR ERROR TYPEOUT
4284 034044  013737  177574  001266          MOV     MMR1,WASSR1     ;SAVE MMR1 FOR ERROR TYPEOUT
4285 034052  013737  177576  001270          MOV     MMR2,WASSR2     ;SAVE MMR2 FOR ERROR TYPEOUT
4286 034060  012637  001260                  MOV     (KSP)+,TRAPPC   ;SAVE PC & PS OF TRAP
4287 034064  012637  001262                  MOV     (KSP)+,TRAPPS
4288 034070  104040                          ERROR   +40             ;TRIED TO READ NON-RESIDENT PAGE
4289                                 ;FOR TIGHTER SCOPE LOOP, REPLACE 1ST MOV INSTRUCTION WITH AN 'RTI' = 000002
4290 034072  013746  001262                  MOV     TRAPPS,-(KSP)   ;PUT PC & PS OF TRAP ON STACK
4291 034076  013746  001260                  MOV     TRAPPC,-(KSP)
4292 034102  000002                          RTI                     ;RETURN TO TEST
```

```
 4300                                   .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.
                                        ;;*****************************************************************
                                        ;;*****************************************************************
     034104  000004                     TST35:  SCOPE
 4301 034106  005037  172516                    CLR     MMR3            ;MAKE SURE MMR3 IS CLEARED, DISABLING CSM
 4302 034112  005037  177776                    CLR     PSW             ;MAKE SURE PSW PREVIOUS=CURRENT=KERNEL
 4303 034116  013737  000010  001204           MOV     RESVEC,$TMP3    ;SAVE TRAPS TO 10 VECTOR
 4304 034124  012737  034140  000010           MOV     #1$,RESVEC      ;TRAPS TO 10 GO TO 1$
 4305 034132  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
 4306 034136  007000                           CSM0                    ;CSM    R0
 4307 034140  013737  001204  000010  1$:      MOV     $TMP3,RESVEC    ;RESTORE TRAPS TO 10 VECTOR
 4308 034146  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
 4309 034152  005037  177776                    CLR     PSW             ;RETURN TO KERNEL MODE
 4310 034156  022726  034140                    CMP     #1$,(SP)+       ;SEE IF IT WAS AN EXPECTED TRAP
 4311 034162  001403                           BEQ     2$              ;BRANCH TO RESTORE STACK IF IT WAS
 4312 034164  022626                           CMP     (SP)+,(SP)+     ;CORRECT STACK
 4313 034166  104041                           ERROR   +41             ;ILLEGAL CSM DID NOT TRAP TO 10
 4314 034170  000401                           BR      TST36           ;;BRANCH AROUND STACK CORRECTION
 4315 034172  005726                    2$:     TST     (SP)+           ;CORRECT STACK
```

```
 4323                                   .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.
                                        ;;*****************************************************************
                                        ;;*****************************************************************
     034174  000004                     TST36:  SCOPE
 4324 034176  005037  172516                    CLR     MMR3            ;MAKE SURE MMR3 IS CLEARED, DISABLING CSM
 4325 034202  052737  040000  177776            BIS     #40000,PSW      ;GO TO SUPERVISOR MODE
 4326 034210  013737  000010  001204            MOV     RESVEC,$TMP3    ;SAVE TRAPS TO 10 VECTOR
 4327 034216  012737  034232  000010            MOV     #1$,RESVEC      ;TRAPS TO 10 GO TO 1$
 4328 034224  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
 4329 034230  007000                            CSM0                    ;CSM    R0
 4330 034232  013737  001204  000010  1$:     MOV     $TMP3,RESVEC    ;RESTORE TRAPS TO 10 VECTOR
 4331 034240  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
 4332 034244  005037  177776                    CLR     PSW             ;GO BACK TO KERNEL MODE
 4333 034250  022726  034232                    CMP     #1$,(SP)+       ;SEE IF IT WAS AN EXPECTED TRAP
 4334 034254  001402                            BEQ     2$              ;BRANCH AROUND ERROR CALL IF IT WAS
 4335 034256  022626                            CMP     (SP)+,(SP)+     ;CORRECT STACK
 4336 034260  104041                            ERROR   +41             ;ILLEGAL CSM DID NOT TRAP TO 10
 4337 034262  000401                            BR      TST37           ;;BRANCH AROUND STACK CORRECTION
 4338 034264  005726                    2$:     TST     (SP)+           ;CORRECT STACK
```

```
 4346                                   .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.
                                        ;;*****************************************************************
                                        ;;*****************************************************************
     034266  000004                     TST37:  SCOPE
 4347 034270  005037  172516                    CLR     MMR3            ;MAKE SURE MMR3 IS CLEARED, DISABLING CSM
 4348 034274  052737  140000  177776            BIS     #140000,PSW     ;GO TO USER MODE
 4349 034302  013737  000010  001204            MOV     RESVEC,$TMP3    ;SAVE TRAPS TO 10 VECTOR
 4350 034310  012737  034324  000010            MOV     #1$,RESVEC      ;TRAPS TO 10 GO TO 1$
 4351 034316  005237  177572                    INC     MMR0            ;TURN ON MEMORY MANAGEMENT
 4352 034322  007000                           CSM0                    ;CSM    R0
 4353 034324  013737  001204  000010  1$:     MOV     $TMP3,RESVEC    ;RESTORE TRAPS TO 10 VECTOR
 4354 034332  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
 4355 034336  005037  177776                    CLR     PSW             ;RETURN TO KERNEL MODE
 4356 034342  022726  034324                    CMP     #1$,(SP)+       ;SEE IF IT WAS AN EXPECTED TRAP
 4357 034346  001403                           BEQ     2$              ;BRANCH AROUND ERROR CALL IF IT WAS
 4358 034350  022626                           CMP     (SP)+,(SP)+     ;CORRECT STACK
 4359 034352  104041                           ERROR   +41             ;ILLEGAL CSM DID NOT TRAP TO 10
 4360 034354  000401                           BR      TST40           ;;BRANCH AROUND STACK CORRECTION
 4361 034356  005726                    2$:     TST     (SP)+           ;CORRECT STACK
```

```
4369                                    .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.
                                        ;;**********************************************************************
                                        ;;**********************************************************************
     034360  000004                     TST40:  SCOPE
4370 034362  052737  000010  172516             BIS     #BIT3,MMR3      ;TURN ON CSM ENABLE
4371 034370  005037  177776                     CLR     PSW             ;MAKE SURE PSW PREVIOUS=CURRENT=KERNEL
4372 034374  013737  000010  001204             MOV     RESVEC,$TMP3    ;SAVE TRAPS TO 10 VECTOR
4373 034402  012737  034416  000010             MOV     #1$,RESVEC      ;TRAPS TO 10 GO TO 1$
4374 034410  005237  177572                     INC     MMR0            ;TURN ON MEMORY MANAGEMENT
4375 034414  007000                             CSM0                    ;CSM    R0
4376 034416  013737  001204  000010  1$:      MOV     $TMP3,RESVEC    ;RESTORE TRAPS TO 10 VECTOR
4377 034424  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4378 034430  022726  034416                     CMP     #1$,(SP)+       ;SEE IF IT WAS AN EXPECTED TRAP
4379 034434  001403                             BEQ     2$              ;BRANCH AROUND ERROR CALL IF IT WAS
4380 034436  022626                             CMP     (SP)+,(SP)+     ;CORRECT STACK
4381 034440  104041                             ERROR   +41             ;ILLEGAL CSM DID NOT TRAP TO 10
4382 034442  000401                             BR      TST41           ;;BRANCH AROUND STACK CORRECTION
4383 034444  005726                     2$:     TST     (SP)+           ;CORRECT STACK
```

```
4392                                    .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.
                                        ;;*****************************************************************
                                        ;;*****************************************************************
     034446  000004                     TST41:  SCOPE
4393 034450  012700  052525                     MOV     #52525,R0       ;SET UP R0
4394 034454  004737  002060                     JSR     PC,APRINIT      ;RESET ALL MEMORY MANAGEMENT REGISTERS
4395 034460  012737  000600  172260           MOV     #600,SDPAR0     ;IF WE GO TO D-SPACE, FLAG AS AN ERROR
4396 034466  012737  000032  172516           MOV     #32,MMR3        ;ENABLE CSM, 22-BIT, SUPERVISOR SPACE
4397 034474  012737  040000  035320           MOV     #40000,PCSMPS   ;SET PRECSM PS IN LOCATION
4398 034502  012706  000700                     MOV     #SUPSTK,SSP     ;SETUP SUPERVISOR STACK POINTER
4399 034506  004737  034530                     JSR     PC,CSMSUB       ;TEST CSM INSTRUCTION SUBROUTINE
4400 034512  000000                           .WORD   0               ; 0 ADDED TO ADDRESS OF TRAP VECTOR
4401 034514  005037  177776                     CLR     PSW             ;RETURN TO KERNEL MODE
4402 034520  005037  172260                     CLR     SDPAR0          ;RESET SDPAR0 TO FIRST 4K
4403 034524  000137  035344                     JMP     TST42           ;JUMP OVER THE CSM SUBROUTINE
```

```
4404                                            .SBTTL  SUBROUTINE TO CHECK OPERATION OF AN EXECUTABLE CSM INST'N
4405 034530  012605                     CSMSUB: MOV     (SP)+,R5        ;PUT RETURN ADDRESS INTO R5
4406 034532  012737  000340  177776             MOV     #340,PSW        ;KERNEL MODE
4407 034540  010037  035324                     MOV     R0,PCSMR0       ;STORE PRECSM R0
4408 034544  012504                             MOV     (R5)+,R4        ;MOVE OFFSET TO R4
4409 034546  010537  035342                     MOV     R5,RETURN       ;SET RETURN LOCATION ADDRESS TO LOCATION 'RETURN'
4410 034552  012737  034704  035316             MOV     #5$,DOWELP      ;MOVE ADDRESS FOR INITIAL TESTING TO DOWELP
4411 034560  012764  034622  000010             MOV     #2$,RESVEC(R4)  ;SETUP TRAPS TO 10 VECTOR TO 2$
4412 034566  012737  034614  001110             MOV     #1$,$LPERR      ;SETUP LOOP ON ERROR TO 1$
4413 034574  053737  035320  177776             BIS     PCSMPS,PSW      ;PUT USER/SUPERVISOR IN THE PSW
4414 034602  010637  035322                     MOV     SP,PCSMSP       ;MOVE STACK POINTER VALUE TO PCSMSP
4415 034606  162737  000006  035322             SUB     #6,PCSMSP       ;SUBTRACT 6 FROM PCSMSP - EXPECT 3 PUSHES ON STACK
4416 034614  005237  177572          1$:        INC     MMR0            ;TURN ON MEMORY MANAGEMENT
4417 034620  007000                             CSM0
4418 034622  012737  000012  000010  2$:        MOV     #RESVEC+2,RESVEC;RESTORE TRAPS TO 10 VECTOR
4419 034630  022716  034622                     CMP     #2$,(SP)        ;SEE IF CSM ABORTED
4420 034634  001011                             BNE     4$              ;BRANCH IF IT DIDN'T TO EXECUTE TEST
4421 034636  012737  000340  177776  3$:        MOV     #340,PSW        ;GO BACK TO KERNEL PRIORITY 7
4422 034644  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4423 034650  022626                             CMP     (SP)+,(SP)+     ;CORRECT STACK
4424 034652  104051                             ERROR   +51             ;CSM ABORTED WHEN IT SHOULD NOT HAVE
4425 034654  000177  000462                     JMP     @RETURN         ;JUMP TO ADDRESS PRESTORED IN LOCATION RETURN
4426 034660  010637  035336          4$:        MOV     SSP,ACSMSP      ;SAVE AFTER CSM STACK POINTER
4427 034664  012637  035326                     MOV     (SP)+,CSM1ST    ;POP 1ST WORD OFF STACK
4428 034670  012637  035330                     MOV     (SP)+,CSM2ND    ;POP 2ND WORD OFF STACK
4429 034674  012637  035332                     MOV     (SP)+,CSM3RD    ;POP 3RD WORD OFF STACK
4430 034700  000177  000412                     JMP     @DOWELP         ;JUMP TO LOCATION IN DOWELP
4431 034704  013703  035320          5$:        MOV     PCSMPS,R3       ;PUT PRE-CSM PSW IN R3
4432 034710  042703  037777                     BIC     #37777,R3       ;WIPE OUT ALL BITS EXCEPT <15:14>,
4433 034714  000241                             CLC                     ;CLEAR THE CARRY BIT
4434 034716  006003                             ROR     R3              ;MOVE <15:14> OVER TO
4435 034720  006003                             ROR     R3              ;THE RIGHT TO <13:12>,
4436 034722  052703  040000                     BIS     #BIT14,R3       ;AND SET <14>, PUTTING CURRENT=SUPERVISOR IN EXPECTED LOC
4437 034726  013737  177776  035334             MOV     PSW,ACSMPS      ;MOVE CURRENT PSW TO ACSMPS
4438 034734  042737  007777  035334             BIC     #7777,ACSMPS    ;WIPE OUT ALL BITS BUT <15:12>
4439 034742  020337  035334                     CMP     R3,ACSMPS       ;SEE IF PSW STATUS BITS MATCHES CALCULATED VALUE
4440 034746  001403                             BEQ     6$              ;BRANCH AROUND ERROR CALL IF OK
4441 034750  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4442 034754  104042                             ERROR   +42             ;NOT SUPERVISOR MODE
4443 034756  052737  000001  177572  6$:        BIS     #BIT00,MMR0     ;MAKE SURE MEMORY MANAGEMENT IS ON
4444 034764  013737  177776  035334             MOV     PSW,ACSMPS      ;MOVE CURRENT PSW TO ACSMPS
4445 034772  042737  037777  035334             BIC     #37777,ACSMPS   ;CLEAR ALL BITS EXCEPT <15:14>
4446 035000  023737  035340  035334             CMP     SUPERM,ACSMPS   ;SEE IF SUPERVISOR MODE
4447 035006  001414                             BEQ     8$              ;BRANCH AROUND ERROR CALL IF OK
4448 035010  012737  034756  035316             MOV     #6$,DOWELP      ;SET ADDRESS FOR THIS CHECK TO DOWELP
4449 035016  005037  177572                     CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4450 035022  013703  035334                     MOV     ACSMPS,R3       ;MOVE RECEIVED CONTENTS TO R3
4451 035026  052703  040000                     BIS     #BIT14,R3       ;SET BIT 14 AND
4452 035032  042703  100000                     BIC     #BIT15,R3       ;CLEAR BIT 15
4453 035036  104042                             ERROR   +42             ;NOT SUPERVISOR MODE
4454 035040  052737  000001  177572  8$:        BIS     #BIT00,MMR0     ;MAKE SURE MEMORY MANAGEMENT IS ON
4455 035046  013704  177776                     MOV     PSW,R4          ;MOVE CURRENT PSW TO R4
4456 035052  042704  147777                     BIC     #147777,R4      ;CLEAR ALL BITS EXCEPT <13:12>
4457 035056  006304                             ASL     R4              ;MOVE <13:12> BITS IN R4
4458 035060  006304                             ASL     R4              ;TO THE <15:14> POSITION
4459 035062  013702  035320                     MOV     PCSMPS,R2       ;MOVE PRECSM PSW TO R2
4460 035066  042702  037777                     BIC     #37777,R2       ;CLEAR BITS 13 TO 0
```

```
4461 035072  020204                          CMP     R2,R4           ;SEE IF PREVIOUS MODE IS OK
4462 035074  001406                          BEQ     10$             ;BRANCH AROUND ERROR CALL IF OK
4463 035076  012737  035040  035316          MOV     #8$,DOWELP      ;SET ADDRESS FOR THIS CHECK TO DOWELP
4464 035104  005037  177572                  CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4465 035110  104043                          ERROR   +43             ;WRONG PREVIOUS MODE
4466 035112  052737  000001  177572  10$:    BIS     #BIT00,MMR0     ;MAKE SURE MEMORY MANAGEMENT IS ON
4467 035120  023737  035336  035322          CMP     ACSMSP,PCSMSP   ;SEE IF STACK POINTER VALUE WAS TRANSFERED
4468 035126  001406                          BEQ     12$             ;BRANCH AROUND ERROR CALL IF SO
4469 035130  012737  035112  035316          MOV     #10$,DOWELP     ;SET ADDRESS FOR THIS CHECK TO DOWELP
4470 035136  005037  177572                  CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4471 035142  104044                          ERROR   +44             ;INCORRECT STACK
4472 035144  052737  000001  177572  12$:    BIS     #BIT00,MMR0     ;MAKE SURE MEMORY MANAGEMENT IS ON
4473 035152  020037  035326                  CMP     R0,CSM1ST       ;COMPARE R0 WITH THE ARGUMENT THAT WAS ON STACK
4474 035156  001406                          BEQ     14$             ;IF EQUAL, BRANCH AROUND ERROR
4475 035160  012737  035144  035316          MOV     #12$,DOWELP     ;SET ADDRESS FOR THIS CHECK TO DOWELP
4476 035166  005037  177572                  CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4477 035172  104045                          ERROR   +45             ;INCORRECT ARGUMENT
4478 035174  052737  000001  177572  14$:    BIS     #BIT00,MMR0     ;MAKE SURE MEMORY MANAGEMENT IS ON
4479 035202  022737  034622  035330          CMP     #2$,CSM2ND      ;SEE IF UPDATED PC WAS CORRECT
4480 035210  001411                          BEQ     16$             ;BRANCH AROUND ERROR IF OK
4481 035212  012737  035174  035316          MOV     #14$,DOWELP     ;SET ADDRESS FOR THIS CHECK TO DOWELP
4482 035220  005037  177572                  CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4483 035224  012737  034622  001176          MOV     #2$,$TMP0       ;MOVE EXPECTED UPDATED PC TO $TMP0
4484 035232  104046                          ERROR   +46             ;WRONG PC
4485 035234  052737  000001  177572  16$:    BIS     #BIT00,MMR0     ;MAKE SURE MEMORY MANAGEMENT IS ON
4486 035242  032737  000017  035332          BIT     #17,CSM3RD      ;SEE IF PSW <3:0> WERE CLEARED
4487 035250  001406                          BEQ     18$             ;BRANCH AROUND ERROR CALL IF OK
4488 035252  012737  035234  035316          MOV     #16$,DOWELP     ;SET ADDRESS FOR THIS CHECK TO DOWELP
4489 035260  005037  177572                  CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
4490 035264  104047                          ERROR   +47             ;BITS <3:0> SET IN PSW
4491 035266  022737  034704  035316  18$:    CMP     #5$,DOWELP      ;SEE IF ANY ERRORS THIS TIME AROUND
4492 035274  001406                          BEQ     19$             ;BRANCH TO NORMAL RETURN JUMP IF NOT
4493 035276  032777  001000  143634          BIT     #BIT09,@SWR     ;SEE IF LOOP ON ERROR IS SET
4494 035304  001402                          BEQ     19$             ;BRANCH IF SO
4495 035306  000137  034614                  JMP     1$              ;JUMP BACK FOR LOOP ON ERROR
4496 035312  000177  000024          19$:    JMP     @RETURN         ;JUMP BACK TO RETURN ADDRESS IN LOCATION RETURN
4497
4498 035316  000000                  DOWELP: .WORD   0               ;LOCATION HOLDING ADDRESS TO SHORTEN POSSIBLE ERROR LOOP
4499 035320  000000                  PCSMPS: .WORD   0               ;LOCATION TO STORE PRE-CSM PSW
4500 035322  000000                  PCSMSP: .WORD   0               ;LOCATION TO STORE PRE-CSM STACK POINTER
4501 035324  000000                  PCSMR0: .WORD   0               ;LOCATION TO STORE PRE-CSM R0 VALUE
4502 035326  000000                  CSM1ST: .WORD   0               ;LOCATION TO STORE 1ST WORD POPPED AFTER CSM EXECUTION
4503 035330  000000                  CSM2ND: .WORD   0               ;LOCATION TO STORE 2ND WORD POPPED AFTER CSM EXECUTION
4504 035332  000000                  CSM3RD: .WORD   0               ;LOCATION TO STORE 3RD WORD POPPED AFTER CSM EXECUTION
4505 035334  000000                  ACSMPS: .WORD   0               ;LOCATION TO STORE POST-CSM PSW
4506 035336  000000                  ACSMSP: .WORD   0               ;LOCATION TO STORE POST-CSM STACK POINTER
4507 035340  040000                  SUPERM: .WORD   40000           ;LOCATION TO STORE VALUE OF PRESENT MODE=SUPERVISOR
4508 035342  000000                  RETURN: .WORD   0               ;LOCATION TO STORE RETURN ADDRESS OF THE CSMSUB
```

```
4517                                    .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.
                                        ;;*****************************************************************
                                        ;;*****************************************************************
     035344  000004                     TST42:  SCOPE
4518 035346  012737  000032  172516             MOV     #32,MMR3        ;ENABLE 22-BIT, CSM, SUPERVISOR
4519 035354  012737  040000  035320             MOV     #40000,PCSMPS   ;LOAD PRECSM LOCATION
4520 035362  052737  040000  177776             BIS     #BIT14,PSW      ;GO TO SUPERVISOR MODE
4521 035370  012737  000600  172240             MOV     #600,SIPAR0     ;IF WE GO TO I-SPACE, FLAG IT AS AN ERROR
4522 035376  012706  000700                     MOV     #SUPSTK,SSP     ;SETUP SUPERVISOR STACK POINTER
4523 035402  004737  034530                     JSR     PC,CSMSUB       ;TEST CSM INSTRUCTION SUBROUTINE
4524 035406  060000                             .WORD   60000           ; 60000 ADDED TO ADDRESS OF TRAP VECTOR
4525 035410  005037  177776                     CLR     PSW             ;RETURN TO KERNEL MODE
4526 035414  005037  172240                     CLR     SIPAR0          ;RESET SIPAR0 TO FIRST 4K
```

```
 4535                                   .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.
                                        ;;*****************************************************************
                                        ;;*****************************************************************
     035420  000004                     TST43:  SCOPE
 4536 035422  012737  000031  172516            MOV     #31,MMR3        ;ENABLE 22-BIT, CSM USER
 4537 035430  012737  140000  035320            MOV     #140000,PCSMPS  ;LOAD PRECSM LOCATION
 4538 035436  052737  140000  177776            BIS     #140000,PSW     ;GO TO USER MODE
 4539 035444  012737  000600  177660            MOV     #600,UDPAR0     ;IF WE GO TO D-SPACE, FLAG AS AN ERROR
 4540 035452  012706  000600                    MOV     #USESTK,SSP     ;SETUP USER STACK POINTER
 4541 035456  004737  034530                    JSR     PC,CSMSUB       ;TEST CSM INSTRUCTION SUBROUTINE
 4542 035462  000000                            .WORD   0               ; 0 ADDED TO ADDRESS OF TRAP VECTOR
 4543 035464  005037  177776                    CLR     PSW             ;RETURN TO KERNEL MODE
 4544 035470  005037  177660                    CLR     UDPAR0          ;RESET UDPAR0 TO FIRST 4K
```

```
4553                                    .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.
                                        ;;*****************************************************************
                                        ;;*****************************************************************
     035474  000004                     TST44:  SCOPE
4554 035476  012737  000031  172516             MOV     #31,MMR3        ;ENABLE 22-BIT, CSM USER
4555 035504  012737  140000  035320             MOV     #140000,PCSMPS  ;LOAD PRECSM LOCATION
4556 035512  012737  000600  177640             MOV     #600,UIPAR0     ;IF WE GO TO I-SPACE, FLAG AS AN ERROR
4557 035520  012706  000600                     MOV     #USESTK,SSP     ;SETUP USER STACK POINTER
4558 035524  004737  034530                     JSR     PC,CSMSUB       ;TEST CSM INSTRUCTION SUBROUTINE
4559 035530  000000                     .WORD   0               ; 0 ADDED TO ADDRESS OF TRAP VECTOR
4560 035532  005037  177776                     CLR     PSW             ;RETURN TO KERNEL MODE
4561 035536  005037  177640                     CLR     UIPAR0          ;RESET UIPAR0 TO FIRST 4K
```

```
 4567                                   .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
                                        ;;*****************************************************************
     035542  000004                     TST45:  SCOPE
 4568 035544  012737  040000  177776            MOV     #40000,PSW      ;GO TO SUPERVISOR MODE
 4569 035552  012737  000032  172516            MOV     #32,MMR3        ;ENABLE 22-BIT, CSM, SUPERVISOR
 4570 035560  012701  000007                    MOV     #7,R1           ;DO 7 MODES
 4571 035564  012702  035744                    MOV     #CSMINS,R2      ;LOAD CSM INSTRUCTION POINTER INTO R2
 4572 035570  012703  035770                    MOV     #REGCHG,R3      ;LOAD REGISTER CHANGE TABLE POINTER TO R3
 4573 035574  012704  036006                    MOV     #REGDAT,R4      ;LOAD ARGUMENT EXPECTED STACK POINTER TO R4
 4574 035600  012737  035622  000010            MOV     #3$,RESVEC      ;SET RETURNS TO 3$
 4575 035606  011237  035616            1$:     MOV     (R2),2$         ;MOVE CSM INSTRUCTION TO TEST LOCATION
 4576 035612  012700  035764                    MOV     #REGLAB,R0      ;RESET R0
 4577 035616  000000  000000            2$:     .WORD   0,0             ;1ST WORD FOR CSM INSTRUCTION, 2ND FOR MODES 6 & 7
 4578 035622  011637  001200            3$:     MOV     (SP),$TMP1      ;MOVE ARGUMENT WORD TO $TMP1
 4579 035626  021300                            CMP     (R3),R0         ;SEE IF REGISTER CHANGED PROPERLY
 4580 035630  001407                            BEQ     4$              ;BRANCH IF OK
 4581 035632  062706  000006                    ADD     #6,SP           ;POP STACK FOR POSSIBLE ERROR LOOPING
 4582 035636  011337  001176                    MOV     (R3),$TMP0      ;MOVE EXPECTED DATA TO $TMP0
 4583 035642  104052                            ERROR   +52             ;CSM FAILED TO INCREMENT/DECREMENT REGISTER PROPERLY
 4584 035644  162706  000006                    SUB     #6,SP           ;RESTORE STACK POINTER TO PRE-ERROR STATE
 4585 035650  021437  001200            4$:     CMP     (R4),$TMP1      ;SEE IF CORRECT ARGUMENT WAS LOADED
 4586 035654  001417                            BEQ     5$              ;BRANCH IF OK
 4587 035656  012637  001204                    MOV     (SP)+,$TMP3     ;POP STACK FOR POSSIBLE ERROR LOOPING
 4588 035662  012637  001206                    MOV     (SP)+,$TMP4
 4589 035666  012637  001210                    MOV     (SP)+,$TMP5
 4590 035672  011437  001202                    MOV     (R4),$TMP2      ;MOVE EXPECTED DATA TO $TMP2
 4591 035676  104053                            ERROR   +53             ;CSM FAILED TO PUT PROPER ARGUMENT ON STACK
 4592 035700  013746  001210                    MOV     $TMP5,-(SP)     ;RESTORE STACK - NO ERROR LOOPING
 4593 035704  013746  001206                    MOV     $TMP4,-(SP)
 4594 035710  013746  001204                    MOV     $TMP3,-(SP)
 4595 035714  022223                    5$:     CMP     (R2)+,(R3)+     ;INCREMENT R2 AND R3 TO NEXT LOCATIONS
 4596 035716  005724                            TST     (R4)+           ;INCREMENT R4 TO NEXT LOCATION
 4597 035720  062706  000006                    ADD     #6,SP           ;POP STACK OF CSM PUSHES
 4598 035724  077150                            SOB     R1,1$           ;SUBTRACT 1 AND BRANCH IF NOT ALL MODES CHECKED
 4599 035726  005037  177572                    CLR     MMR0            ;TURN OFF MEMORY MANAGEMENT
 4600 035732  005037  172516                    CLR     MMR3            ;CLEAR MMR3
 4601 035736  005037  177776                    CLR     PSW             ;RETURN MODE BACK TO KERNEL
 4602 035742  000430                            BR      $EOP            ;SKIP OVER TABLES AND LOCATIONS
 4603 035744  007010  007020  007030    CSMINS: .WORD   CSM1,CSM2,CSM3,CSM4,CSM5,CSM6,CSM7
 4604 035762  035766                            .WORD   REGLAB+2
 4605 035764  035762                    REGLAB: .WORD   REGLAB-2
 4606 035766  052525                            .WORD   52525
 4607 035770  035764  035766  035766    REGCHG: .WORD   REGLAB,REGLAB+2,REGLAB+2,REGLAB-2,REGLAB-2,REGLAB,REGLAB
 4608 036006  035762  035762  035766    REGDAT: .WORD   REGLAB-2,REGLAB-2,REGLAB+2,REGLAB+2,52525,REGLAB-2,REGLAB+2
```

```
460;                                    .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

036024                                  $EOP:
036024  000004                                 SCOPE
036026  005037  001102                         CLR     $TSTNM          ;;ZERO THE TEST NUMBER
036032  005237  001232                         INC     $PASS           ;;INCREMENT THE PASS NUMBER
036036  042737  100000  001232                 BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
036044  005327                                 DEC     (PC)+           ;;LOOP?
036046  000001                          $EOPCT: .WORD   1
036050  003104                                 BGT     $DOAGN          ;;YES
036052  012737                                 MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
036054  000001                          $ENDCT: .WORD   1
036056  036046                                 $EOPCT
036060  032737  000077  001232                 BIT     #77,$PASS       ;>>ARE WE TO PRINT THIS PASS?           ;DPM002
036066  001403                                 BEQ     1000$           ;>>BRANCH IF WE ARE                     ;DPM002
036070  005737  001112                         TST     $ERTTL          ;>>TEST $ERTTL FOR ANY ERRORS THIS PASS ;DPM002
036074  001453                                 BEQ     $GET42          ;>>BRANCH IF NO ERRORS THIS PASS        ;DPM002
036076                                  1000$:
036076  104401  036104                         TYPE    ,65$            ;;TYPE ASCIZ STRING
036102  000407                                 BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  <12><15>/END PASS #/
036122                                  64$:
036122  013746  001232                         MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                                                        ;;TYPE PASS NUMBER
036126  104405                                 TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
036130  005737  001112                         TST     $ERTTL          ;>>ANY ERRORS THIS PASS?                ;DPM002
036134  001433                                 BEQ     $GET42          ;>>BRANCH OVER 'TOTAL' REPORT IF NONE   ;DPM002
036136  104401  036144                         TYPE    ,67$            ;;TYPE ASCIZ STRING
036142  000421                                 BR      66$             ;;GET OVER THE ASCIZ
                                        ;;67$:  .ASCIZ  /  TOTAL ERRORS SINCE LAST REPORT /
036206                                  66$:
036206  013746  001112                         MOV     $ERTTL,-(SP)    ;;SAVE $ERTTL FOR TYPEOUT
                                                                        ;;TOTAL NUMBER OF ERRORS
036212  104405                                 TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
036214  104401  001221                         TYPE    ,$CRLF          ;;TYPE CARRIAGE RETURN, LINE FEED
036220  005037  001112                         CLR     $ERTTL          ;;CLEAR ERROR TOTAL
036224  013700  000042                  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
036230  001414                                 BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
036232  005046                                 CLR     -(SP)           ;;INSURE THE "T" BIT IS CLEAR
036234  012746  036242                         MOV     #$CLR.T,-(SP)   ;;SETUP FOR AN RTI OR RTT
036240  000426                                 BR      $RTRN           ;;GO DO AN RTI OR RTT TO LOAD THE PSW
                                                                        ;;WITH A CLEARED "T" BIT
036242                                  $CLR.T:
036242  013700  000042                         MOV     @#42,R0         ;;INSURE R0 CONTAINS THE MONITORS
036246  001405                                 BEQ     $DOAGN          ;;RETURN ADDRESS
036250  000005                                 RESET                   ;;CLEAR THE WORLD
036252  004710                          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
036254  000240                                 NOP                     ;;SAVE ROOM
036256  000240                                 NOP                     ;;FOR
```

END OF PASS ROUTINE

```
        036260  000240                          NOP                     ;;ACT11
        036262                          $DOAGN:
        036262  104400                          TRAP                    ;;PUSH OLD PSW AND PC ON STACK
        036264  042716  000020                  BIC     #20,(SP)        ;;CLEAR THE ''T'' BIT
        036270  032777  010000  142642          BIT     #BIT12,@SWR     ;;RUN WITH TRACE TRAP?
        036276  001005                          BNE     1$              ;;BR IF NO
        036300  005137  001310                  COM     $TBIT           ;;IS IT TIME FOR TRACE TRAP
        036304  100402                          BMI     1$              ;;BR IF NO
        036306  052716  000020                  BIS     #20,(SP)        ;;SET TRACE TRAP
        036312  012746  036320          1$:     MOV     #$LOOP,-(SP)    ;;JUMP TO START OF TEST
        036316  000002                  $RTRN:  RTI                     ;;RETURN--THIS IS CHANGED TO
                                                                        ;;AN 'RTT'' IF 'RTT'' IS A LEGAL
                                                                        ;;INSTRUCTION
        036320                          $LOOP:
        036320  000137                          JMP     @(PC)+          ;;RETURN
        036322  020560                  $RTNAD: .WORD   LOOP
        036324     377     377     000  $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
                                                .EVEN
  4610          000001                          .END
```

SYMBOL TABLE

```
ABASE  = 000000        BIT00  = 000001        DH14    013242        DT55    016010        KDPAR3= 172366
ACDW1  = 000000        BIT01  = 000002        DH15    013322        DT57    016022        KDPAR4= 172370
ACDW2  = 000000        BIT02  = 000004        DH16    013362        EMTVEC= 000030        KDPAR5= 172372
ACPUOP= 000000         BIT03  = 000010        DH17    013442        EM1     007522        KDPAR6= 172374
ACSMPS  035334         BIT04  = 000020        DH2     012752        EM10    007631        KDPAR7= 172376
ACSMSP  035336         BIT05  = 000040        DH20    013522        EM11    007701        KDPDR0= 172320
ADDW0  = 000000        BIT06  = 000100        DH21    013562        EM12    007750        KDPDR1= 172322
ADDW1  = 000000        BIT07  = 000200        DH22    013677        EM13    010022        KDPDR2= 172324
ADDW10= 000000         BIT08  = 000400        DH24    013737        EM14    010073        KDPDR3= 172326
ADDW11= 000000         BIT09  = 001000        DH26    013777        EM15    010154        KDPDR4= 172330
ADDW12= 000000         BIT1   = 000002        DH27    014111        EM16    010237        KDPDR5= 172332
ADDW13= 000000         BIT10  = 002000        DH30    014306        EM2     007562        KDPDR6= 172334
ADDW14= 000000         BIT11  = 004000        DH32    014362        EM21    010315        KDPDR7= 172336
ADDW15= 000000         BIT12  = 010000        DH33    014401        EM22    010362        KERSTK= 001100
ADDW2  = 000000        BIT13  = 020000        DH34    014441        EM23    010431        KIPAR0= 172340
ADDW3  = 000000        BIT14  = 040000        DH36    014532        EM24    010464        KIPAR1= 172342
ADDW4  = 000000        BIT15  = 100000        DH37    014572        EM25    010523        KIPAR2= 172344
ADDW5  = 000000        BIT2   = 000004        DH40    014632        EM26    010570        KIPAR3= 172346
ADDW6  = 000000        BIT3   = 000010        DH41    014703        EM27    010636        KIPAR4= 172350
ADDW7  = 000000        BIT4   = 000020        DH42    015007        EM30    010713        KIPAR5= 172352
ADDW8  = 000000        BIT5   = 000040        DH43    015047        EM31    010756        KIPAR6= 172354
ADDW9  = 000000        BIT6   = 000100        DH44    015117        EM32    011021        KIPAR7= 172356
ADEVCT= 000000         BIT7   = 000200        DH45    015157        EM33    011057        KIPDR0= 172300
ADEVM  = 000000        BIT8   = 000400        DH55    015207        EM34    011135        KIPDR1= 172302
AENV   = 000000        BIT9   = 001000        DH57    015246        EM35    011216        KIPDR2= 172304
AENVM  = 000000        BPTVEC= 000014         DISPLA  001142        EM36    011252        KIPDR3= 172306
AFATAL= 000000         CKSWR  = 104410        DISPRE  000174        EM37    011327        KIPDR4= 172310
AMADR1= 000000         CMSG     005361        DOWELP  035316        EM40    011372        KIPDR5= 172312
AMADR2= 000000         CNTRLC   005302        DSWR   = 177570        EM41    011457        KIPDR6= 172314
AMADR3= 000000         CPSAVE   003266        DT1      015306        EM42    011523        KIPDR7= 172316
AMADR4= 000000         CPUERR= 177766         DT10     015344        EM43    011566        KSP    =%000006
AMAMS1= 000000         CR     = 000015        DT12     015356        EM44    011633        LF     = 000012
AMAMS2= 000000         CRLF   = 000200        DT13     015374        EM45    011701        LOOP     020560
AMAMS3= 000000         CSMINS   035744        DT14     015412        EM46    011740        MFPDLD   003004
AMAMS4= 000000         CSMSUB   034530        DT15     015430        EM47    012002        MFPDLP   002770
AMSGAD= 000000         CSM0   = 007000        DT16     015442        EM50    012041        MFPDPS   002772
AMSGLG= 000000         CSM1   = 007010        DT17     015460        EM51    012070        MFPDTS   002746
AMSGTY= 000000         CSM1ST   035326        DT2      015324        EM52    012126        MFPDVC   003040
AMTYP1= 000000         CSM2   = 007020        DT20     015476        EM53    012152        MFPDV1   033564
AMTYP2= 000000         CSM2ND   035330        DT21     015510        EM54    012217        MFPILD   002532
AMTYP3= 000000         CSM3   = 007030        DT22     015526        EM55    012263        MFPILP   002516
AMTYP4= 000000         CSM3RD   035332        DT24     015540        EM56    012327        MFPIPS   002520
APASS  = 000000        CSM4   = 007040        DT26     015552        EM57    012413        MFPITS   002464
APRINI   002060        CSM5   = 007050        DT27     015570        EM58    012466        MFPIVC   002622
APRIOR= 000000         CSM6   = 007060        DT30     015606        ERLOOP  002624        MFPIV1   027566
APTCSU= 000040         CSM7   = 007070        DT32     015620        ERPREP  002444        MFPIV2   030164
APTENV= 000001         DDISP  - 177570        DT34     015626        ERROR  = 104000        MFPIV3   031674
APTSIZ= 000200         DF1      016034        DT36     015642        ERRTYP  003650        MFPIV4   032304
APTSPO= 000100         DF12     016057        DT40     015654        ERRVEC= 000004        MFPIV5   032562
ASWREG= 000000         DF2      016041        DT41     015664        GTSWR  = 104407        MFPIV6   034036
ATESTN= 000000         DF3      016050        DT42     015700        HT     = 000011        MGMERR   016142
AUNIT  = 000000        DF32     016065        DT43     015712        IBSAVE  003646        MGMFLG   016144
AUSWR  = 000000        DF5      016054        DT45     015726        INIT    020666        MMR0   = 177572
AVECT1= 000000         DH1      012672        DT46     015742        IOTVEC= 000020        MMR1   = 177574
AVECT2= 000000         DH10     013042        DT47     015754        KDPAR0= 172360        MMR2   = 177576
BADPC    001304        DH12     013102        DT50     015766        KDPAR1= 172362        MMR3   = 172516
BIT0   = 000001        DH13     013162        DT52     016000        KDPAR2= 172364        MMVEC  = 000250
```

SYMBOL TABLE

```
MTPILD  002676          SDPDR0= 172220          SW5    = 000040          TYPDS  = 104405          $CKSWR  004306
MTPILP  002654          SDPDR1= 172222          SW6    = 000100          TYPE   = 104401          $CLR.T  036242
MTPIPM  002656          SDPDR2= 172224          SW7    = 000200          TYPOC  = 104402          $CMTAG  001100
MTPITA  002720          SDPDR3= 172226          SW8    = 000400          TYPON  = 104404          $CM1    = 000006
MTPITS  002626          SDPDR4= 172230          SW9    = 001000          TYPOS  = 104403          $CM2    = 000014
MTPIVC  002744          SDPDR5= 172232          TBIT   = 000020          UDPAR0= 177660          $CM3    = 000006
MTPIV1  030614          SDPDR6= 172234          TBITPS  001274          UDPAR1= 177662          $CM4    = 000006
MTPIV2  031260          SDPDR7= 172236          TBITVE= 000014          UDPAR2= 177664          $CNTLC  001312
MTPIV3  033022          SIPAR0= 172240          TESTNO  001254          UDPAR3= 177666          $CNTLG  005253
MTPIV4  033242          SIPAR1= 172242          TIMERR  016070          UDPAR4= 177670          $CNTLU  005246
NDFLAG  002242          SIPAR2= 172244          TIMFLG  016072          UDPAR5= 177672          $CPUOP  001252
NODSPA  002232          SIPAR3= 172246          TKVEC  = 000060          UDPAR6= 177674          $CRLF   001221
PATCH   016232          SIPAR4= 172250          TOFF    002356          UDPAR7= 177676          $DBLK   006762
PBAHI   001302          SIPAR5= 172252          TON     002412          UDPDR0= 177620          $DB20   007066
PBALO   001300          SIPAR6= 172254          TPVEC  = 000064          UDPDR1= 177622          $DEVCT  001234
PCSMPS  035320          SIPAR7= 172256          TRAPPC  001260          UDPDR2= 177624          $DOAGN  036262
PCSMR0  035324          SIPDR0= 172200          TRAPPS  001262          UDPDR3= 177626          $DTBL   006752
PCSMSP  035322          SIPDR1= 172202          TRAPVE= 000034          UDPDR4= 177630          $ENDAD  036252
PFECDF  004302          SIPDR2= 172204          TRTVEC= 000014          UDPDR5= 177632          $ENDCT  036054
PFECDH  004242          SIPDR3= 172206          TST1    020704          UDPDR6= 177634          $ENULL  036324
PFECDT  004272          SIPDR4= 172210          TST10   023160          UDPDR7= 177636          $ENV    001244
PFECEM  004202          SIPDR5= 172212          TST11   023462          UIPAR0= 177640          $ENVM   001245
PFECWS  004172          SIPDR6= 172214          TST12   023552          UIPAR1= 177642          $EOP    036024
PIRQ   = 177772         SIPDR7= 172216          TST13   023742          UIPAR2= 177644          $EOPCT  036046
PIRQVE= 000240          SR0    = 177572         TST14   024256          UIPAR3= 177646          $ERFLG  001103
PR0    = 000000         SR1    = 177574         TST15   025234          UIPAR4= 177650          $ERMAX  001115
PR1    = 000040         SR2    = 177576         TST16   026272          UIPAR5= 177652          $ERROR  003270
PR2    = 000100         SR3    = 172516         TST17   026542          UIPAR6= 177654          $ERRPC  001116
PR3    = 000140         SSP    =%000006         TST2    021222          UIPAR7= 177656          $ERRTB  001320
PR4    = 000200         STACK  = 001100         TST20   027016          UIPDR0= 177600          $ERTTL  001112
PR5    - 000240         START   020000          TST21   027232          UIPDR1= 177602          $ESCAP  001212
PR6    = 000300         STKLMT= 177774          TST22   027634          UIPDR2= 177604          $ETABL  001244
PR7    - 000340         SUPERM  035340          TST23   030232          UIPDR3= 177606          $ETEND  001254
PS     = 177776         SUPSTK= 000700          TST24   030662          UIPDR4= 177610          $FATAL  001226
PSW    = 177776         SWR     001140          TST25   031326          UIPDR5= 177612          $FFLG   006242
PWRMSG  007460          SWREG   000176          TST26   031742          UIPDR6= 177614          $FILLC  001156
PWRVEC- 000024          SW0    = 000001         TST27   032352          UIPDR7= 177616          $FILLS  001155
RDCHR  = 104411         SW00   = 000001         TST3    021456          USESTK= 000600          $GDADR  001120
RDLIN  = 104412         SW01   = 000002         TST30   032636          USP    =%000006         $GDDAT  001124
REGCHG  035770          SW02   = 000004         TST31   033042          VIRT1   001276          $GET42  036224
REGDAT  036006          SW03   = 000010         TST32   033262          WASR6   001256          $GTSWR  004356
REGLAB  035764          SW04   = 000020         TST33   033332          WASSR0  001264          $HD    = 000000
RESREG= 104414          SW05   = 000040         TST34   033632          WASSR1  001266          $HIBTS  000204
RESVEC- 000010          SW06   = 000100         TST35   034104          WASSR2  001270          $ICNT   001104
RETURN  035342          SW07   = 000200         TST36   034174          WASSR3  001272          $ILLUP  007452
R6     =%000006         SW08   = 000400         TST37   034266          WBIT   = 000100          $INTAG  001135
R7     =%000007         SW09   = 001000         TST4    021632          $APTHD  000204          $ITEMB  001114
SAVREG= 104413          SW1    = 000002         TST40   034360          $ATYC   006022          $LF     001222
SCOPE  = 000004         SW10   = 002000         TST41   034446          $ATY1   005776          $LFLG   006241
SDPAR0= 172260          SW11   = 004000         TST42   035344          $ATY3   006004          $LOOP   036320
SDPAR1= 172262          SW12   = 010000         TST43   035420          $ATY4   006014          $LPADR  001106
SDPAR2= 172264          SW13   = 020000         TST44   035474          $AUTOB  001134          $LPERR  001110
SDPAR3= 172266          SW14   = 040000         TST45   035542          $BDADR  001122          $MAIL   001224
SDPAR4= 172270          SW15   = 100000         TST5    022126          $BDDAT  001126          $MBADR  000206
SDPAR5= 172272          SW2    = 000004         TST6    022406          $BELL   001214          $MFLG   006240
SDPAR6= 172274          SW3    = 000010         TST7    022566          $BIN    006316          $MNEW   005271
SDPAR7= 172276          SW4    = 000020         TYPBN  - 104406          $CHARC  005772          $MSGAD  001240
```

SYMBOL TABLE

```
$MSGLG  001242          $QUES   001220          $SCOPE  003042          $TMP4   001206          $TYPE   005432
$MSGTY  001224          $RDCHR  004630          $SETUP= 000137          $TMP5   001210          $TYPEC  005644
$MSWR   005260          $RDLIN  004760          $STUP = 177777          $TN    = 000046         $TYPEX  005774
$MXCNT  001306          $RDSZ = 000010          $SVLAD  003216          $TPB    001152          $TYPOC  006344
$NULL   001154          $REGAD  001160          $SVPC = 000204          $TPFLG  001157          $TYPON  006360
$NWTST= 000001          $REG0   001162          $SWR  = 173400          $TPS    001150          $TYPOS  006320
$OCNT   006542          $REG1   001164          $SWREG  001246          $TRAP   007206          $UNIT   001236
$OCTVL  007170          $REG2   001166          $SWRMK= 000000          $TRAP2  007230          $UNITM  000214
$CMODE  006544          $REG3   001170          $TBIT   001310          $TRP   = 000015         $USWR   001250
$OVER   003252          $REG4   001172          $TESTN  001230          $TRPAD  007242          $XOFF = 000023
$PASS   001232          $REG5   001174          $TKB    001146          $TSTM   000210          $XON  = 000021
$PASTM  000212          $RESRE  007030          $TKS    001144          $TSTNM  001102          $XTSTR  003054
$PWRAD  007434          $RTNAD  036322          $TMP0   001176          $TTYIN  005236          $$GET4= 000001
$PWRDN  007274          $RTRN   036316          $TMP1   001200          $TYPBN  006244          $OFILL  006543
$PWRMG  007430          $SAVRE  006772          $TMP2   001202          $TYPDS  006546          .$X   - 000204
$PWRUP  007346          $SAVR6  007456          $TMP3   001204

. ABS.  036330      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  55080 WORDS  ( 216 PAGES)
DYNAMIC MEMORY:  20034 WORDS  ( 77 PAGES)
ELAPSED TIME:  00:10:18
CKKTBD.BIN,CKKTBD/CR/-SP/NL:TOC=CKKTBD.MLB/ML,CKKTBD.P11
```

```
CKRTBD     CREATED BY  MACRO  ON 12-JAN-82 AT 12:24     PAGE 1
                                                  SEQUENCE  149
SYMBOL CROSS REFERENCE                                    CREF   V01
SYMBOL   VALUE            REFERENCES
ABASE    = 000000          20-1249
ACDW1    = 000000          20-1249
ACDW2    = 000000          20-1249
ACPUOP   = 000000          20-1249    20-1249
ACSMPS     035334          54-2027   *105-4437  *105-4438   105-4439  *105-4444  *105-4445   105-4446   105-4450  #105-4505
ACSMSP     035336          54-2018   *105-4426   105-4467  #105-4506
ADDW0    = 000000          20-1249
ADDW1    = 000000          20-1249
ADDW10   = 000000          20-1249
ADDW11   = 000000          20-1249
ADDW12   = 000000          20-1249
ADDW13   = 000000          20-1249
ADDW14   = 000000          20-1249
ADDW15   = 000000          20-1249
ADDW2    = 000000          20-1249
ADDW3    = 000000          20-1249
ADDW4    = 000000          20-1249
ADDW5    = 000000          20-1249
ADDW6    = 000000          20-1249
ADDW7    = 000000          20-1249
ADDW8    = 000000          20-1249
ADDW9    = 000000          20-1249
ADEVCT   = 000000          20-1249    20-1249
ADEVM    = 000000          20-1249
AENV     = 000000          20-1249    20-1249
AENVM    = 000000          20-1249    20-1249
AFATAL   = 000000          20-1249    20-1249
AMADR1   = 000000          20-1249
AMADR2   = 000000          20-1249
AMADR3   = 000000          20-1249
AMADR4   = 000000          20-1249
AMAMS1   = 000000          20-1249
AMAMS2   = 000000          20-1249
AMAMS3   = 000000          20-1249
AMAMS4   = 000000          20-1249
AMSGAD   = 000000          20-1249    20-1249
AMSGLG   = 000000          20-1249    20-1249
AMSGTY   = 000000          20-1249    20-1249
AMTYP1     000000          20-1249
AMTYP2   - 000000          20-1249
AMTYP3   - 000000          20-1249
AMTYP4   = 000000          20-1249
APASS    - 000000          20-1249    20-1249
APRINI     002060         #27-1532    58-2126     79-3360     104-4394
APRIOR   - 000000          20-1249
APTCSU   - 000040          43-1903   #44-1904
APTENV   - 000001          37-1787    43-1903     44-1904    #44-1904
APTSIZ   - 000200         #44-1904    58-2106
APTSPO   - 000100          43-1903    44-1904    #44-1904
ASWREG   = 000000          20-1249    20-1249
ATESTN   = 000000          20-1249    20-1249
AUNIT    = 000000          20-1249    20-1249
```

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
AUSWR	= 000000	20-1249	20-1249							
AVECT1	= 000000	20-1249								
AVECT2	= 000000	20-1249								
BADPC	001304	#20-1249	*28-1591	54-2023						
BIT0	= 000001	#15-1197	93-4084	93-4121	98-4244	98-4277				
BIT00	= 000001	#15-1197	15-1197	36-1785	36-1785	37-1787	37-1787	105-4443	105-4454	105-4466
		105-4472	105-4478	105-4485						
BIT01	= 000002	#15-1197	15-1197							
BIT02	= 000004	#15-1197	15-1197							
BIT03	= 000010	#15-1197	15-1197							
BIT04	= 000020	#15-1197	15-1197							
BIT05	= 000040	#15-1197	15-1197							
BIT06	= 000100	#15-1197	15-1197							
BIT07	= 000200	#15-1197	15-1197							
BIT08	= 000400	#15-1197	15-1197	36-1785						
BIT09	= 001000	#15-1197	15-1197	36-1785	37-1787	105-4493				
BIT1	= 000002	#15-1197	89-3971	89-4004	91-4034	91-4071	96-4180	96-4212		
BIT10	= 002000	#15-1197	37-1787							
BIT11	= 004000	#15-1197								
BIT12	= 010000	#15-1197	110-4609							
BIT13	= 020000	#15-1197	37-1787							
BIT14	= 040000	#15-1197	36-1785	105-4436	105-4451	106-4520				
BIT15	= 100000	#15-1197	105-4452							
BIT2	= 000004	#15-1197	28-1582	28-1606	77-3312	77-3320				
BIT3	= 000010	#15-1197	103-4370							
BIT4	= 000020	#15-1197	15-1208	58-2125						
BIT5	= 000040	#15-1197								
BIT6	= 000100	#15-1197	15-1209							
BIT7	= 000200	#15-1197								
BIT8	= 000400	#15-1197								
BIT9	= 001000	#15-1197								
BPTVEC	= 000014	#15-1197								
CKSWR	= 104410	36-1785	37-1787	37-1787	#50-1910					
CMSG	005361	41-1888	#41-1899							
CNTRLC	005302	40-1882	40-1882	#41-1886						
CPSAVE	003266	*36-1785	36-1785	#36-1785	*37-1787	37-1787	39-1880			
CPUERR	= 177766	#15-1210	54-2003	*56-2066						
CR	= 000015	#15-1197	43-1903	43-1903						
CRLF	= 000200	#15-1197	43-1903	43-1903	52-1957	52-1958	53-1973	53-1977	53-1979	53-1980
		53-1981	53-1982	53-1984	53-1992	58-2107	58-2107			
CSMINS	035744	109-4571	#109-4603							
CSMSUB	034530	104-4399	#105-4405	106-4523	107-4541	108-4558				
CSM0	= 007000	#15-1185	100-4306	101-4329	102-4352	103-4375	105-4417			
CSM1	007010	#15-1186	109-4603							
CSM1ST	035326	54-2028	*105-4427	105-4473	#105-4502					
CSM2	= 007020	#15-1187	109-4603							
CSM2ND	035330	54-2029	*105-4428	105-4479	#105-4503					
CSM3	= 007030	#15-1188	109-4603							
CSM3RD	035332	54-2030	*105-4429	105-4486	#105-4504					
CSM4	= 007040	#15-1189	109-4603							
CSM5	= 007050	#15-1190	109-4603							
CSM6	= 007060	#15-1191	109-4603							

SYMBOL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES								
CSM7	= 007070	#15-1192	109-4603							
DDISP	= 177570	#15-1197	20-1249	58-2106						
DF1	016034	23-1392	24-1422	24-1434	24-1447	25-1458	25-1464	25-1471	25-1478	25-1485
		#55-2035								
DF12	016057	21-1254	21-1278	21-1284	21-1290	22-1301	22-1307	22-1320	23-1350	23-1360
		#55-2039								
DF2	016041	21-1260	24-1453	#55-2036						
DF3	016050	21-1266	21-1272	22-1295	22-1313	22-1326	22-1332	22-1338	23-1367	23-1374
		23-1386	24-1403	24-1409	24-1428	24-1441	25-1491	25-1509	26-1514	26-1522
		#55-2037								
DF32	016065	23-1380	23-1398	#55-2040						
DF5	016054	23-1343	24-1415	25-1497	25-1503	#55-2038				
DH1	012672	21-1252	#53-1963							
DH10	013042	21-1264	21-1270	#53-1965						
DH12	013102	21-1276	#53-1966							
DH13	013162	21-1282	#53-1967							
DH14	013242	21-1288	#53-1968							
DH15	013322	22-1293	#53-1969							
DH16	013362	22-1299	#53-1970							
DH17	013442	22-1305	#53-1971							
DH2	012752	21-1258	#53-1964							
DH20	013522	22-1311	22-1330	23-1341	26-1520	#53-1972				
DH21	013562	22-1317	#53-1973							
DH22	013677	22-1324	#53-1975							
DH24	013737	22-1336	#53-1976							
DH26	013777	23-1347	#53-1977							
DH27	014111	23-1354	#53-1979							
DH30	014306	23-1364	23-1371	24-1438	25-1468	25-1475	25-1482	#53-1984		
DH32	014362	23-1378	23-1396	24-1451	25-1495	25-1501	#53-1986			
DH33	014401	23-1384	#53-1987							
DH34	014441	23-1390	#53-1988							
DH36	014532	24-1401	#53-1989							
DH37	014572	24-1407	#53-1990							
DH40	014632	24-1413	#53-1991							
DH41	014703	24-1419	#53-1992							
DH42	015007	24-1426	#53-1994							
DH43	015047	24-1432	24-1445	#53-1995						
DH44	015117	25-1456	25-1462	25-1489	#53-1996					
DH45	015157	#53-1997								
DH55	015207	25-1507	#53-1998							
DH57	015246	26-1512	#53-1999							
DISPLA	001142	#20-1249	36-1785	37-1787	*58-2106	*58-2106				
DISPRE	000174	#17-1246	58-2106							
DOWELP	035316	*105-4410	105-4430	*105-4448	*105-4463	*105-4469	*105-4475	*105-4481	*105-4488	105-4491
		#105-4498								
DSWR	= 177570	#15-1197	20-1249	58-2106						
DT1	015306	21-1253	#54-2003							
DT10	015344	21-1265	21-1271	#54-2005						
DT12	015356	21-1277	#54-2006							
DT13	015374	21-1283	#54-2007							
DT14	015412	21-1289	#54-2008							
DT15	015430	22-1294	#54-2009							

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES					
DT16	015442	22-1300	#54-2010				
DT17	015460	22-1306	#54-2011				
DT2	015324	21-1259	#54-2004				
DT20	015476	22-1312	22-1331	23-1342	26-1521	#54-2012	
DT21	015510	22-1319	#54-2013				
DT22	015526	22-1325	23-1385	#54-2014			
DT24	015540	22-1337	24-1408	#54-2015			
DT26	015552	23-1349	#54-2016				
DT27	015570	23-1359	#54-2017				
DT30	015606	23-1366	23-1373	24-1440	25-1470	#54-2018	
DT32	015620	23-1379	23-1397	24-1452	25-1496	25-1502	#54-2019
DT34	015626	23-1391	#54-2020				
DT36	015642	24-1402	#54-2021				
DT40	015654	24-1414	#54-2022				
DT41	015664	24-1421	#54-2023				
DT42	015700	24-1427	#54-2024				
DT43	015712	24-1433	#54-2025				
DT45	015726	24-1446	#54-2026				
DT46	015742	25-1457	25-1463	#54-2027			
DT47	015754	25-1477	#54-2028				
DT50	015766	25-1484	#54-2029				
DT52	016000	25-1490	#54-2030				
DT55	016010	25-1508	#54-2031				
DT57	016022	26-1513	#54-2032				
EMTVEC =	000030	#15-1197	*58-2106	*58-2106			
EM1	007522	21-1251	#52-1917				
EM10	007631	21-1263	#52-1919				
EM11	007701	21-1269	#52-1920				
EM12	007750	21-1275	#52-1921				
EM13	010022	21-1281	22-1304	22-1310	#52-1922		
EM14	010073	21-1287	#52-1923				
EM15	010154	22-1292	#52-1924				
EM16	010237	22-1298	#52-1925				
EM2	007562	21-1257	#52-1918				
EM21	010315	22-1316	#52-1926				
EM22	010362	22-1323	#52-1927				
EM23	010431	22-1329	#52-1928				
EM24	010464	22-1335	#52-1929				
EM25	010523	23-1340	#52-1930				
EM26	010570	23-1346	#52-1931				
EM27	010636	23-1353	#52-1932				
EM30	010713	23-1363	#52-1933				
EM31	010756	23-1370	#52-1934				
EM32	011021	23-1377	#52-1935				
EM33	011057	23-1383	#52-1936				
EM34	011135	23-1389	#52-1937				
EM35	011216	23-1395	#52-1938				
EM36	011252	24-1400	#52-1939				
EM37	011327	24-1406	#52-1940				
EM40	011372	24-1412	#52-1941				
EM41	011457	24-1418	#52-1942				
EM42	011523	24-1425	#52-1943				

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
EM43	011566	24-1431	24-1444	#52-1944						
EM44	011633	24-1437	#52-1945							
EM45	011701	24-1450	#52-1946							
EM46	011740	25-1455	#52-1947							
EM47	012002	25-1461	#52-1948							
EM50	012041	25-1467	#52-1949							
EM51	012070	25-1474	#52-1950							
EM52	012126	25-1481	#52-1951							
EM53	012152	25-1488	#52-1952							
EM54	012217	25-1494	#52-1953							
EM55	012263	25-1500	#52-1954							
EM56	012327	25-1506	#52-1955							
EM57	012413	26-1511	#52-1956							
EM58	012466	26-1517	#52-1957							
ERLOOP	002624	*31-1655	*32-1676	32-1697	#32-1704	*33-1722	33-1744	*34-1765	34-1780	
ERPREP	002444	#31-1654	32-1695	33-1742	34-1779					
ERROR	= 104000	#15-1197	28-1599	28-1601	56-2064	57-2093	60-2171	60-2179	60-2187	60-2194
		61-2242	61-2250	61-2258	61-2265	62-2299	62-2310	62-2317	62-2321	64-2374
		64-2385	64-2392	64-2406	65-2445	65-2456	65-2463	65-2477	66-2518	67-2553
		67-2562	67-2586	67-2602	67-2611	68-2653	68-2679	69-2710	70-2737	70-2755
		71-2822	77-3326	77-3330	79-3377	79-3380	79-3392	79-3399	79-3404	79-3411
		79-3419	79-3425	79-3432	80-3443	81-3479	81-3482	81-3493	81-3500	81-3505
		81-3512	81-3520	81-3526	81-3533	82-3544	83-3575	83-3591	83-3600	83-3607
		83-3616	83-3626	83-3634	83-3643	83-3652	84-3670	84-3687	84-3696	84-3703
		84-3712	84-3722	84-3730	84-3740	84-3749	85-3786	85-3789	85-3800	85-3807
		85-3812	85-3819	85-3827	85-3833	85-3840	86-3853	87-3876	87-3879	87-3891
		87-3898	87-3903	87-3910	87-3918	87-3924	87-3931	88-3943	89-3981	89-3988
		89-3994	89-4001	90-4014	91-4045	91-4052	91-4061	91-4069	92-4078	93-4095
		93-4102	93-4111	93-4119	94-4128	95-4148	95-4153	96-4190	96-4197	96-4203
		96-4210	97-4224	98-4255	98-4262	98-4268	98-4275	99-4288	100-4313	101-4336
		102-4359	103-4381	105-4424	105-4442	105-4453	105-4465	105-4471	105-4477	105-4484
		105-4490	109-4583	109-4591						
ERRTYP	003650	37-1787	#39-1789							
ERRVEC	= 000004	#15-1197	36-1785	*36-1785	*36-1785	*36-1785	58-2106	*58-2106	*58-2106	*58-2116
		*58-2117								
GNS	= ******	17-1246	17-1246	50-1910	50-1910	50-1910	50-1910	50-1910	50-1910	50-1910
		50-1910	50-1910	50-1910	50-1910	50-1910	50-1910	50-1910	50-1910	50-1910
		50-1910	50-1910	50-1910	50-1910	50-1910	50-1910	50-1910	50-1910	58-2107
		58-2108	110-4609	110-4609						
GTSWR	= 104407	41-1895	#50-1910	58-2107						
HT	= 000011	#15-1197	43-1903	43-1903						
IBSAVE	003646	*37-1787	37-1787	*37-1787	*37-1787	37-1787	37-1787	37-1787	37-1787	#37-1787
INIT	020666	#58-2124								
IOTVEC	= 000020	#15-1197	*58-2106	*58-2106						
KDPAR0	= 172360	#15-1198	27-1548	*77-3313	*77-3318					
KDPAR1	= 172362	#15-1198								
KDPAR2	= 172364	#15-1198								
KDPAR3	= 172366	#15-1198								
KDPAR4	= 172370	#15-1198	*73-2860	*77-3311						
KDPAR5	= 172372	#15-1198								
KDPAR6	= 172374	#15-1198								
KDPAR7	= 172376	#15-1198								

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES							
KDPDR0	= 172320	#15-1198							
KDPDR1	= 172322	#15-1198	*73-2857	*74-3011					
KDPDR2	= 172324	#15-1198	*73-2858	*74-3012					
KDPDR3	= 172326	#15-1198	*73-2859	*74-3013					
KDPDR4	= 172330	#15-1198	*77-3310	*89-3965	*95-4140	*96-4173			
KDPDR5	= 172332	#15-1198							
KDPDR6	= 172334	#15-1198							
KDPDR7	= 172336	#15-1198	27-1539						
KERSTK	= 001100	#15-1211	62-2305	64-2362	64-2366	64-2379	65-2433	65-2437	65-2450	66-2514
		67-2570	67-2575	68-2636	68-2658	68-2684	70-2741	71-2798	74-3177	77-3319
		79-3371	81-3473	85-3783	85-3844	87-3873				
KIPAR0	= 172340	#15-1198	27-1547							
KIPAR1	= 172342	#15-1198								
KIPAR2	= 172344	#15-1198								
KIPAR3	= 172346	#15-1198	*60-2152	*66-2501						
KIPAR4	= 172350	#15-1198	*60-2153	*66-2502	*70-2730	*79-3362				
KIPAR5	= 172352	#15-1198	*67-2545							
KIPAR6	= 172354	#15-1198								
KIPAR7	= 172356	#15-1198	27-1557							
KIPDR0	= 172300	#15-1198	27-1535							
KIPDR1	= 172302	#15-1198								
KIPDR2	= 172304	#15-1198								
KIPDR3	= 172306	#15-1198	*64-2355	*66-2503	*74-3014	*74-3176				
KIPDR4	= 172310	#15-1198	*33-1732	*33-1736	*60-2163	*61-2234	*64-2360	*65-2431	*66-2504	*66-2528
		*67-2546	*67-2615	*70-2732	*70-2762	*73-2856	*74-3010	*79-3366	*79-3434	*81-3466
		*81-3535	*89-3972	*89-4003	*96-4181	*96-4214	*98-4246	*98-4279		
KIPDR5	= 172312	#15-1198	*64-2356	*67-2547	*67-2616					
KIPDR6	= 172314	#15-1198								
KIPDR7	= 172316	#15-1198								
KSP	=%000006	#15-1205	28-1591	*28-1592	*28-1593	*28-1604	*28-1605	*33-1731	*39-1790	*39-1818
		*39-1843	39-1845	*39-1846	*39-1850	*39-1851	*39-1857	39-1859	*39-1860	*39-1863
		*39-1864	*39-1870	*39-1871	*56-2061	*56-2062	56-2063	*56-2067	*56-2068	*57-2087
		*57-2088	57-2089	*57-2095	*57-2096	*58-2110	*64-2362	*64-2366	*64-2379	*64-2401
		*64-2402	*64-2410	*64-2411	*65-2433	*65-2437	*65-2450	*65-2472	*65-2473	*65-2481
		*65-2482	*66-2514	*67-2570	*67-2575	*68-2636	*68-2658	*68-2684	*70-2741	71-2792
		71-2793	*71-2798	*74-3177	*77-3306	*77-3319	79-3371	*79-3373	*80-3438	*80-3439
		*80-3445	*80-3446	81-3473	*81-3475	*82-3539	*82-3540	*82-3546	*82-3547	*83-3567
		*83-3572	*83-3578	*83-3647	*83-3648	*83-3654	*83-3655	*84-3662	*84-3667	*84-3673
		*84-3744	*84-3745	*84-3751	*84-3752	85-3778	*85-3844	*86-3848	*86-3849	*86-3855
		*86-3856	87-3868	*88-3938	*88-3939	*88-3945	*88-3946	*90-4012	*90-4013	*90-4016
		*90-4017	*95-4143	95-4144	95-4145	95-4151	*95-4155	*97-4222	*97-4223	*97-4226
		*97-4227	*99-4286	*99-4287	*99-4290	*99-4291				
LF	= 000012	#15-1197	43-1903	43-1903						
LOOP	020560	#58-2109	110-4609							
MFPDLD	003004	*34-1767	*34-1768	#34-1774						
MFPDLP	002770	#34-1770	34-1781	96-4183						
MFPDPS	002772	#34-1771	*96-4184							
MFPDTS	002746	#34-1765	96-4187	96-4194	96-4201	96-4207				
MFPDVC	003040	34-1773	#34-1783	*96-4185						
MFPDV1	033564	96-4185	#97-4219							
MFPILD	002532	*32-1678	*32-1679	#32-1687						
MFPILP	002516	#32-1683	32-1698	79-3384	81-3485	85-3792	87-3883	89-3974	98-4248	

SP
ST

SW
TRI
TY
TY
TY
TY
TY
TY
TY
US
$$
$$
$$
$$

$$

$$
$$
$$
$$

.E
.E

.H
.K
.S
.S
.S
.$

CKKTBD CREATED BY MACRO ON 12-JAN-82 AT 12:24 PAGE 7
SEQUENCE 155

SYMBOL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES								
MFPIPS	002520	#32-1684	*79-3386	*81-3486	*85-3793	*87-3884	*89-3975	*98-4249		
MFPITS	002464	#32-1676	79-3389	79-3396	79-3402	79-3408	79-3416	79-3423	79-3430	81-3490
		81-3497	81-3503	81-3509	81-3517	81-3524	81-3531	85-3797	85-3804	85-3810
		85-3816	85-3824	85-3831	85-3838	87-3888	87-3895	87-3901	87-3907	87-3915
		87-3922	87-3929	89-3978	89-3985	89-3992	89-3998	98-4252	98-4259	98-4266
		98-4272								
MFPIVC	002622	32-1686	#32-1703	*79-3388	*81-3488	*85-3794	*87-3885	*89-3976	*98-4250	
MFPIV1	027566	79-3388	#80-3438							
MFPIV2	030164	81-3470	81-3488	#82-3539						
MFPIV3	031674	85-3777	85-3794	#86-3848						
MFPIV4	032304	87-3865	87-3885	#88-3938						
MFPIV5	032562	89-3976	#90-4008							
MFPIV6	034036	98-4250	#99-4283							
MGMERR	016142	32-1688	33-1735	34-1775	#57-2079	58-2118	60-2216	61-2281	62-2301	62-2304
		64-2414	65-2485	66-2529	67-2617	70-2761	71-2831	77-3332	81-3472	83-3571
		83-3581	84-3666	84-3676	85-3779	87-3869				
MGMFLG	016144	#57-2080	*57-2094	*58-2122						
MMR0	= 177572	#15-1201	28-1594	*28-1602	*58-2124	*60-2167	*60-2198	*61-2238	*61-2269	*62-2297
		*73-2854	*73-2863	*73-2882	*73-2887	*73-2892	*73-2897	*73-2902	*73-2911	*73-2916
		*73-2922	*73-2927	*73-2932	*73-2937	*73-2942	*73-2947	*73-2952	*73-2957	*73-2963
		*73-2968	*73-2972	*73-2977	*73-2982	*73-2986	*73-2992	*73-2997	*74-3027	*74-3033
		*74-3037	*74-3043	*74-3055	*74-3081	*74-3087	*74-3093	*74-3099	*74-3103	*74-3108
		*74-3114	*74-3125	*74-3131	*74-3136	*74-3142	*74-3149	*74-3155	*74-3162	*74-3165
		*74-3172	*74-3175	*75-3191	*75-3197	*75-3203	*75-3206	*75-3211	*75-3215	*75-3221
		*75-3224	*75-3228	*75-3231	*76-3247	*76-3253	*76-3259	*76-3262	*76-3267	*76-3271
		*76-3276	*76-3279	*76-3284	*76-3287	*77-3307	77-3321	*77-3331	*79-3361	90-4008
		92-4075	94-4125	97-4219	99-4283	*100-4305	*100-4308	*101-4328	*101-4331	*102-4351
		*102-4354	*103-4374	*103-4377	*105-4416	*105-4422	*105-4441	*105-4443	*105-4449	*105-4454
		*105-4464	*105-4466	*105-4470	*105-4472	*105-4476	*105-4478	*105-4482	*105-4485	*105-4489
		*109-4599								
MMR1	= 177574	#15-1202	28-1595	77-3322	90-4009	92-4076	94-4126	97-4220	99-4284	
MMR2	= 177576	#15-1203	28-1596	77-3323	90-4010	92-4077	94-4127	97-4221	99-4285	
MMR3	= 172516	#15-1204	*28-1582	*28-1606	*58-2125	*73-2864	*75-3189	*75-3232	*76-3245	*76-3288
		*77-3312	*77-3320	*89-3971	*89-4004	90-4011	*91-4034	*91-4071	*93-4084	*93-4121
		*96-4180	*96-4212	*98-4244	*98-4277	*100-4301	*101-4324	*102-4347	*103-4370	*104-4396
		*106-4518	*107-4536	*108-4554	*109-4569	*109-4600				
MMVEC	= 000250	#15-1198	*32-1686	*32-1688	*33-1733	*33-1735	*34-1773	*34-1775	*58-2118	*58-2119
		*60-2162	*60-2216	*61-2233	*61-2281	*62-2296	*62-2301	*62-2304	*64-2359	*64-2372
		*64-2414	*65-2430	*65-2443	*65-2485	*66-2507	*66-2529	*67-2567	*67-2573	*67-2617
		*73-2861	*77-3302	*77-3303	*77-3332	*77-3333	*81-3470	*81-3472	*83-3568	*83-3571
		*83-3579	*83-3581	*84-3663	*84-3666	*84-3674	*84-3676	*85-3777	*85-3779	*87-3865
		*87-3869								
MTPILD	002676	*33-1724	*33-1725	#33-1734						
MTPILP	002654	#33-1728	33-1745	83-3583	84-3678	91-4036	93-4086			
MTPIPM	002656	#33-1729	*83-3584	*84-3679	*91-4037	*93-4087				
MTPITA	002720	*33-1726	#33-1738							
MTPITS	002626	#33-1722	83-3587	83-3596	83-3604	83-3612	83-3622	83-3631	83-3640	84-3683
		84-3692	84-3700	84-3708	84-3718	84-3727	84-3737	91-4041	91-4049	91-4057
		91-4066	93-4091	93-4099	93-4107	93-4116				
MTPIVC	002744	33-1733	#33-1747	*84-3680	*91-4038	*93-4088				
MTPIV1	030614	83-3568	83-3579	#83-3647						
MTPIV2	031260	84-3663	84-3674	84-3680	#84-3744					

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
MTPIV3	033022	91-4038	#92-4075							
MTPIV4	033242	93-4088	#94-4125							
NDFLAG	002242	#28-1584	*28-1603							
NODSPA	002232	#28-1581	73-2861							
PATCH	016232	#57-2100								
PBAHI	001302	#20-1249								
PBALO	001300	#20-1249								
PCSMPS	035320	*104-4397	105-4413	105-4431	105-4459	#105-4499	*106-4519	*107-4537	*108-4555	
PCSMRO	035324	*105-4407	#105-4501							
PCSMSP	035322	*105-4414	*105-4415	105-4467	#105-4500					
PFECDF	004302	39-1876	#39-1881							
PFECDH	004242	39-1876	#39-1878							
PFECDT	004272	39-1876	#39-1880							
PFECEM	004202	39-1876	#39-1877							
PFECWS	004172	39-1801	#39-1876							
PIRQ	= 177772	#15-1197								
PIRQVE	= 000240	#15-1197								
PR0	= 000000	#15-1197								
PR1	= 000040	#15-1197								
PR2	= 000100	#15-1197								
PR3	= 000140	#15-1197								
PR4	= 000200	#15-1197								
PR5	= 000240	#15-1197								
PR6	= 000300	#15-1197								
PR7	= 000340	#15-1197								
PS	= 177776	#15-1197	15-1197							
PSW	= 177776	#15-1197	29-1617	29-1619	32-1685	*32-1692	*32-1699	33-1730	34-1772	*58-2111
		*58-2113	*58-2115	60-2169	*60-2202	*60-2207	*60-2213	*60-2215	61-2240	*61-2273
		*61-2278	*61-2280	*62-2298	*68-2635	*68-2643	68-2650	*68-2652	*68-2657	*68-2660
		*68-2662	*68-2669	68-2676	*68-2678	*68-2683	*68-2686	*68-2688	*69-2701	69-2705
		*69-2707	*71-2788	*71-2797	*71-2799	*71-2801	*75-3190	*76-3246	*76-3290	77-3315
		77-3317	*79-3369	*81-3469	*83-3566	*83-3577	*84-3661	*84-3672	*85-3770	*85-3776
		*85-3842	*87-3860	*87-3867	*87-3933	*95-4141	*98-4278	*100-4302	*100-4309	*101-4325
		*101-4332	*102-4348	*102-4355	*103-4371	*104-4401	*105-4406	*105-4413	*105-4421	105-4437
		105-4444	105-4455	*106-4520	*106-4525	*107-4538	*107-4543	*108-4560	*109-4568	*109-4601
PWRMSG	007460	51-1911	#51-1912							
PWRVEC	= 000024	#15-1197	*51-1911	*51-1911	*51-1911	*51-1911	*51-1911	*51-1911	*58-2106	*58-2106
RDCHR	= 104411	40-1882	#50-1910							
RDLIN	= 104412	#50-1910								
REGCHG	035770	109-4572	#109-4607							
REGDAT	036006	109-4573	#109-4608							
REGLAB	035764	109-4576	109-4604	#109-4605	109-4605	109-4607	109-4607	109-4607	109-4607	109-4607
		109-4607	109-4607	109-4608	109-4608	109-4608	109-4608	109-4608	109-4608	
RESREG	= 104414	49-1909	#50-1910							
RESVEC	= 000010	#15-1197	*58-2106	58-2106	*58-2106	100-4303	*100-4304	*100-4307	101-4326	*101-4327
		*101-4330	102-4349	*102-4350	*102-4353	103-4372	*103-4373	*103-4376	*105-4411	105-4418
		*105-4418	*109-4574							
RETURN	035342	*105-4409	105-4425	105-4496	#105-4508					
R6	=%000006	#15-1197	*58-2106	*58-2106	58-2106					
R7	=%000007	#15-1197								
SAVREG	= 104413	49-1909	#50-1910							
SCOPE	- 000004	#15-1197	60-2150	61-2226	62-2294	64-2354	65-2427	66-2500	67-2544	68-2632

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
		69-2697	70-2729	71-2777	73-2852	74-3008	75-3186	76-3242	77-3299	79-3359
		81-3462	83-3563	84-3657	85-3768	87-3858	89-3962	91-4033	93-4081	95-4139
		96-4169	98-4240	100-4300	101-4323	102-4346	103-4369	104-4392	106-4517	107-4535
		108-4553	109-4567	110-4609						
SDPAR0	= 172260	#15-1198	27-1560	*104-4395	*104-4402					
SDPAR1	= 172262	#15-1198								
SDPAR2	= 172264	#15-1198								
SDPAR3	= 172266	#15-1198								
SDPAR4	= 172270	#15-1198	*96-4170							
SDPAR5	= 172272	#15-1198								
SDPAR6	= 172274	#15-1198								
SDPAR7	= 172276	#15-1198								
SDPDR0	= 172220	#15-1198								
SDPDR1	= 172222	#15-1198	*75-3187	*75-3198						
SDPDR2	= 172224	#15-1198								
SDPDR3	= 172226	#15-1198								
SDPDR4	= 172230	#15-1198	*89-3966	*96-4177	*98-4245					
SDPDR5	= 172232	#15-1198								
SDPDR6	= 172234	#15-1198								
SDPDR7	= 172236	#15-1198	27-1543							
SIPAR0	= 172240	#15-1198	27-1559	*68-2666	*68-2685	*106-4521	*106-4526			
SIPAR1	= 172242	#15-1198								
SIPAR2	= 172244	#15-1198								
SIPAR3	= 172246	#15-1198	*60-2154							
SIPAR4	= 172250	#15-1198	*60-2155	*79-3363						
SIPAR5	= 172252	#15-1198								
SIPAR6	= 172254	#15-1198								
SIPAR7	= 172256	#15-1198	27-1562							
SIPDR0	= 172200	#15-1198	27-1541							
SIPDR1	= 172202	#15-1198								
SIPDR2	= 172204	#15-1198								
SIPDR3	= 172206	#15-1198	*75-3199	*75-3233						
SIPDR4	= 172210	#15-1198	*60-2164	*61-2235	*83-3564	*85-3775	*85-3843	*93-4082	*96-4174	
SIPDR5	= 172212	#15-1198								
SIPDR6	= 172214	#15-1198								
SIPDR7	= 172216	#15-1198								
SR0	= 177572	#15-1198	15-1201	57-2090	*57-2092	60-2183	61-2254	62-2306	62-2309	*62-2326
		64-2380	*64-2396	64-2403	*64-2405	65-2451	*65-2467	65-2474	*65-2476	*66-2522
		67-2571	67-2576	67-2592	*67-2606	70-2743	*70-2759	71-2794	*71-2796	80-3440
		*80-3442	82-3541	*82-3543	83-3649	*83-3651	84-3746	*84-3748	86-3850	*86-3852
		88-3940	*88-3942							
SR1	= 177574	#15-1198	15-1202							
SR2	= 177576	#15-1198	15-1203	57-2091	60-2184	61-2255	62-2303	64-2381	64-2404	65-2452
		65-2475	66-2515	67-2549	67-2558	67-2572	67-2577	67-2596	67-2607	70-2744
		71-2795	80-3441	82-3542	83-3650	84-3747	86-3851	88-3941		
SR3	- 172516	#15-1198	15-1204							
SSP	%000006	#15-1206	*58-2112	*68-2670	*68-2687	79-3370	*83-3569	83-3570	*83-3580	85-3780
		*85-3782	*104-4398	105-4426	*106-4522	*107-4540	*108-4557			
STACK	= 001100	#15-1197	15-1211	15-1212	15-1213	58-2106	58-2110	95-4142	95-4155	
START	020000	17-1246	51-1911	#58-2106						
STKLMT	= 177774	#15-1197								
SUPERM	035340	105-4446	#105-4507							

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
SUPSTK	= 000700	#15-1212	58-2112	68-2670	68-2687	79-3374	83-3578	85-3780	104-4398	106-4522
SWR	001140	#20-1249	36-1785	36-1785	36-1785	36-1785	37-1787	37-1787	37-1787	37-1787
		40-1882	40-1882	51-1911	51-1911	58-2106	*58-2106	58-2106	*58-2106	*58-2106
		58-2107	105-4493	110-4609						
SWREG	000176	#17-1246	40-1882	40-1882	58-2106	58-2107				
SW0	= 000001	#15-1197								
SW00	= 000001	#15-1197	15-1197							
SW01	= 000002	#15-1197	15-1197							
SW02	= 000004	#15-1197	15-1197							
SW03	= 000010	#15-1197	15-1197							
SW04	= 000020	#15-1197	15-1197							
SW05	= 000040	#15-1197	15-1197							
SW06	= 000100	#15-1197	15-1197							
SW07	= 000200	#15-1197	15-1197							
SW08	= 000400	#15-1197	15-1197							
SW09	= 001000	#15-1197	15-1197							
SW1	= 000002	#15-1197								
SW10	= 002000	#15-1197								
SW11	= 004000	#15-1197								
SW12	= 010000	#15-1197								
SW13	= 020000	#15-1197								
SW14	= 040000	#15-1197								
SW15	= 100000	#15-1197								
SW2	= 000004	#15-1197								
SW3	= 000010	#15-1197								
SW4	= 000020	#15-1197								
SW5	= 000040	#15-1197								
SW6	= 000100	#15-1197								
SW7	= 000200	#15-1197								
SW8	= 000400	#15-1197								
SW9	= 001000	#15-1197								
TBIT	= 000020	#15-1208	29-1617	29-1622	30-1634					
TBITPS	001274	#20-1249	*29-1620	30-1634	30-1636	*30-1637	*58-2123			
TBITVE	= 000014	#15-1197	*58-2106	*58-2106						
TESTNO	001254	#20-1249	*37-1787	54-2003	54-2004	54-2005	54-2006	54-2007	54-2008	54-2009
		54-2010	54-2011	54-2012	54-2013	54-2014	54-2015	54-2016	54-2017	54-2018
		54-2019	54-2020	54-2021	54-2022	54-2023	54-2024	54-2025	54-2026	54-2027
		54-2028	54-2029	54-2030	54-2031	54-2032				
TIMERR	016070	#56-2053	58-2116	68-2689	70-2760	71-2829				
TIMFLG	016072	#56-2054	*56-2065	*58-2121						
TKVEC	= 000060	#15-1197								
TOFF	002356	#29-1617	68-2633	71-2778	77-3300					
TON	002412	#30-1634	68-2690	71-2833						
TPVEC	= 000064	#15-1197								
TRAPPC	001260	#20-1249	*28-1592	28-1605	54-2003	54-2004	*56-2061	56-2068	*57-2087	57-2096
		*64-2401	64-2411	*65-2472	65-2482	*80-3438	80-3446	*82-3539	82-3547	*83-3647
		83-3655	*84-3744	84-3752	*86-3848	86-3856	*88-3938	88-3946	*90-4012	90-4017
		*97-4222	97-4227	*99-4286	99-4291					
TRAPPS	001262	#20-1249	*28-1593	28-1604	54-2003	54-2004	*56-2062	56-2067	*57-2088	57-2095
		*64-2402	64-2410	*65-2473	65-2481	*80-3439	80-3445	*82-3540	82-3546	*83-3648
		83-3654	*84-3745	84-3751	*86-3849	86-3855	*88-3939	88-3945	*90-4013	90-4016
		*97-4223	97-4226	*99-4287	99-4290					

SYMBOL CROSS REFEPENCE

SYMBOL	VALUE	REFERENCES							
TRAPVE	- 000034	#15-1197	*58-2106	*58-2106					
TRTVEC	= 000014	#15-1197							
TST*	020704	#60-2150							
TST*0	023160	#68-2632							
TST11	023462	#69-2697							
TST12	023552	69-2715	#70-2729						
TST13	023742	#71-2777							
TST14	024256	#73-2852							
TST15	025234	#74-3008							
TST16	026272	#75-3186							
TST17	026542	#76-3242							
TST2	021222	#61-2226							
TST20	027016	#77-3299							
TST21	027232	#79-3359							
TST22	027634	79-3435	#81-3462						
TST23	030232	81-3536	#83-3563						
TST24	030662	83-3645	#84-3657						
TST25	031326	84-3742	#85-3768						
TST26	031742	85-3845	#87-3858						
TST27	032352	87-3935	#89-3962						
TST3	021456	#62-2294							
TST30	032636	89-4005	#91-4033						
TST31	033042	91-4072	#93-4081						
TST32	033262	93-4122	#95-4139						
TST33	033332	#96-4169							
TST34	033632	96-4216	#98-4240						
TST35	034104	98-4280	#100-4300						
TST36	034174	100-4314	#101-4323						
TST37	034266	101-4337	#102-4346						
TST4	021632	#64-2354							
TST40	034360	102-4360	#103-4369						
TST41	034446	103-4382	#104-4392						
TST42	035344	104-4403	#106-4517						
TST43	035420	#107-4535							
TST44	035474	#108-4553							
TST45	035542	#109-4567							
TST5	022126	#65-2427							
TST6	022406	#66-2500							
TST7	022566	#67-2544							
TYPBN	= 104406	39-1838	#50-1910						
TYPDS	= 104405	39-1832	41-1894	#50-1910	110-4609	110-4609			
TYPE	= 104401	37-1787	37-1787	39-1789	39-1810	39-1812	39-1815	39-1817	39-1847	39-1861
		39-1867	39-1872	40-1882	40-1882	40-1882	40-1882	40-1882	40-1882	40-1882
		40-1882	40-1882	40-1882	40-1882	40-1882	40-1882	40-1882	40-1882	40-1882
		40-1882	41-1888	41-1892	43-1903	45-1905	46-1906	47-1907	#50-1910	51-1911
		58-2107	58-2108	110-4609	110-4609	110-4609				
TYPOC	= 104402	39-1797	39-1826	40-1882	41-1891	#50-1910				
TYPON	= 104404	#50-1910								
TYPOS	= 104403	#50-1910								
UDPAR0	= 177660	#15-1198	27-1566	*107-4539	*107-4544					
UDPAR1	= 177662	#15-1198								
UDPAR2	= 177664	#15-1198								

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES
UDPAR3	= 177666	#15-1198
UDPAR4	= 177670	#15-1198 *98-4241
UDPAR5	= 177672	#15-1198
UDPAR6	= 177674	#15-1198
UDPAR7	= 177676	#15-1198
UDPDR0	= 177620	#15-1198
UDPDR1	= 177622	#15-1198 *76-3243 *76-3254
UDPDR2	= 177624	#15-1198
UDPDR3	= 177626	#15-1198
UDPDR4	= 177630	#15-1198 *89-3967 *96-4175 *96-4215
UDPDR5	= 177632	#15-1198
UDPDR6	= 177634	#15-1198
UDPDR7	= 177636	#15-1198
UIPAR0	= 177640	#15-1198 27-1565 *68-2640 *68-2659 *108-4556 *108-4561
UIPAR1	= 177642	#15-1198
UIPAR2	= 177644	#15-1198
UIPAR3	= 177646	#15-1198 *60-2156 *71-2779
UIPAR4	= 177650	#15-1198 *60-2157 *71-2780 *81-3464 *84-3659
UIPAR5	= 177652	#15-1198
UIPAR6	= 177654	#15-1198
UIPAR7	= 177656	#15-1198
UIPDR0	= 177600	#15-1198 27-1545
UIPDR1	= 177602	#15-1198
UIPDR2	= 177604	#15-1198
UIPDR3	= 177606	#15-1198 *71-2781 *71-2832 *76-3255 *76-3289
UIPDR4	= 177610	#15-1198 *60-2165 *61-2236 *71-2782 *84-3658 *87-3866 *87-3934 *89-3968 *93-4083 *96-4176
UIPDR5	= 177612	#15-1198
UIPDR6	= 177614	#15-1198
UIPDR7	= 177616	#15-1198
USESTK	= 000600	#15-1213 58-2114 68-2644 68-2661 71-2800 81-3476 84-3673 87-3870 107-4540 108-4557
USP	=%000006	#15-1207 *58-2114 *68-2644 *68-2661 *71-2789 *71-2800 81-3471 *84-3664 84-3665 *84-3675 87-3870 *87-3872
VIRT1	001276	#20-1249
WASR6	001256	#20-1249 54-2003 54-2004 *56-2063 *57-2089
WASSR0	001264	#20-1249 *28-1594 28-1597 54-2004 54-2006 54-2008 54-2010 54-2013 54-2014 54-2016 54-2017 54-2021 54-2023 54-2026 *57-2090 *60-2183 60-2185 *61-2254 61-2256 *62-2309 *64-2380 64-2383 *64-2403 *65-2451 65-2454 *65-2474 *67-2576 67-2578 *67-2592 67-2593 *70-2743 70-2746 *71-2794 71-2813 *80-3440 *82-3541 *83-3649 *84-3746 *86-3850 *88-3940 *90-4008 *97-4219 *99-4283
WASSR1	001266	#20-1249 *28-1595 54-2023 54-2026 *90-4009 *97-4220 *99-4284
WASSR2	001270	#20-1249 *28-1596 54-2004 54-2007 54-2008 54-2011 54-2012 54-2013 54-2014 54-2016 54-2017 54-2023 54-2026 *57-2091 *60-2184 60-2192 *61-2255 61-2263 *62-2303 62-2312 62-2314 *64-2381 64-2390 *64-2404 *65-2452 65-2461 *65-2475 *66-2515 66-2516 *67-2549 67-2551 *67-2558 67-2560 *67-2577 67-2581 *67-2596 67-2597 *67-2607 67-2609 *70-2744 70-2750 *71-2795 71-2816 *80-3441 *82-3542 *83-3650 *84-3747 *86-3851 *88-3941 *90-4010 *97-4221 *99-4285
WASSR3	001272	#20-1249 *90-4011
WBIT	= 000100	#15-1209
$APTHD	000204	19-1248 #19-1248
$ASTAT	= ******	44-1904 44-1904

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
$ATYC	006022	44-1904	#44-1904							
$ATY1	005776	#44-1904								
$ATY3	006004	43-1903	#44-1904							
$ATY4	006014	37-1787	#44-1904							
$AUTOB	001134	#20-1249	40-1882	40-1882	40-1882	*58-2107				
$BDADR	001122	#20-1249								
$BDDAT	001126	#20-1249								
$BELL	001214	#20-1249	37-1787	38-1787	38-1787					
$BIN	006316	*45-1905	*45-1905	45-1905	#45-1905					
$CHARC	005772	*43-1903	*43-1903	43-1903	*43-1903	#43-1903				
$CKSWR	004306	#40-1882	50-1910	50-1910						
$CLR.T	036242	110-4609	#110-4609							
$CMTAG	001100	#20-1249	58-2106	58-2106	58-2106	58-2106	58-2106	58-2106		
$CM1	= 000006	#20-1249	20-1249	20-1249	#20-1249	20-1249	20-1249	#20-1249	20-1249	20-1249
		#20-1249	20-1249	20-1249	#20-1249	20-1249	20-1249	#20-1249	20-1249	20-1249
		#20-1249								
$CM2	= 000014	#20-1249	20-1249	20-1249	#20-1249	20-1249	20-1249	#20-1249	20-1249	20-1249
		#20-1249	20-1249	20-1249	#20-1249	20-1249	20-1249	#20-1249	20-1249	20-1249
		#20-1249								
$CM3	= 000006	#20-1249	20-1249	20-1249						
$CM4	= 000006	#20-1249	20-1249	20-1249	#20-1249	20-1249	20-1249	#20-1249	20-1249	20-1249
		#20-1249	20-1249	20-1249	#20-1249	20-1249	20-1249	#20-1249	20-1249	20-1249
		#20-1249								
$CNTLC	001312	#20-1249	40-1882	40-1882	40-1882	40-1882				
$CNTLG	005253	40-1882	#40-1882							
$CNTLU	005246	40-1882	40-1882	#40-1882						
$CPUOP	001252	#20-1249								
$CRLF	001221	#20-1249	37-1787	38-1787	38-1787	39-1789	39-1812	39-1817	39-1872	40-1882
		40-1882	40-1882	40-1882	43-1903	43-1903	43-1903	110-4609		
$DBLK	006762	47-1907	47-1907	#47-1907						
$DB20	007066	39-1844	39-1858	#49-1909						
$DEVCT	001234	#20-1249								
$DOAGN	036262	110-4609	110-4609	110-4609	#110-4609					
$DTBL	006752	47-1907	#47-1907							
$ENDAD	036252	18-1247	37-1787	58-2107	#110-4609					
$ENDCT	036054	58-2106	#110-4609							
$ENULL	036324	#110-4609								
$ENV	001244	#20-1249	37-1787	43-1903	44-1904	44-1904	58-2107			
$ENVM	001245	#20-1249	43-1903	43-1903	44-1904	58-2106				
$EOP	036024	41-1896	109-4602	#110-4609						
$EOPCT	036046	*58-2106	#110-4609	110-4609						
$ERFLG	001103	#20-1249	36-1785	*36-1785	36-1785	*36-1785	36-1785	36-1785	*37-1787	38-1787
		38-1787								
$ERMAX	001115	#20-1249	*36-1785	36-1785	36-1785	*58-2106				
$ERROR	003270	#37-1787	58-2106							
$ERRPC	001116	#20-1249	*37-1787	*37-1787	37-1787	38-1787	38-1787	39-1795	39-1880	54-2003
		54-2004	54-2005	54-2006	54-2007	54-2008	54-2009	54-2010	54-2011	54-2012
		54-2013	54-2014	54-2015	54-2016	54-2017	54-2018	54-2019	54-2020	54-2021
		54-2022	54-2024	54-2025	54-2026	54-2027	54-2028	54-2029	54-2030	54-2031
		54-2032								
$ERRTB	001320	#21-1249	39-1807							
$ERTTL	001112	#20-1249	*37-1787	38-1787	38-1787	110-4609	110-4609	110-4609	*110-4609	

SYMBOL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES								
$ESCAP	001212	#20-1249	*36-1785	37-1787	37-1787	38-1787	38-1787	*58-2106		
$ETABL	001244	#20-1249								
$ETEND	001254	19-1248	#20-1249							
$FATAL	001226	#20-1249	*44-1904							
$FFLG	006242	*44-1904	*44-1904	44-1904	*44-1904	#44-1904				
$FILLC	001156	#20-1249	43-1903	43-1903	43-1903					
$FILLS	001155	#20-1249	43-1903	43-1903						
$GDADR	001120	#20-1249								
$GDDAT	001124	#20-1249								
$GET42	036224	110-4609	110-4609	#110-4609						
$GTSWR	004356	#40-1882	50-1910	50-1910						
$HD	= 000000	15-1195	15-1195	15-1195						
$HIBTS	000204	#19-1248								
$ICNT	001104	#20-1249								
$ILLUP	007452	51-1911	51-1911	#51-1911						
$INTAG	001135	#20-1249	40-1882	40-1882	40-1882	40-1882				
$ITEMB	001114	#20-1249	*37-1787	37-1787	37-1787	*37-1787	37-1787	38-1787	38-1787	39-1792
$LF	001222	#20-1249	38-1787	38-1787	40-1882	40-1882	40-1882	43-1903	43-1903	
$LFLG	006241	*44-1904	#44-1904							
$LOOP	036320	110-4609	#110-4609							
$LPADR	001106	#20-1249	*36-1785	*36-1785	36-1785	36-1785	36-1785	*58-2106		
$LPERR	001110	#20-1249	36-1785	*36-1785	36-1785	36-1785	37-1787	*58-2106	*60-2166	*61-2237
		*62-2295	*64-2361	*64-2365	*64-2371	*65-2432	*65-2436	*65-2442	*66-2508	*67-2548
		*67-2557	*67-2566	*68-2634	*69-2699	*70-2735	*71-2787	*73-2862	*73-2886	*73-2895
		*73-2909	*73-2920	*73-2931	*73-2940	*73-2950	*73-2960	*73-2971	*73-2980	*73-2989
		*74-3022	*74-3036	*74-3047	*74-3086	*74-3098	*74-3107	*74-3124	*74-3134	*74-3148
		*74-3158	*74-3168	*75-3188	*75-3202	*75-3209	*75-3218	*75-3227	*76-3244	*76-3258
		*76-3265	*76-3274	*76-3282	*77-3301	*79-3368	*79-3384	*81-3468	*81-3485	*83-3583
		*84-3678	*85-3769	*85-3792	*87-3859	*87-3883	*89-3974	*91-4036	*93-4086	*96-4183
		*98-4248	*105-4412							
$MAIL	001224	19-1248	19-1248	#20-1249	36-1785	37-1787	43-1903	58-2106	58-2107	
$MBADR	000206	#19-1248								
$MFLG	006240	*44-1904	44-1904	*44-1904	#44-1904					
$MNEW	005271	40-1882	#40-1882							
$MSGAD	001240	#20-1249	*44-1904	44-1904						
$MSGLG	001242	#20-1249	*44-1904							
$MSGTY	001224	#20-1249	44-1904	*44-1904	44-1904	*44-1904				
$MSWR	005260	40-1882	#40-1882							
$MXCNT	001306	#20-1249								
$NULL	001154	#20-1249	43-1903	43-1903	43-1903					
$NWTST	= 000001	#59-2150	59-2150	#60-2150	60-2150	#60-2226	60-2226	#61-2226	61-2226	#61-2294
		61-2294	#62-2294	62-2294	#63-2354	63-2354	#64-2354	64-2354	#64-2427	64-2427
		#65-2427	65-2427	#65-2500	65-2500	#66-2500	66-2500	#66-2544	66-2544	#67-2544
		67-2544	#67-2632	67-2632	#68-2632	68-2632	#68-2697	68-2697	#69-2697	69-2697
		#69-2729	69-2729	#70-2729	70-2729	#70-2777	70-2777	#71-2777	71-2777	#72-2852
		72-2852	#73-2852	73-2852	#73-3008	73-3008	#74-3008	74-3008	#74-3186	74-3186
		#75-3186	75-3186	#75-3242	75-3242	#76-3242	76-3242	#76-3299	76-3299	#77-3299
		77-3299	#78-3359	78-3359	#79-3359	79-3359	#80-3462	80-3462	#81-3462	81-3462
		#82-3563	82-3563	#83-3563	83-3563	#83-3657	83-3657	#84-3657	84-3657	#84-3768
		84-3768	#85-3768	85-3768	#86-3858	86-3858	#87-3858	87-3858	#88-3962	88-3962
		#89-3962	89-3962	#90-4033	90-4033	#91-4033	91-4033	#92-4081	92-4081	#93-4081
		93-4081	#94-4139	94-4139	#95-4139	95-4139	#95-4169	95-4169	#96-4169	96-4169

SYMBOL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES								
		#97-4240	97-4240	#98-4240	98-4240	#99-4300	99-4300	#100-4300	100-4300	#100-4323
		100-4323	#101-4323	101-4323	#101-4346	101-4346	#102-4346	102-4346	#102-4369	102-4369
		#103-4369	103-4369	#103-4392	103-4392	#104-4392	104-4392	#105-4517	105-4517	#106-4517
		106-4517	#106-4535	106-4535	#107-4535	107-4535	#107-4553	107-4553	#108-4553	108-4553
		#108-4567	108-4567	#109-4567	109-4567					
$OCNT	006542	*46-1906	*46-1906	#46-1906						
$OCTVL	007170	49-1909	#49-1909							
$OMODE	006544	*46-1906	*46-1906	46-1906	*46-1906	*46-1906	#46-1906			
$OVER	003252	36-1785	36-1785	36-1785	#36-1785					
$PASS	001232	#20-1249	41-1886	*58-2106	*110-4609	*110-4609	110-4609	110-4609	110-4609	110-4609
$PASTM	000212	#19-1248								
$PWRAD	007434	#51-1911								
$PWRDN	007274	#51-1911	51-1911	58-2106						
$PWRMG	007430	#51-1911								
$PWRUP	007346	51-1911	#51-1911							
$QUES	001220	#20-1249	38-1787	38-1787	40-1882	40-1882	40-1882	40-1882	43-1903	43-1903
$RDCHR	004630	#40-1882	50-1910	50-1910						
$RDDEC	= ******	50-1910								
$RDLIN	004760	#40-1882	50-1910	50-1910						
$RDOCT	= ******	50-1910								
$RDSZ	= 000010	#40-1882	40-1882							
$REGAD	001160	#20-1249								
$REG0	001162	#20-1249	*37-1787	54-2008	54-2009	54-2010	54-2011	54-2016	54-2022	54-2024
		54-2025	54-2028	54-2031						
$REG1	001164	#20-1249	*37-1787	54-2012	54-2015	54-2016	54-2017	54-2020	54-2021	54-2024
		54-2025								
$REG2	001166	#20-1249	*37-1787	54-2005	54-2006	54-2007	54-2010	54-2015	54-2020	54-2025
$REG3	001170	#20-1249	*37-1787	54-2006	54-2011	54-2017	54-2020	54-2027		
$REG4	001172	#20-1249	*37-1787	54-2007	54-2008	54-2009	54-2010	54-2011		
$REG5	001174	#20-1249	*37-1787							
$RESRE	007030	#48-1908	50-1910							
$RTNAD	036322	#110-4609								
$RTRN	036316	58-2106	*58-2106	*58-2106	110-4609	#110-4609				
$R2A	= ******	50-1910								
$SAVRE	006772	#48-1908	50-1910	50-1910						
$SAVR6	007456	*51-1911	51-1911	*51-1911	*51-1911	#51-1911				
$SCOPE	003042	#36-1785	58-2106							
$SETUP	= 000137	#16-1245	16-1245	#16-1245	16-1245	#16-1245	16-1245	#16-1245	16-1245	#16-1245
		16-1245	#16-1245	16-1245	#16-1245	36-1785	37-1787	37-1787	37-1787	37-1787
		40-1882	40-1882	51-1911	58-2106	58-2106	58-2106	58-2106	58-2106	58-2106
		58-2106	58-2106	58-2106	58-2106	58-2106	58-2106	58-2107	58-2107	58-2107
		110-4609	110-4609							
$STUP	= 177777	#16-1245	#16-1245	16-1245	#16-1245	#16-1245	16-1245	#16-1245	#16-1245	16-1245
		#16-1245	#16-1245	16-1245	#16-1245	#16-1245	16-1245	#16-1245	#16-1245	16-1245
$SVLAD	003216	36-1785	36-1785	#36-1785						
$SVPC	= 000204	#18-1247	18-1247							
$SWR	= 173400	#15-1183	15-1195	15-1196	15-1196	15-1196	15-1196	15-1196	15-1196	15-1196
		15-1196	20-1249	20-1249	20-1249	36-1785	36-1785	36-1785	36-1785	36-1785
		36-1785	36-1785	36-1785	36-1785	36-1785	36-1785	36-1785	36-1785	36-1785
		36-1785	36-1785	36-1785	36-1785	36-1785	36-1785	37-1787	37-1787	37-1787
		37-1787	37-1787	37-1787	37-1787	37-1787	37-1787	38-1787	38-1787	51-1911
		58-2106	58-2106	58-2106	58-2106	58-2106	60-2150	61-2226	62-2294	64-2354

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
		65-2427	66-2500	67-2544	68-2632	69-2697	70-2729	71-2777	73-2852	74-3008
		75-3186	76-3242	77-3299	79-3359	81-3462	83-3563	84-3657	85-3768	87-3858
		89-3962	91-4033	93-4081	95-4139	96-4169	98-4240	100-4300	101-4323	102-4346
		103-4369	104-4392	106-4517	107-4535	108-4553	109-4567	110-4609	110-4609	110-4609
		110-4609	110-4609							
$SWREG	001246	#20-1249	58-2106							
$SWRMK	= 000000	15-1196	15-1196	15-1196	15-1196	15-1196	15-1196	15-1196	15-1196	15-1196
		36-1785	36-1785	36-1785	36-1785	36-1785	36-1785	36-1785	36-1785	36-1785
		36-1785								
$TBIT	001310	#20-1249	*51-1911	*58-2106	*110-4609	110-4609	110-4609			
$TESTN	001230	#20-1249	*36-1785	39-1880						
$TKB	001146	#20-1249	40-1882	40-1882	40-1882	40-1882	40-1882	40-1882	43-1903	43-1903
		43-1903	43-1903							
$TKS	001144	#20-1249	40-1882	40-1882	40-1882	40-1882	40-1882	40-1882	40-1882	40-1882
		43-1903	43-1903	43-1903	43-1903					
$TMP0	001176	#20-1249	*32-1681	32-1693	32-1700	54-2005	54-2006	54-2007	54-2013	54-2018
		54-2029	54-2031	54-2032	*60-2169	60-2199	60-2204	*61-2240	61-2270	61-2275
		*67-2571	67-2578	*71-2802	*71-2808	*71-2812	*71-2815	*71-2819	71-2820	*105-4483
		*109-4582								
$TMP1	001200	#20-1249	*32-1682	32-1694	32-1701	54-2032	*67-2568	*67-2580	*67-2583	67-2584
		*67-2590	*67-2595	*67-2599	67-2600	*70-2742	*70-2748	*70-2752	70-2753	*109-4578
		109-4585								
$TMP2	001202	#20-1249	54-2013	*67-2572	67-2581	*79-3414	79-3415	*79-3428	79-3429	*81-3515
		81-3516	*81-3529	81-3530	83-3620	*83-3621	83-3625	*83-3638	83-3639	83-3642
		84-3716	*84-3717	84-3721	*84-3734	84-3736	84-3739	*85-3822	85-3823	*85-3836
		85-3837	*87-3913	87-3914	*87-3927	87-3928	*109-4590			
$TMP3	001204	#20-1249	*100-4303	100-4307	*101-4326	101-4330	*102-4349	102-4353	*103-4372	103-4376
		*109-4587	109-4594							
$TMP4	001206	#20-1249	*39-1855	39-1857	*109-4588	109-4593				
$TMP5	001210	#20-1249	*39-1856	*41-1886	*41-1887	41-1893	*109-4589	109-4592		
$TN	= 000046	#15-1184	15-1195	59-2150	60-2150	#60-2150	60-2226	61-2226	#61-2226	61-2294
		62-2294	#62-2294	63-2354	64-2354	#64-2354	64-2427	65-2427	#65-2427	65-2500
		66-2500	#66-2500	66-2544	67-2544	#67-2544	67-2632	68-2632	#68-2632	68-2697
		69-2697	#69-2697	69-2715	69-2729	70-2729	#70-2729	70-2777	71-2777	#71-2777
		72-2852	73-2852	#73-2852	73-3008	74-3008	#74-3008	74-3186	75-3186	#75-3186
		75-3242	76-3242	#76-3242	76-3299	77-3299	#77-3299	78-3359	79-3359	#79-3359
		79-3435	80-3462	81-3462	#81-3462	81-3536	82-3563	83-3563	#83-3563	83-3645
		83-3657	84-3657	#84-3657	84-3742	84-3768	85-3768	#85-3768	85-3845	86-3858
		87-3858	#87-3858	87-3935	88-3962	89-3962	#89-3962	89-4005	90-4033	91-4033
		#91-4033	91-4072	92-4081	93-4081	#93-4081	93-4122	94-4139	95-4139	#95-4139
		95-4169	96-4169	#96-4169	96-4216	97-4240	98-4240	#98-4240	98-4280	99-4300
		100-4300	#100-4300	100-4314	100-4323	101-4323	#101-4323	101-4337	101-4346	102-4346
		#102-4346	102-4360	102-4369	103-4369	#103-4369	103-4382	103-4392	104-4392	#104-4392
		105-4517	106-4517	#106-4517	106-4535	107-4535	#107-4535	107-4553	108-4553	#108-4553
		108-4567	109-4567	#109-4567						
$TPB	001152	#20-1249	43-1903	43-1903	43-1903					
$TPFLG	001157	#20-1249	43-1903	43-1903	43-1903					
$TPS	001150	#20-1249	43-1903	43-1903	43-1903					
$TRAP	007206	#50-1910	58-2106							
$TRAP2	007230	#50-1910	50-1910							
$TRP	= 000015	#50-1910	50-1910	50-1910	50-1910	50-1910	#50-1910	50-1910	50-1910	50-1910
		50-1910	#50-1910	50-1910	50-1910	50-1910	50-1910	#50-1910	50-1910	50-1910

SYMBOL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES								
		50-1910	50-1910	#50-1910	50-1910	50-1910	50-1910	50-1910	#50-1910	50-1910
		50-1910	50-1910	50-1910	#50-1910	50-1910	50-1910	50-1910	50-1910	#50-1910
		50-1910	50-1910	50-1910	50-1910	#50-1910	50-1910	50-1910	50-1910	50-1910
		#50-1910	50-1910	50-1910	50-1910	50-1910	#50-1910	50-1910	50-1910	50-1910
		50-1910	#50-1910	50-1910	50-1910	50-1910	50-1910	#50-1910		
$TRPAD	007242	50-1910	#50-1910							
$TSTM	000210	#19-1248								
$TSTNM	001102	#20-1249	36-1785	36-1785	*36-1785	36-1785	36-1785	36-1785	36-1785	37-1787
		37-1787	38-1787	38-1787	41-1889	*110-4609				
$TTYIN	005236	40-1882	40-1882	40-1882	40-1882	40-1882	#40-1882			
$TYPBN	006244	#45-1905	50-1910	50-1910						
$TYPDS	006546	#47-1907	50-1910	50-1910						
$TYPE	005432	#43-1903	44-1904	50-1910	50-1910					
$TYPEC	005644	40-1882	43-1903	43-1903	43-1903	#43-1903				
$TYPEX	005774	43-1903	43-1903	43-1903	#43-1903					
$TYPOC	006344	#46-1906	50-1910	50-1910						
$TYPON	006360	46-1906	#46-1906	50-1910						
$TYPOS	006320	#46-1906	50-1910							
$UNIT	001236	#20-1249								
$UNITM	000214	#19-1248								
$USWR	001250	*20-1249								
$XOFF	= 000023	43-1903	43-1903							
$XON	= 000021	40-1882	43-1903	43-1903	43-1903					
$XTSTR	003054	#36-1785								
$$GET4	= 000001	#110-4609	#110-4609	110-4609						
$OFILL	006543	*46-1906	*46-1906	46-1906	#46-1906					
$40CAT	= ******	36-1785	37-1787							
.$ASTA	= ******	44-1904	44-1904							
.$X	= 000204	#19-1248	19-1248							

MACRO CROSS REFERENCE

MACRO NAME	REFERENCES									
COMMEN	#15-1197									
ENDCOM	#15-1197									
ESCAPE	#15-1197									
GETPRI	#15-1197	110-4609								
GETSWR	#15-1197	#58-2107	58-2107							
MES350	#99-4293	100-4300								
MES351	#100-4316	#101-4323								
MES352	#101-4339	102-4346								
MES353	#102-4362	#103-4369								
MES354	#103-4384	#104-4392								
MES355	#105-4509	106-4517								
MES356	#106-4527	#107-4535								
MES357	#107-4545	#108-4553	#108-4562	#109-4567						
MSG30	#59-2142	60-2150								
MSG31	#60-2218	#61-2226								
MSG31A	#61-2283	62-2294								
MSG32	#63-2343	#64-2354								
MSG33	#64-2416	65-2427								
MSG34	#65-2487	66-2500								
MSG35	#66-2531	#67-2544								
MSG36	#67-2619	68-2632								
MSG36A	#68-2691	69-2697								
MSG37	#69-2718	#70-2729								
MSG40	#70-2763	#71-2777								
MSG40A	#76-3291	#77-3299								
MSG40B	#72-2837	73-2852								
MSG40C	#73-2998	#74-3008								
MSG40D	#74-3178	#75-3186								
MSG40E	#75-3234	76-3242								
MSG41	#78-3345	#79-3359								
MSG41A	#80-3448	81-3462								
MSG42	#82-3549	#83-3563	#84-3657							
MSG43	#84-3754	#85-3768	#87-3858							
MSG43A	#88-3948	89-3962								
MSG44	#90-4019	#91-4033	#93-4081							
MSG45	#94-4131	95-4139								
MSG46	#95-4156	#96-4169								
MSG47	#97-4229	98-4240								
MULT	#15-1197									
NEWTST	#15-1178	#15-1197	#59-2150	#60-2226	#61-2294	#63-2354	#64-2427	#65-2500	#66-2544	#67-2632
	#68-2697	#69-2729	#70-2777	#72-2852	#73-3008	#74-3186	#75-3242	#76-3299	#78-3359	#80-3462
	#82-3563	#83-3657	#84-3768	#86-3858	#88-3962	#90-4033	#92-4081	#94-4139	#95-4169	#97-4240
	#99-4300	#100-4323	#101-4346	#102-4369	#103-4392	#105-4517	#106-4535	#107-4553	#108-4567	
POP	#15-1197	#44-1904	#44-1904	#47-1907	#48-1908	#51-1911	#51-1911			
PUSH	#15-1197	44-1904	44-1904	44-1904	47-1907	48-1908	51-1911	51-1911		
REPORT	#15-1197									
SAVR	#15-1166	#37-1787								
SETPRI	#15-1197									
SETTRA	#50-1910	50-1910	50-1910	50-1910	50-1910	50-1910	50-1910	50-1910	50-1910	50-1910
	50-1910	50-1910	50-1910							
SETUP	#15-1197	58-2106								
SKIP	#15-1197	69-2715	79-3435	81-3536	83-3645	84-3742	85-3845	87-3935	89-4005	91-4072

MACRO CROSS REFERENCE

MACRO NAME	REFERENCES									
	93-4122	96-4216	98-4280	100-4314	101-4337	102-4360	103-4382			
SLASH	#15-1197									
SPACE	#15-1163	#15-1197								
STARS	#15-1197	18-1247	19-1248	19-1248	19-1248	20-1249	20-1249	20-1249	27-1526	27-1531
	28-1569	28-1580	29-1609	29-1616	30-1627	30-1633	31-1642	31-1653	32-1660	32-1675
	33-1706	33-1721	34-1749	34-1764	36-1785	37-1787	40-1882	40-1882	40-1882	40-1882
	43-1903	44-1904	45-1905	46-1906	47-1907	48-1908	49-1909	50-1910	51-1911	51-1911
	56-2045	56-2052	57-2071	57-2078	59-2130	59-2140	60-2150	60-2150	61-2226	61-2226
	62-2294	62-2294	63-2327	63-2342	64-2354	64-2354	65-2427	65-2427	66-2500	66-2500
	67-2544	67-2544	68-2632	68-2632	69-2697	69-2697	70-2729	70-2729	71-2777	71-2777
	73-2852	73-2852	74-3008	74-3008	75-3186	75-3186	76-3242	76-3242	77-3299	77-3299
	78-3338	78-3343	79-3359	79-3359	81-3462	81-3462	83-3563	83-3563	84-3657	84-3657
	85-3768	85-3768	87-3858	87-3858	89-3962	89-3962	91-4033	91-4033	93-4081	93-4081
	95-4139	95-4139	96-4169	96-4169	98-4240	98-4240	100-4300	100-4300	100-4300	100-4300
	101-4323	101-4323	101-4323	101-4323	102-4346	102-4346	102-4346	102-4346	103-4369	103-4369
	103-4369	103-4369	104-4392	104-4392	104-4392	104-4392	106-4517	106-4517	106-4517	106-4517
	107-4535	107-4535	107-4535	107-4535	108-4553	108-4553	108-4553	108-4553	109-4567	109-4567
	110-4609									
SWRSU	#15-1197	#58-2106	#58-2106							
TRMTRP	#50-1910									
TYPBIN	#15-1197									
TYPDEC	#15-1197	110-4609	110-4609							
TYPNAM	#15-1197	#58-2107								
TYPNUM	#15-1197									
TYPOCS	#15-1197									
TYPOCT	#15-1197	40-1882								
TYPTXT	#15-1197	58-2108	110-4609	110-4609						
USER	#15-1223	20-1249								
$$CMRE	#19-1249	20-1249	20-1249	20-1249	20-1249	20-1249	20-1249			
$$CMTM	#19-1249	20-1249	20-1249	20-1249	20-1249	20-1249	20-1249			
$$ESCA	#15-1197									
$$NEWT	#15-1178	#15-1197	#59-2150	#60-2226	#61-2294	#63-2354	#64-2427	#65-2500	#66-2544	#67-2632
	#68-2697	#69-2729	#70-2777	#72-2852	#73-3008	#74-3186	#75-3242	#76-3299	#78-3359	#80-3462
	#82-3563	#83-3657	#84-3768	#86-3858	#88-3962	#90-4033	#92-4081	#94-4139	#95-4169	#97-4240
	#99-4300	#100-4323	#101-4346	#102-4369	#103-4392	#105-4517	#106-4535	#107-4553	#108-4567	
$$SET	#50-1910	#50-1910	#50-1910	#50-1910	#50-1910	#50-1910	#50-1910	#50-1910	#50-1910	#50-1910
	#50-1910	#50-1910	#50-1910							
$$SETM	#58-2106	58-2106								
$$SETU	#58-2106	58-2106								
$$SKIP	#15-1197	69-2715	79-3435	81-3536	83-3645	84-3742	85-3845	87-3935	89-4005	91-4072
	93-4122	96-4216	98-4280	100-4314	101-4337	102-4360	103-4382			
.EQUAT	#15-1178	15-1197								
.EQUIV	#15-1160	#15-1177	#15-1197	#15-1197	#15-1197	#15-1197	#15-1197	#15-1197	#15-1197	#15-1197
	#15-1197	#15-1197	#15-1197	#15-1197	#15-1197	#15-1197	#15-1197	#15-1197	#15-1197	#15-1197
	#15-1197	#15-1197	#15-1197	#15-1197	#15-1197	#15-1201	#15-1202	#15-1203	#15-1204	#15-1205
	#15-1206	#15-1207	#15-1208	#15-1209						
.HEADE	#15-1178	15-1195								
.KT11	#15-1178	15-1198								
.SETUP	#15-1178	16-1245								
.SWRHI	#15-1178	15-1196								
.SWRLO	#15-1196									
.$ACT1	#15-1179	#17-1247								

MACRO CROSS REFERENCE

MACRO NAME	REFERENCES	
.$APTB	#20-1249	#20-1249
.$APTH	#15-1181	#18-1248
.$APTY	#15-1181	43-1904
.$CATC	#15-1179	16-1246
.$CMTA	#15-1179	19-1249
.$DB20	#15-1182	#48-1909
.$EOP	#15-584	110-4609
.$ERRO	#15-771	37-1787
.$POWE	#15-1180	50-1911
.$READ	#15-1181	#40-1882
.$SAVE	#15-1182	#47-1908
.$SCOP	#15-950	#36-1785
.$TRAP	#15-1180	49-1910
.$TYPB	#15-1181	#44-1905
.$TYPD	#15-1180	#46-1907
.$TYPE	#15-1179	#42-1903
.$TYPO	#15-1180	45-1906