K 6

```
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
```

FILEID**MCHECKS

```
MM      MM    CCCCCCCC  HH      HH  EEEEEEEEEE    CCCCCCCC  KK      KK    SSSSSSSS
MM      MM    CCCCCCCC  HH      HH  EEEEEEEEEE    CCCCCCCC  KK      KK    SSSSSSSS
MMMM  MMMM  CC          HH      HH  EE          CC          KK      KK  SS
MMMM  MMMM  CC          HH      HH  EE          CC          KK      KK  SS
MM  MM  MM  CC          HH      HH  EE          CC          KK    KK    SS
MM  MM  MM  CC          HH      HH  EE          CC          KK    KK    SS
MM      MM  CC          HHHHHHHHHH  EEEEEEEE    CC          KKKKKK        SSSSSS
MM      MM  CC          HHHHHHHHHH  EEEEEEEE    CC          KKKKKK        SSSSSS
MM      MM  CC          HH      HH  EE          CC          KK    KK            SS
MM      MM  CC          HH      HH  EE          CC          KK    KK            SS
MM      MM  CC          HH      HH  EE          CC          KK      KK          SS
MM      MM  CC          HH      HH  EE          CC          KK      KK          SS    ....
MM      MM    CCCCCCCC  HH      HH  EEEEEEEEEE    CCCCCCCC  KK      KK  SSSSSSSS      ....
MM      MM    CCCCCCCC  HH      HH  EEEEEEEEEE    CCCCCCCC  KK      KK  SSSSSSSS      ....

LL          IIIIII      SSSSSSSS
LL          IIIIII      SSSSSSSS
LL            II      SS
LL            II      SS
LL            II      SS
LL            II      SS
LL            II        SSSSSS
LL            II        SSSSSS
LL            II              SS
LL            II              SS
LL            II              SS
LL            II              SS
LLLLLLLLLL  IIIIII    SSSSSSSS
LLLLLLLLLL  IIIIII    SSSSSSSS
```

```
C
C Version:       'V04-000'
C
C****************************************************************************
C*                                                                          *
C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
C*  ALL RIGHTS RESERVED.                                                    *
C*                                                                          *
C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
C*  TRANSFERRED.                                                            *
C*                                                                          *
C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
C*  CORPORATION.                                                            *
C*                                                                          *
C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
C*                                                                          *
C*                                                                          *
C****************************************************************************
C

c       author  Brian Porter              Creation Date   11-MAY-1981

c++
c       Function description:
c
c       This module dispatches to the correct machine check display routine
c       based on emb$l_hd_sid.
c
c       Modified by:
c
C       V03-007 SAR0249         Sharon A. Reynolds        11-Apr-1984
C               - Changed carriage control in format statements for
C               use with ERF.
C               - Removed the MCHECKS routine, re-written and named
C               MCHK_DISPATCHER.
C
c       V03-006 EAD0007         Elliott A. Drayton        21-Feb-1984
c               Added UVAX-1 support.
c
C       V03-005 SAR0082         Sharon A. Reynolds,       20-Jun-1983
C               Changed the carriage control in the 'format' statements
C               for use with ERF.
C
c       v03-004 BP0004          Brian Porter,             28-MAR-1983
c               Corrected 11/750 TB parity error stuff.
c
c       v03-003 BP0003          Brian Porter,             11-JAN-1982
c               Corrected mcheck type 7, ms7zz code.
c
c       v03-002 BP0002          Brian Porter,             27-AUG-1981
```

```
c               Corrected mcheck_780, SBITA register.
c
c       v03-001 BP0001          Brian Porter,           29-JUN-1981
c               Added call to LOGGER.
c**
c--

c++
c       Functional description:
c
c       This routine displays mchecks for the 11/780.  The format of the
c       cpu specific piece of the error log is as follows.
c       The summary parameter is divided up into the following pieces
c       bytes 1 & 2; summary code.  Byte 3; the opcode of the failing
c       instruction if the machine check is a control store parity error
c       abort.  Byte 4, bit 0; byte 4, bit 1; if the machine check is either
c       a cp or ib cache parity error fault or abort then each bit represents
c       the cache group if any that was disabled by vms as a result of the
c       machine check.
c
c       +-----------------------------------+
c       :        summary parameter          :
c       +-----------------------------------+
c       :    cpu error status register      :
c       +-----------------------------------+
c       :           trapped uPC             :
c       +-----------------------------------+
c       :             VA/VIBA               :
c       +-----------------------------------+
c       :              D reg                :
c       +-----------------------------------+
c       :              TBER0                :
c       +-----------------------------------+
c       :              TBER1                :
c       +-----------------------------------+
c       :              SBITA                :
c       +-----------------------------------+
c       :   cache parity error register     :
c       +-----------------------------------+
c       :              SBIER                :
c       +-----------------------------------+
c       :          exception PC             :
c       +-----------------------------------+
c       :          exception PSL            :
c       +-----------------------------------+
c--



        subroutine mcheck_780 (lun,start_of_cpu_specific_stuff)


        include 'src$:opcodes.for /nolist'
        include 'src$:modes.for /nolist'
```

```
        byte            lun
        integer*4       start_of_cpu_specific_stuff
        integer*4       buffer(12)
        integer*4       summary_code
        byte            summary_array(4)
        integer*2       summary_parameter
        byte            cs_parity_error_opcode
        byte            disabled_cache_group_flags
        integer*4       cpu_error_status_register
        integer*4       trapped_upc
        integer*4       va_viba_register
        integer*4       d_register
        integer*4       tber0_register
        integer*4       tber1_register
        integer*4       sbi_timeout_address_register
        integer*4       cache_parity_error_register
        integer*4       sbi_error_register
        integer*4       exception_pc
        integer*4       exception_psl
        equivalence     (buffer(1),summary_array)
        equivalence     (buffer(1),summary_code)
        equivalence     (summary_array(1),summary_parameter)
        equivalence     (summary_array(3),cs_parity_error_opcode)
        equivalence     (summary_array(4),disabled_cache_group_flags)
        equivalence     (buffer(2),cpu_error_status_register)
        equivalence     (buffer(3),trapped_upc)
        equivalence     (buffer(4),va_viba_register)
```

```
        equivalence     (buffer(5),d_register)
        equivalence     (buffer(6),tber0_register)
        equivalence     (buffer(7),tber1_register)
        equivalence     (buffer(8),sbi_timeout_address_register)
        equivalence     (buffer(9),cache_parity_error_register)
        equivalence     (buffer(10),sbi_error_register)
        equivalence     (buffer(11),exception_pc)
        equivalence     (buffer(12),exception_psl)
        character*27    v1cpu_error_status_register(3:3)
        data            v1cpu_error_status_register(3)
        1 /'PERFORMANCE MONITOR ENABLE*'/
        character*20    traps(1:7)
        data            traps(1)
        1 /'INTEGER OVERFLOW*'/
        data            traps(2)
        1 /'INTEGER DIVIDE BY 0*'/
        data            traps(3)
        1 /'FLOAT OVERFLOW*'/
        data            traps(4)
        1 /'FLOAT DIVIDE BY 0*'/
        data            traps(5)
        1 /'FLOAT UNDERFLOW*'/
        data            traps(6)
        1 /'DECIMAL OVERFLOW*'/
        data            traps(7)
        1 /'DECIMAL DIVIDE BY 0*'/
        character*24    v2cpu_error_status_register(7:16)
        data            v2cpu_error_status_register(7)
        1 /'ALU C31*'/
        data            v2cpu_error_status_register(8)
        1 /'ALU Z*'/
        data            v2cpu_error_status_register(9)
        1 /'ALU N*'/
        data            v2cpu_error_status_register(10)
        1 /'E ALU Z*'/
```

```
        data            v2cpu_error_status_register(11)
        1 /'E ALU N*'/

        data            v2cpu_error_status_register(12)
        1 /'CS PARITY ERROR GROUP 0*'/

        data            v2cpu_error_status_register(13)
        1 /'CS PARITY ERROR GROUP 1*'/

        data            v2cpu_error_status_register(14)
        1 /'CS PARITY ERROR GROUP 2*'/

        data            v2cpu_error_status_register(15)
        1 /'CS PARITY ERROR SUMMARY*'/

        data            v2cpu_error_status_register(16)
        1 /'NESTED ERROR*'/

        character*25    v1tber0_register(0:0)

        data            v1tber0_register(0)
        1 /'ENABLE MEMORY MANAGEMENT*'/

        character*15    v2tber0_register(6:9)

        data            v2tber0_register(6)
        1 /'TB HIT GROUP 0*'/

        data            v2tber0_register(7)
        1 /'TB HIT GROUP 1*'/

        data            v2tber0_register(8)
        1 /'IB AUTO RELOAD*'/

        data            v2tber0_register(9)
        1 /'IB ''WCHK''*'/

        character*17    v3tber0_register(14:15)

        data            v3tber0_register(14)
        1 /'MICRO CODE ''ADS''*'/

        data            v3tber0_register(15)
        1 /'MICRO CODE ''FS''*'/

        character*34    v1tber1_register(0:3)

        data            v1tber1_register(0)
        1 /'AUTOMATIC HARDWARE INITIATED LOAD*'/

        data            v1tber1_register(1)
        1 /'PROTECTION VIOLATION OR MISS*'/

        data            v1tber1_register(2)
        1 /'TB PARITY ERROR*'/
```

```
        data            v1tber1_register(3)
        1 /'TB MISS ON LOAD*'/

        character*24    v2tber1_register(8:20)

        data            v2tber1_register(8)
        1 /'CP TB PARITY ERROR*'/

        data            v2tber1_register(9)
        1 /'GROUP 0, ADDRESS BYTE 0*'/

        data            v2tber1_register(10)
        1 /'GROUP 0, ADDRESS BYTE 1*'/

        data            v2tber1_register(11)
        1 /'GROUP 0, ADDRESS BYTE 2*'/

        data            v2tber1_register(12)
        1 /'GROUP 1, ADDRESS BYTE 0*'/

        data            v2tber1_register(13)
        1 /'GROUP 1, ADDRESS BYTE 1*'/

        data            v2tber1_register(14)
        1 /'GROUP 1, ADDRESS BYTE 2*'/

        data            v2tber1_register(15)
        1 /'GROUP 0, DATA BYTE 0*'/

        data            v2tber1_register(16)
        1 /'GROUP 0, DATA BYTE 1*'/

        data            v2tber1_register(17)
        1 /'GROUP 0, DATA BYTE 2*'/

        data            v2tber1_register(18)
        1 /'GROUP 1, DATA BYTE 0*'/

        data            v2tber1_register(19)
        1 /'GROUP 1, DATA BYTE 1*'/

        data            v2tber1_register(20)
        1 /'GROUP 1, DATA BYTE 2*'/

        character*32    v1cache_parity_error_register(0:15)

        data            v1cache_parity_error_register(0)
        1 /'PARITY OK "CAM" GROUP 1, BYTE 2*'/

        data            v1cache_parity_error_register(1)
        1 /'PARITY OK "CAM" GROUP 1, BYTE 1*'/

        data            v1cache_parity_error_register(2)
        1 /'PARITY OK "CAM" GROUP 1, BYTE 0*'/

        data            v1cache_parity_error_register(3)
        1 /'PARITY OK "CAM" GROUP 0, BYTE 2*'/
```

```
        data            v1cache_parity_error_register(4)
        1 /'PARITY OK ''CAM'' GROUP 0, BYTE 1*'/

        data            v1cache_parity_error_register(5)
        1 /'PARITY OK ''CAM'' GROUP 0, BYTE 0*'/

        data            v1cache_parity_error_register(6)
        1 /'PARITY OK ''CDM'' GROUP 0, BYTE 3*'/

        data            v1cache_parity_error_register(7)
        1 /'PARITY OK ''CDM'' GROUP 0, BYTE 2*'/

        data            v1cache_parity_error_register(8)
        1 /'PARITY OK ''CDM'' GROUP 0, BYTE 1*'/

        data            v1cache_parity_error_register(9)
        1 /'PARITY OK ''CDM'' GROUP 0, BYTE 0*'/

        data            v1cache_parity_error_register(10)
        1 /'PARITY OK ''CDM'' GROUP 1, BYTE 3*'/

        data            v1cache_parity_error_register(11)
        1 /'PARITY OK ''CDM'' GROUP 1, BYTE 2*'/

        data            v1cache_parity_error_register(12)
        1 /'PARITY OK ''CDM'' GROUP 1, BYTE 1*'/

        data            v1cache_parity_error_register(13)
        1 /'PARITY OK ''CDM'' GROUP 1, BYTE 0*'/

        data            v1cache_parity_error_register(14)
        1 /'CP ERROR*'/

        data            v1cache_parity_error_register(15)
        1 /'ANY ERROR*'/

        integer*4       cache_group_disabled

        integer*4       mnemonic

        integer*4       ast_level

        integer*4       arithmetic_trap_code

        integer*4       last_tb_write_pulse

        equivalence     (cache_group_disabled,mnemonic,ast_level,
        1 arithmetic_trap_code,last_tb_write_pulse)

        logical*1       diagnostic_mode

        integer*4       lib$extzv

        integer*4       compress4

        integer*4       compressc
```

```
        call movc3 (%val(48),start_of_cpu_specific_stuff,buffer)

        diagnostic_mode = .false.

        if (iand(tber0_register,'001f001e'x) .ne. 0) diagnostic_mode = .true.

        call linchk (lun,2)

        write(lun,10) 'EXCEPTION PC',exception_pc
10      format(/' ',t8,a,t24,z8.8)

        if (.not. diagnostic_mode) then

        call vaxpsl (lun,exception_psl)
        else

        call linchk (lun,1)

        write(lun,10) 'ERROR PSL',exception_psl
        endif

        call linchk (lun,1)

        write(lun,10) 'SUMMARY CODE',summary_code

        if (.not. diagnostic_mode) then

        call linchk (lun,1)

        if (summary_parameter .eq. '00'x) then

        call linchk (lun,1)

        write(lun,15) 'CP READ TIMEOUT'
15      format(' ',t40,a,/,' ',t40,' - OR ERROR CONFIRMATION FAULT')

        else if (summary_parameter .eq. '02'x) then

        write(lun,20) 'CP TB'
20      format(' ',t40,a,' PARITY ERROR FAULT')

        else if (summary_parameter .eq. '03'x) then

        write(lun,20) 'CP CACHE'

        else if (summary_parameter .eq. '05'x) then

        write(lun,25) 'CP'
25      format(' ',t40,a,' READ DATA SUBSTITUTE FAULT')

        else if (summary_parameter .eq. '0a'x) then

        write(lun,20) 'IB TB'
```

```
        else if (summary_parameter .eq. '0c'x) then

        write(lun,25) 'IB'

        else if (summary_parameter .eq. '0d'x) then

        call linchk (lun,1)

        write(lun,15) 'IB READ TIMEOUT'

        else if (summary_parameter .eq. '0f'x) then

        write(lun,20) 'IB CACHE'

        else if (summary_parameter .eq. 'f1'x) then

        write(lun,30) 'CONTROL STORE'
30      format(' ',t40,a,' PARITY ERROR ABORT')

        else if (summary_parameter .eq. 'f2'x) then

        write(lun,30) 'CP TB'

        else if (summary_parameter .eq. 'f3'x) then

        write(lun,30) 'CP CACHE'

        else if (summary_parameter .eq. 'f4'x) then

        call linchk (lun,1)

        write(lun,35) 'CP READ TIMEOUT',' - OR ERROR CONFIRMATION ABORT'
35      format(' ',t40,a,/,' ',t40,a)

        else if (summary_parameter .eq. 'f5'x) then

        write(lun,40) 'CP READ DATA SUBSTITUTE ABORT'
40      format(' ',t40,a)

        else if (summary_parameter .eq. 'f6'x) then

        write(lun,35) 'MICROCODE ''NOT SUPPOSED',' - TO BE HERE'' ABORT'
        else

        write(lun,40) 'UNKNOWN SUMMARY CODE'
        endif

        if (summary_parameter .eq. 'f1'x) then

        mnemonic = lib$extzv(0,8,cs_parity_error_opcode)

        call linchk (lun,1)

        write(lun,42) opcodes(mnemonic)
42      format(' ',t40,'OPCODE MNEMONIC = ',a<compressc (opcodes(mnemonic))>)
        endif
```

```
        if (summary_parameter .eq. '03'x
        1 .or.
        1 summary_parameter .eq. '0f'x
        1 .or.
        1 summary_parameter .eq. 'f3'x) then
        cache_group_disabled = lib$extzv(0,8,disabled_cache_group_flags)
        if (cache_group_disabled .ne. 0) then
        call linchk (lun,1)
        write(lun,45) 'CACHE GROUP ',cache_group_disabled,'. DISABLED BY VMS'
45      format(' ',t40,a,i<compress4 (cache_group_disabled)>,a)
        endif
        endif
        endif
        call linchk (lun,1)
        write(lun,50) 'CES',cpu_error_status_register
50      format(' ',t8,a,t24,z8.8)
        if (.not. diagnostic_mode) then
        ast_level = lib$extzv(1,2,cpu_error_status_register)
        call linchk (lun,1)
        write(lun,55) modes(ast_level)
55      format(' ',t40,a<compressc (modes(ast_level))>,' AST PENDING')
        call output (lun,cpu_error_status_register,v1cpu_error_status_register,
        1 3,3,3,'0')
        arithmetic_trap_code = lib$extzv(4,3,cpu_error_status_register)
        if (arithmetic_trap_code .ne. 0) then
        call linchk (lun,1)
        write(lun,60) traps(arithmetic_trap_code)
60      format(' ',t40,a<compressc (traps(arithmetic_trap_code))>)
        endif
        call output (lun,cpu_error_status_register,v2cpu_error_status_register,
        1 7,7,16,'0')
        endif
        call linchk (lun,6)
        write(lun,65) 'MICRO PC',trapped_upc,'VA/VIBA',va_viba_register,
        1 'D REGISTER',d_register
65      format(3(' ',t8,a,t24,z8.8,:/))
        call linchk (lun,1)
```

```
        write(lun,50) 'TBER0',tber0_register

        if (.not. diagnostic_mode) then

        call output (lun,tber0_register,v1tber0_register,0,0,0,'0')

        call output (lun,tber0_register,v2tber0_register,6,6,9,'0')

        call linchk (lun,1)

        write(lun,70) 'MICRO CODE ''MCT'' FIELD = ',
        1 lib$extzv(10,4,tber0_register)
70      format(' ',t40,a,Z2.2)

        call output (lun,tber0_register,v3tber0_register,14,14,15,'0')
        else

        call linchk (lun,1)

        write(lun,40) 'DIAGNOSTIC MODE'
        endif

        call linchk (lun,1)

        write(lun,50) 'TBER1',tber1_register

        if (.not. diagnostic_mode) then

        if (iand(tber1_register,'00000010'x) .eq. 0) then

        call output (lun,tber1_register,v1tber1_register,0,0,3,'0')
        else

        write(lun,40) 'BAD ''IPA'''
        endif

        last_tb_write_pulse = lib$extzv(5,2,tber1_register)

        call linchk (lun,1)

        if (last_tb_write_pulse .eq. 1) then

        write(lun,75) 'TO GROUP 0'
75      format(' ',t40,'LAST TB WRITE PULSE ',a)

        else if (last_tb_write_pulse .eq. 2) then

        write(lun,75) 'TO GROUP 1'

        else if (last_tb_write_pulse .eq. 3) then

        write(lun,75) 'UNPREDICTABLE'
        endif

        call output (lun,tber1_register,v2tber1_register,8,8,20,'0')
        endif
```

```
        if (.not. diagnostic_mode) then

        if (iand(sbi_error_register,'00001000'x) .ne. 0) then

        call sbi_timeout (lun,sbi_timeout_address_register)

        goto 80
        endif
        endif

        call linchk (lun,1)

        write(lun,50) 'SBITA',sbi_timeout_address_register

80      call linchk (lun,1)

        write(lun,50) 'CACHE PE REG',cache_parity_error_register

        if (.not. diagnostic_mode) then

        call output (lun,cache_parity_error_register,
        1 v1cache_parity_error_register,0,0,15,'0')
        endif

        if (.not. diagnostic_mode) then

        call sbi_error (lun,sbi_error_register)
        else

        call linchk (lun,1)

        write(lun,50) 'SBIER',sbi_error_register
        endif

        return

        end
```

```
PROGRAM SECTIONS

    Name                                  Bytes   Attributes

  0 $CODE                                  2492   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                                  786   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                                 2580   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG
  3 OPCODE                                 3840   PIC OVR REL GBL   SHR NOEXE   RD   WRT LONG
  4 MODE                                     55   PIC OVR REL GBL   SHR NOEXE   RD   WRT LONG

    Total Space Allocated                  9753

ENTRY POINTS

    Address    Type  Name

  0-00000000         MCHECK_780

VARIABLES

    Address    Type  Name                                  Address    Type  Name

  2-00000000   I*4   ARITHMETIC_TRAP_CODE                2-00000000   I*4   AST_LEVEL
  2-00000000   I*4   CACHE_GROUP_DISABLED                2-00000024   I*4   CACHE_PARITY_ERROR_REGISTER
  2-00000008   I*4   CPU_ERROR_STATUS_REGISTER           2-00000006   L*1   CS_PARITY_ERROR_OPCODE
  2-00000602   L*1   DIAGNOSTIC_MODE                     2-00000007   L*1   DISABLED_CACHE_GROUP_FLAGS
  2-00000014   I*4   D_REGISTER                          2-0000002C   I*4   EXCEPTION_PC
  2-00000030   I*4   EXCEPTION_PSL                       2-00000000   I*4   LAST_TB_WRITE_PULSE
 AP-00000004@  L*1   LUN                                 2-00000000   I*4   MNEMONIC
  2-00000028   I*4   SBI_ERROR_REGISTER                  2-00000020   I*4   SBI_TIMEOUT_ADDRESS_REGISTER
 AP-00000008@  I*4   START_OF_CPU_SPECIFIC_STUFF         2-00000004   I*4   SUMMARY_CODE
  2-00000004   I*2   SUMMARY_PARAMETER                   2-00000018   I*4   TBERO_REGISTER
  2-0000001C   I*4   TBER1_REGISTER                      2-0000000C   I*4   TRAPPED_UPC
  2-00000010   I*4   VA_VIBA_REGISTER

ARRAYS

    Address    Type  Name                                      Bytes  Dimensions

  2-00000004   I*4   BUFFER                                       48  (12)
  4-00000000   CHAR  MODES                                        55  (0:4)
  3-00000000   CHAR  OPCODES                                    3840  (0:255)
  2-00000004   L*1   SUMMARY_ARRAY                                 4  (4)
  2-0000004F   CHAR  TRAPS                                       140  (7)
  2-00000402   CHAR  V1CACHE_PARITY_ERROR_REGISTER               512  (0:15)
  2-00000034   CHAR  V1CPU_ERROR_STATUS_REGISTER                  27  (3:3)
  2-000001CB   CHAR  V1TBERO_REGISTER                             25  (0:0)
  2-00000242   CHAR  V1TBER1_REGISTER                            136  (0:3)
  2-000000DB   CHAR  V2CPU_ERROR_STATUS_REGISTER                 240  (7:16)
  2-000001E4   CHAR  V2TBERO_REGISTER                             60  (6:9)
  2-000002CA   CHAR  V2TBER1_REGISTER                            312  (8:20)
  2-00000220   CHAR  V3TBERO_REGISTER                             34  (14:15)
```

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-000001CD	10'	1-000001DA	15'	1-00000207	20'	1-00000223	25'	1-00000247	30'	1-00000263	35'
1-00000271	40'	1-0C000278	42'	1-00000298	45'	1-000002A6	50'	1-000002B2	55'	1-000002CC	60'
1-000002D8	65'	1-000002EB	70'	1-000002F5	75'	0-000008A3	80				

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	COMPRESSC	I*4	LIB$EXTZV		LINCHK		MOVC3		OUTPUT
	SBI_ERROR		SBI_TIMEOUT		VAXPSL						

```
c++
c       Functional description:
c
c       This routine displays machine checks for the 11/750.  The format
c       of the entry is as follows.
c
c       +------------------------------------+
c       |            error code              |
c       +------------------------------------+
c       |          va last fetch             |
c       +------------------------------------+
c       |           pc at error              |
c       +------------------------------------+
c       |   memory data last reference       |
c       +------------------------------------+
c       |       saved mode register          |
c       +------------------------------------+
c       |      write vector register         |
c       +------------------------------------+
c       |              tber                  |
c       +------------------------------------+
c       |              caer                  |
c       +------------------------------------+
c       |        bus error register          |
c       +------------------------------------+
c       |          error summary             |
c       +------------------------------------+
c       |          exception pc              |
c       +------------------------------------+
c       |          exception psl             |
c       +------------------------------------+
c--



        subroutine mcheck_750 (lun,start_of_cpu_specific_stuff)


        include 'src$:opcodes.for /nolist'


        byte            lun

        integer*4       start_of_cpu_specific_stuff

        integer*4       buffer(12)

        integer*4       error_code

        integer*4       va_last_fetch
```

```
        integer*4       pc_at_error

        integer*4       memory_data_last_reference

        integer*4       saved_mode_register

        integer*4       write_vector_register

        integer*4       tber_register

        integer*4       caer_register

        integer*4       bus_error_register

        integer*4       machine_check_error_summary

        integer*4       exception_pc

        integer*4       exception_psl

        equivalence     (buffer(1),error_code)

        equivalence     (buffer(2),va_last_fetch)

        equivalence     (buffer(3),pc_at_error)

        equivalence     (buffer(4),memory_data_last_reference)

        equivalence     (buffer(5),saved_mode_register)

        equivalence     (buffer(6),write_vector_register)

        equivalence     (buffer(7),tber_register)

        equivalence     (buffer(8),caer_register)

        equivalence     (buffer(9),bus_error_register)

        equivalence     (buffer(10),machine_check_error_summary)

        equivalence     (buffer(11),exception_pc)

        equivalence     (buffer(12),exception_psl)

        integer*4       mnemonic

        integer*4       compressc

        logical*1       diagnostic_mode





        call movc3 (%val(48),start_of_cpu_specific_stuff,buffer)

        diagnostic_mode = .false.
```

```
              call linchk (lun,2)

              write(lun,10) 'EXCEPTION PC',exception_pc
        10    format(/' ',t8,a,t24,z8.8)

              call vaxpsl (lun,exception_psl)

              call linchk (lun,3)

              write(lun,15) 'SUMMARY CODE',error_code
        15    format(/' ',t8,a,t24,z8.8)

              if (.not. diagnostic_mode) then

              if (error_code .eq. 1) then

              write(lun,20) 'CONTROL STORE PARITY ERROR'
        20    format(' ',t40,a)

              else if (error_code .eq. 2) then

              write(lun,20) 'TRANSLATION BUFFER OR BUS ERROR'

              else if (error_code .eq. 3) then

              write(lun,20) 'CACHE PARITY ERROR'

              else if (error_code .eq. 6) then

              write(lun,20) 'MICRO CODE LOST'

              else if (error_code .eq. 7) then

              write(lun,20) 'IRD ROM LOST'
              else

              write(lun,20) 'UNKNOWN SUMMARY CODE'
              endif
              endif

              call linchk (lun,4)

              write(lun,25) 'VA LAST REF',va_last_fetch,'PC AT ERROR',pc_at_error,
              1 'MDR',memory_data_last_reference,'SMR',saved_mode_register
        25    format(4(' ',t8,a,t24,z8.8,:/))

              if (.not. diagnostic_mode) then

              call cmier_1916 (lun,ishft(iand(saved_mode_register,'0000000f'x),16))
              endif

              call linchk (lun,1)

              write(lun,30) 'RLTO',write_vector_register
        30    format(' ',t8,a,t24,z8.8)

```

```
	if (.not. diagnostic_mode) then

	call cmier_1512 (lun,ishft(iand(write_vector_register,'00000001'x),12))
	endif

	call linchk (lun,1)

	write(lun,30) 'TBER',tber_register

	if (.not. diagnostic_mode) then

	call cmier_118 (lun,ishft(iand(tber_register,'0000000f'x),8))

	if (iand(tber_register,'00010000'x) .ne. 0) then

	call linchk (lun,1)

	write(lun,35) 'TB GROUP 0 DISABLED (VMS)'
35	format(' ',t40,a)

	else if (iand(tber_register,'00020000'x) .ne. 0) then

	write(lun,35) 'TB GROUP 1 DISABLED (VMS)'
	endif
	else

	call linchk (lun,1)

	write(lun,37) 'DIAGNOSTIC MODE'
37	format(' ',t40,a)
	endif

	if (.not. diagnostic_mode) then

	call comet_caer (lun,iand(caer_register,'fffcffff'x))

	if (error_code .eq. 3) then

	if (iand(caer_register,'00010000'x) .ne. 0) then

	write(lun,35) 'CACHE DISABLED (VMS)'
	endif
	endif
	else

	call linchk (lun,1)

	write(lun,10) 'CAER',caer_register
	endif

	call linchk (lun,1)

	write(lun,30) 'BER',bus_error_register

	if (.not. diagnostic_mode) then

	call cmier_30 (lun,bus_error_register)
```

```
        call comet_mcesr (lun,machine_check_error_summary)

        if (error_code .eq. 1
        1 .or.
        1 error_code .eq. 6
        1 .or.
        1 error_code .eq. 7) then

        mnemonic = lib$extzv(16,8,machine_check_error_summary)

        call linchk (lun,1)

        write(lun,40) opcodes(mnemonic)
40      format(' ',t40,'OPCODE MNEMONIC = ',a<compressc (opcodes(mnemonic))>)
        endif
        else

        call linchk (lun,1)

        write(lun,10) 'MCESR',machine_check_error_summary
        endif

        return

        end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	1376	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	413	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	376	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	OPCODE	3840	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	6005	

ENTRY POINTS

Address	Type	Name
0-00000000		MCHECK_750

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000020	I*4	BUS_ERROR_REGISTER	2-0000001C	I*4	CAER_REGISTER
2-00000030	L*1	DIAGNOSTIC_MODE	2-00000000	I*4	ERROR_CODE
2-00000028	I*4	EXCEPTION_PC	2-0000002C	I*4	EXCEPTION_PSL
AP-00000004@	L*1	LUN	2-00000024	I*4	MACHINE_CHECK_ERROR_SUMMARY

```
  2-0000000C   I*4  MEMORY_DATA_LAST_REFERENCE         2-00000034  I*4  MNEMONIC
  2-00000008   I*4  PC_AT_ERROR                        2-00000010  I*4  SAVED_MODE_REGISTER
 AP-00000008a  I*4  START_OF_CPU_SPECIFIC_STUFF        2-00000018  I*4  TBER_REGISTER
  2-00000004   I*4  VA_LAST_FETCH                      2-00000014  I*4  WRITE_VECTOR_REGISTER
```

ARRAYS

```
    Address  Type  Name                  Bytes  Dimensions

  2-00000000  I*4  BUFFER                   48  (12)
  3-00000000  CHAR OPCODES                3840  (0:255)
```

LABELS

```
    Address   Label       Address   Label       Address   Label       Address   Label       Address   Label       Address   Label

  1-0000012F  10'       1-0000013C  15'       1-00000149  20'       1-00000150  25'       1-00000163  30'       1-0000016F  35'
  1-00000176  37'       1-0000017D  40'
```

FUNCTIONS AND SUBROUTINES REFERENCED

```
  Type  Name             Type  Name             Type  Name             Type  Name             Type  Name             Type  Name

        CMIER_118              CMIER_1512             CMIER_1916             CMIER_30               COMET_CAER             COMET_MCESR
   I*4  COMPRESSC         I*4  LIB$EXTZV              LINCHR                 MOVC3                  VAXPSC
```

```
c++
c       Functional description:
c
c       This routine displays machine checks for the 11/7zz.  The format
c       of the entry is as follows.
c
c       +-----------------------------------+
c       :      machine check type code      :
c       +-----------------------------------+
c       :          first parameter          :
c       +-----------------------------------+
c       :         second parameter          :
c       +-----------------------------------+
c       :           exception pc            :
c       +-----------------------------------+
c       :           exception psl           :
c       +-----------------------------------+
c
c--
        subroutine mcheck_7zz (lun,start_of_cpu_specific_stuff)
        byte            lun
        integer*4       start_of_cpu_specific_stuff
        integer*4       buffer(5)
        integer*4       machine_check_type_code
        integer*4       first_parameter
        integer*4       second_parameter
        integer*4       exception_pc
        integer*4       exception_psl
        equivalence     (buffer(1),machine_check_type_code)
        equivalence     (buffer(2),first_parameter)
        equivalence     (buffer(3),second_parameter)
        equivalence     (buffer(4),exception_pc)
        equivalence     (buffer(5),exception_psl)
        integer*4       error_pfn
```

```

        integer*4           compress4

        character*21        v1fpa_error_bits(0:2)

        data                v1fpa_error_bits(0)
        1 /'PARITY ERROR*'/

        data                v1fpa_error_bits(1)
        1 /'GROUP 0 PARITY ERROR*'/

        data                v1fpa_error_bits(2)
        1 /'GROUP 1 PARITY ERROR*'/




        call movc3 (%val(20),start_of_cpu_specific_stuff,buffer)

        call linchk (lun,2)

        write(lun,10) 'EXCEPTION PC',exception_pc
10      format(/' ',t8,a,t24,z8.8)

        call vaxpsl (lun,exception_psl)

        call linchk (lun,3)

        write(lun,10) 'SUMMARY CODE',machine_check_type_code

        if (machine_check_type_code .eq. 0) then

        write(lun,20) 'MICRO CODE SHOULDN''T BE HERE'
20      format(' ',t40,a)

        if (first_parameter .eq. 2) then

        call linchk (lun,2)

        write(lun,25) first_parameter,'CANNOT WRITE BACK PTE <M> BIT'
25      format(' ',t8,'1ST PARAMETER',t24,z8.8,:/,:' ',:t40,:a)

        else if (first_parameter .eq. 3) then

        call linchk (lun,2)

        write(lun,25) first_parameter,'BAD 8085 INTERRUPT'
        else

        goto 50
        endif

        goto 60

        else if (machine_check_type_code .eq. 1) then

        write(lun,20) 'TRANSLATION BUFFER PARITY ERROR'
```

```
          call linchk (lun,2)
          write(lun,25) first_parameter,'PTE FROM TRANSLATION BUFFER'
          call linchk (lun,2)
          write(lun,30) second_parameter,'PTE FROM MEMORY'
30        format(' ',t8,'2ND PARAMETER',t24,z8.8,:/,:t40,:a)
          return
          else if (machine_check_type_code .eq. 3) then
          write(lun,20) 'IMPOSSIBLE VALUE IN MEMORY CSR'
          call linchk (lun,2)
          write(lun,25) first_parameter,'VIRTUAL ADDRESS REFERENCED'
          goto 60
          else if (machine_check_type_code .eq. 4) then
          write(lun,20) 'FAST INTERRUPT WITHOUT SUPPORT'
          if (first_parameter .ne. 0
          1 .or.
          1 second_parameter .ne. 0) goto 50
          return
          else if (machine_check_type_code .eq. 5) then
          write(lun,20) 'FPA PARITY ERROR'
          call output (lun,first_parameter,v1fpa_error_bits,0,0,2,'0')
          goto 60
          else if (machine_check_type_code .eq. 6) then
          write(lun,20) 'ERROR ON ''SPTE'' READ'
          call linchk (lun,2)
          write(lun,25) first_parameter,'PHYSICAL ADDRESS OF ''SPTE'''
          goto 60
          else if (machine_check_type_code .eq. 7) then
          write(lun,20) 'UNCORRECTABLE ECC ERROR'
          call linchk (lun,2)
          write(lun,35) first_parameter
```

```
35      format(' ',t8,'1ST PARAMETER',t24,z8.8)

        error_pfn = lib$extzv(9,15,first_parameter)

        write(lun,37) error_pfn
37      format(' ',t40,'PAGE #',i<compress4 (error_pfn)>,'. IN ERROR')

        goto 60

        else if (machine_check_type_code .eq. 8) then

        write(lun,20) 'NON-EXISTENT MEMORY'

        call linchk (lun,1)

        write(lun,40) first_parameter
40      format(' ',t8,'1ST PARAMETER',t24,z8.8,/,
        1 t40,'PHYSICAL ADDRESS REFERENCED')

        goto 60

        else if (machine_check_type_code .eq. 9
        1 .or.
        1 machine_check_type_code .eq. 10
        1 .or.
        1 machine_check_type_code .eq. 11) then

        if (machine_check_type_code .ne. 11) then

        write(lun,20) 'ILLEGAL I/O SPACE REFERENCE'
        else

        write(lun,20) 'ILLEGAL UNIBUS REFERENCE'
        endif

        call linchk (lun,1)

        write(lun,40) first_parameter

        goto 60
        else

        write(lun,20) 'UNKNOWN SUMMARY CODE'
        endif

50      continue

        call linchk (lun,2)

        write(lun,55) first_parameter,second_parameter
55      format(' ',t8,'1ST PARAMETER',t24,z8.8,/,' ',t8,'2ND PARAMETER',t24,
        1 z8.8)

        return

60      continue
```

```
        if (second_parameter .ne. 0) then

        call linchk (lun,1)

        write(lun,30) second_parameter
        endif

        return

        end
```

PROGRAM SECTIONS

```
   Name                                   Bytes   Attributes

 0 $CODE                                   1353   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
 1 $PDATA                                   717   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
 2 $LOCAL                                   400   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG

   Total Space Allocated                   2470
```

ENTRY POINTS

```
   Address    Type  Name

 0-00000000         MCHECK_7ZZ
```

VARIABLES

```
   Address    Type  Name                                 Address    Type  Name

 2-00000054   I*4   ERROR_PFN                          2-0000000C   I*4   EXCEPTION_PC
 2-00000010   I*4   EXCEPTION_PSL                      2-00000004   I*4   FIRST_PARAMETER
AP-00000004@  L*1   LUN                                2-00000000   I*4   MACHINE_CHECK_TYPE_CODE
 2-00000008   I*4   SECOND_PARAMETER                  AP-00000008@  I*4   START_OF_CPU_SPECIFIC_STUFF
```

ARRAYS

```
   Address    Type  Name                                    Bytes  Dimensions

 2-00000000   I*4   BUFFER                                     20  (5)
 2-00000014   CHAR  V1FPA_ERROR_BITS                           63  (0:2)
```

LABELS

```
   Address    Label      Address    Label      Address    Label      Address    Label      Address    Label      Address    Label

 1-000001CB   10'     1-000001D8   20'      1-000001DF   25'      1-00000204   30'      1-00000225   35'      1-0000023F   37'
 1-0000025F   40'     0-000004CA   50       1-00000299   55'      0-00000504   60
```

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK		MOVC3		OUTPUT		VAXPSL

```
c++
c       Functional description:
c
c       This routine displays machine checks for the UVAX1.  The format
c       of the entry is as follows.
c
c       +------------------------------------+
c       |         machine check code         |
c       +------------------------------------+
c       |           first parameter          |
c       +------------------------------------+
c       |          second parameter          |
c       +------------------------------------+
c       |            exception pc            |
c       +------------------------------------+
c       |            exception psl           |
c       +------------------------------------+
c
c--

        subroutine mcheck_uv1 (lun,start_of_cpu_specific_stuff)

        implicit        none
        byte            lun
        integer*4       start_of_cpu_specific_stuff
        integer*4       buffer(5)
        integer*4       machine_check_code
        integer*4       parameter_1
        integer*4       parameter_2
        integer*4       exception_pc
        integer*4       exception_psl
        equivalence     (buffer(1),machine_check_code)
        equivalence     (buffer(2),parameter_1)
        equivalence     (buffer(3),parameter_2)
        equivalence     (buffer(4),exception_pc)
        equivalence     (buffer(5),exception_psl)
```

```
        integer*4       compress4

        character*27    address_previous_function

        data            address_previous_function
        1 /'PHYS ADDR PREVIOUS FUNCTION'/

        character*26    address_current_function

        data            address_current_function
        1 /'PHYS ADDR CURRENT FUNCTION'/


        character*25    physical_pte_address

        data            physical_pte_address
        1 /'PHYSICAL ADDRESS OF "PTE"'/

        character*25    virtual_pte_address

        data            virtual_pte_address
        1 /'VIRTUAL ADDRESS FOR "PTE"'/



        call movc3 (%val(20),start_of_cpu_specific_stuff,buffer)

        call linchk (lun,2)

        write(lun,5) 'EXCEPTION PC',exception_pc
5       format(' ',t8,a,t24,z8.8)

        call vaxpsl (lun,exception_psl)

        call linchk (lun,3)

        write(lun,5) 'MACHINE CHECK',machine_check_code

        if (machine_check_code .eq. 1) then

        write(lun,10) 'MEMORY CONTROLLER BUGCHECK'
10      format(' ',t40,a)

        call parameter_one (lun,parameter_1,.true.)

        call linchk (lun,1)

        write(lun,10) address_previous_function

        call parameter_two (lun,parameter_2,.true.)

        write(lun,10) address_current_function

        else if (machine_check_code .eq. 2) then

        write(lun,10) 'UNRECOVERABLE READ ERROR'

```

```
        call parameter_one (lun,parameter_1,.true.)

        call linchk (lun,1)

        write(lun,10) 'PHYSICAL PAGE ADDRESS OF ERROR'

        call parameter_two (lun,parameter_2,.true.)

        call linchk (lun,1)

        write(lun,10) address_current_function

        else if (machine_check_code .eq. 3) then

        write(lun,10) 'NON-EXISTENT MEMORY'

        call parameter_one (lun,parameter_1,.true.)

        call linchk (lun,1)

        write(lun,10) 'PHYSICAL ADDRESS OF ERROR'

        call parameter_two (lun,parameter_2,.true.)

        write(lun,10) address_current_function

        else if (machine_check_code .eq. 4) then

        write(lun,10) 'ILLEGAL OPERATION'

        call parameter_one (lun,parameter_1,.true.)

        call linchk (lun,1)

        write(lun,10) 'UNALIGNED I/O REFERENCE'

        call parameter_two (lun,parameter_2,.true.)

        call linchk (lun,1)

        write(lun,10) address_current_function

        else if (machine_check_code .eq. 5) then

        write(lun,10) 'FATAL PAGE TABLE READ ERROR'

        call parameter_one (lun,parameter_1,.true.)

        call linchk (lun,1)

        write(lun,10) physical_pte_address

        call parameter_two (lun,parameter_2,.true.)

        call linchk (lun,1)

        write(lun,10) virtual_pte_address
```

```
        else if (machine_check_code .eq. 6) then
        write(lun,10) 'FATAL PAGE TABLE WRITE ERROR'
        call parameter_one (lun,parameter_1,.true.)
        call linchk (lun,1)
        write(lun,10) physical_pte_address
        call parameter_two (lun,parameter_2,.true.)
        call linchk (lun,1)
        write(lun,10) virtual_pte_address
        else if (machine_check_code .eq. 7) then
        write(lun,10) 'CONTROL STORE PARITY ERROR'
        call parameter_one (lun,parameter_1,.false.)
        call parameter_two (lun,parameter_2,.false.)
        else if (machine_check_code .eq. 8) then
        write(lun,10) 'MICRO MACHINE BUGCHECK'
        call parameter_one (lun,parameter_1,.false.)
        call parameter_two (lun,parameter_2,.false.)
        else if (machine_check_code .eq. 9) then
        write(lun,10) '''Q-BUS'' VECTOR READ ERROR'
        call parameter_one (lun,parameter_1,.false.)
        call parameter_two (lun,parameter_2,.false.)
        else if (machine_check_code .eq. 10) then
        write(lun,10) 'WRITE PARAMETER ERROR'
        call parameter_one (lun,parameter_1,.true.)
        call linchk (lun,1)
        write(lun,10) 'VIRTUAL ADDRESS BEING WRITTEN'
        call parameter_two (lun,parameter_2,.false.)

c
c       The IF-THEN-ELSE should be expanded at this point to add
c       additional mcheck types for uv1.
c
```

MCH
V04

;

;

;

; S
; R
; E
; L
; L
; M
; C

```
        else

        write(lun,15) 'TYPE CODE #',machine_check_code
15      format(' ',t40,a,i<compress4 (machine_check_code)>,'.')

        call parameter_one (lun,parameter_1,.true.)

        call parameter_two (lun,parameter_2,.true.)
        endif

        return

        end
```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 $CODE	1578	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 $PDATA	433	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 $LOCAL	428	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	2439	

ENTRY POINTS

Address	Type	Name
0-00000000		MCHECK_UV1

VARIABLES

Address	Type	Name	Address	Type	Name
2-0000002F	CHAR	ADDRESS_CURRENT_FUNCTION	2-00000014	CHAR	ADDRESS_PREVIOUS_FUNCTION
2-0000000C	I*4	EXCEPTION_PC	2-00000010	I*4	EXCEPTION_PSL
AP-00000004@	L*1	LUN	2-00000000	I*4	MACHINE_CHECK_CODE
2-00000004	I*4	PARAMETER_1	2-00000008	I*4	PARAMETER_2
2-00000049	CHAR	PHYSICAL_PTE_ADDRESS	AP-00000008@	I*4	START_OF_CPU_SPECIFIC_STUFF
2-00000062	CHAR	VIRTUAL_PTE_ADDRESS			

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	I*4	BUFFER	20	(5)

**F

LABELS

Address	Label	Address	Label	Address	Label
1-0000018E	5'	1-0000019A	10'	1-000001A1	15'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name	Type	Name	Type	Name
I*4	COMPRESS4		LINCHK		MOVC3		PARAMETER_ONE		PARAMETER_TWO		VAXPSL

```
        subroutine parameter_one (lun,parameter_1,expand)



        implicit        none

        byte            lun

        integer*4       parameter_1

        logical*1       expand



        if (.not. expand) then

        if (parameter_1 .eq. 0) goto 20
        endif

        call linchk (lun,1)

        write(lun,10) parameter_1
10      format(' ',t8,'PARAMETER 1',t24,z8.8)

20      return

        end
```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 $CODE	66	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 $PDATA	28	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 $LOCAL	12	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	106	

ENTRY POINTS

Address	Type	Name
0-00000000		PARAMETER_ONE

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name
AP-0000000C@	L*1	EXPAND	AP-00000004@	L*1	LUN	AP-00000008@	I*4	PARAMETER_1

LABELS

Address	Label	Address	Label
1-00000004	10'	0-00000041	20

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
	LINCHK

```
        subroutine parameter_two (lun,parameter_2,expand)


        implicit        none

        byte            lun

        integer*4       parameter_2

        logical*1       expand


        if (.not. expand) then

        if (parameter_2 .eq. 0) goto 20
        endif

        call linchk (lun,1)

        write(lun,10) parameter_2
10      format(' ',t8,'PARAMETER 2',t24,z8.8)

20      return

        end
```

```
PROGRAM SECTIONS

   Name                              Bytes   Attributes

 0 $CODE                                66   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
 1 $PDATA                               28   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
 2 $LOCAL                               12   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG

   Total Space Allocated               106

ENTRY POINTS

   Address   Type  Name

 0-00000000        PARAMETER_TWO

VARIABLES

   Address   Type  Name                Address   Type  Name                Address   Type  Name

 AP-0000000C@ L*1  EXPAND           AP-00000004@ L*1  LUN               AP-00000008@ I*4  PARAMETER_2

LABELS

   Address    Label        Address    Label

 1-00000004   10'        0-00000041   20

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name

        LINCHK

COMMAND QUALIFIERS

  FORTRAN /LIS=LIS$:MCHECKS/OBJ=OBJ$:MCHECKS MSRC$:MCHECKS

  /CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
  /DEBUG=(NOSYMBOLS,TRACEBACK)
  /STANDARD=(NOSYNTAX,NOSOURCE_FORM)
  /SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP)
  /F77  /NOG_FLOATING  /I4  /OPTIMIZE  /WARNINGS  /NOD_LINES  /NOCROSS_REFERENCE  /NOMACHINE_CODE  /CONTINUATIONS=19

COMPILATION STATISTICS

  Run Time:           17.33 seconds
  Elapsed Time:       35.52 seconds
  Page Faults:        255
  Dynamic Memory:     224 pages
```

0150 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

LPA11K
LIS

MCHECKS
LIS

LCPRINTER
LIS

MBA
LIS

LOGMSCP
LIS

LINCHK
LIS

MBAINT
LIS

LABEL
LIS

LOGGER
LIS

MASSTAPE
LIS

MASSDISK
LIS

0151 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MESSAGE
LIS

ML11
LIS

MSCP
LIS

MFTAPE
LIS

MOUNT
LIS

MEMORYS
LIS

MOVXX
LIS

MCHK_DISP
LIS