```
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
```

```
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
```

```
0001     C
0002     C Version:      'V04-000'
0003     C
0004     C*****************************************************************************
0005     C*                                                                           *
0006     C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0007     C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0008     C*  ALL RIGHTS RESERVED.                                                     *
0009     C*                                                                           *
0010     C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0011     C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0012     C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0013     C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0014     C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0015     C*  TRANSFERRED.                                                             *
0016     C*                                                                           *
0017     C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0018     C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0019     C*  CORPORATION.                                                             *
0020     C*                                                                           *
0021     C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0022     C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0023     C*                                                                           *
0024     C*                                                                           *
0025     C*****************************************************************************
0026     C
0027
0028     c       author  Brian Porter            Creation Date   11-MAY-1981
0029
0030     c++
0031     c       Function description:
0032     c
0033     c       This module dispatches to the correct machine check display routine
0034     c       based on emb$l_hd_sid.
0035     c
0036     c       Modified by:
0037     c
0038     C       V03-007 SAR0249         Sharon A. Reynolds      11-Apr-1984
0039     C               - Changed carriage control in format statements for
0040     C               use with ERF.
0041     C               - Removed the MCHECKS routine, re-written and named
0042     C               MCHK_DISPATCHER.
0043     C
0044     c       V03-006 EAD0007         Elliott A. Drayton      21-Feb-1984
0045     c               Added UVAX-1 support.
0046     c
0047     C       V03-005 SAR0082         Sharon A. Reynolds,     20-Jun-1983
0048     C               Changed the carriage control in the 'format' statements
0049     C               for use with ERF.
0050     C
0051     c       v03-004 BP0004          Brian Porter,           28-MAR-1983
0052     c               Corrected 11/750 TB parity error stuff.
0053     c
0054     c       vC3-003 BP0003          Brian Porter,           11-JAN-1982
0055     c               Corrected mcheck type 7, ms7zz code.
0056     c
0057     c       v03-002 BP0002          Brian Porter,           27-AUG-1981
```

```
0058     c                Corrected mcheck_780, SBITA register.
0059     c
0060     c        v03-001 BP0001          Brian Porter,          29-JUN-1981
0061     c                Added call to LOGGER.
0062     c**
0063     c--
0064
0065     c++
0066     c        Functional description:
0067     c
0068     c        This routine displays mchecks for the 11/780.  The format of the
0069     c        cpu specific piece of the error log is as follows.
0070     c        The summary parameter is divided up into the following pieces
0071     c        bytes 1 & 2; summary code.  Byte 3; the opcode of the failing
0072     c        instruction if the machine check is a control store parity error
0073     c        abort.  Byte 4, bit 0; byte 4, bit 1; if the machine check is either
0074     c        a cp or ib cache parity error fault or abort then each bit represents
0075     c        the cache group if any that was disabled by vms as a result of the
0076     c        machine check.
0077     c
0078     c        +--------------------------------------+
0079     c        |          summary parameter           |
0080     c        +--------------------------------------+
0081     c        |       cpu error status register      |
0082     c        +--------------------------------------+
0083     c        |             trapped uPC              |
0084     c        +--------------------------------------+
0085     c        |                VA/VIBA               |
0086     c        +--------------------------------------+
0087     c        |                 D reg                |
0088     c        +--------------------------------------+
0089     c        |                 TBER0                |
0090     c        +--------------------------------------+
0091     c        |                 TBER1                |
0092     c        +--------------------------------------+
0093     c        |                 SBITA                |
0094     c        +--------------------------------------+
0095     c        |     cache parity error register      |
0096     c        +--------------------------------------+
0097     c        |                 SBIER                |
0098     c        +--------------------------------------+
0099     c        |              exception PC            |
0100     c        +--------------------------------------+
0101     c        |              exception PSL           |
0102     c        +--------------------------------------+
0103     c--
0104
0105
0106
0107             subroutine mcheck_780 (lun,start_of_cpu_specific_stuff)
0108
0109
0110
0111             include 'src$:opcodes.for /nolist'
0149
0150             include 'src$:modes.for /nolist'
0188
```

```
0189
0190
0191
0192            byte            lun
0193
0194            integer*4       start_of_cpu_specific_stuff
0195
0196            integer*4       buffer(12)
0197
0198            integer*4       summary_code
0199
0200            byte            summary_array(4)
0201
0202            integer*2       summary_parameter
0203
0204            byte            cs_parity_error_opcode
0205
0206            byte            disabled_cache_group_flags
0207
0208            integer*4       cpu_error_status_register
0209
0210            integer*4       trapped_upc
0211
0212            integer*4       va_viba_register
0213
0214            integer*4       d_register
0215
0216            integer*4       tber0_register
0217
0218            integer*4       tber1_register
0219
0220            integer*4       sbi_timeout_address_register
0221
0222            integer*4       cache_parity_error_register
0223
0224            integer*4       sbi_error_register
0225
0226            integer*4       exception_pc
0227
0228            integer*4       exception_psl
0229
0230            equivalence     (buffer(1),summary_array)
0231
0232            equivalence     (buffer(1),summary_code)
0233
0234            equivalence     (summary_array(1),summary_parameter)
0235
0236            equivalence     (summary_array(3),cs_parity_error_opcode)
0237
0238            equivalence     (summary_array(4),disabled_cache_group_flags)
0239
0240            equivalence     (buffer(2),cpu_error_status_register)
0241
0242            equivalence     (buffer(3),trapped_upc)
0243
0244            equivalence     (buffer(4),va_viba_register)
0245
```

```
0246            equivalence     (buffer(5),d_register)
0247
0248            equivalence     (buffer(6),tber0_register)
0249
0250            equivalence     (buffer(7),tber1_register)
0251
0252            equivalence     (buffer(8),sbi_timeout_address_register)
0253
0254            equivalence     (buffer(9),cache_parity_error_register)
0255
0256            equivalence     (buffer(10),sbi_error_register)
0257
0258            equivalence     (buffer(11),exception_pc)
0259
0260            equivalence     (buffer(12),exception_psl)
0261
0262            character*27    v1cpu_error_status_register(3:3)
0263
0264            data            v1cpu_error_status_register(3)
0265            1 /'PERFORMANCE MONITOR ENABLE*'/
0266
0267            character*20    traps(1:7)
0268
0269            data            traps(1)
0270            1 /'INTEGER OVERFLOW*'/
0271
0272            data            traps(2)
0273            1 /'INTEGER DIVIDE BY 0*'/
0274
0275            data            traps(3)
0276            1 /'FLOAT OVERFLOW*'/
0277
0278            data            traps(4)
0279            1 /'FLOAT DIVIDE BY 0*'/
0280
0281            data            traps(5)
0282            1 /'FLOAT UNDERFLOW*'/
0283
0284            data            traps(6)
0285            1 /'DECIMAL OVERFLOW*'/
0286
0287            data            traps(7)
0288            1 /'DECIMAL DIVIDE BY 0*'/
0289
0290            character*24    v2cpu_error_status_register(7:16)
0291
0292            data            v2cpu_error_status_register(7)
0293            1 /'ALU C31*'/
0294
0295            data            v2cpu_error_status_register(8)
0296            1 /'ALU Z*'/
0297
0298            data            v2cpu_error_status_register(9)
0299            1 /'ALU N*'/
0300
0301            data            v2cpu_error_status_register(10)
0302            1 /'E ALU Z*'/
```

```
0303
0304            data            v2cpu_error_status_register(11)
0305            1 /'E ALU N*'/
0306
0307            data            v2cpu_error_status_register(12)
0308            1 /'CS PARITY ERROR GROUP 0*'/
0309
0310            data            v2cpu_error_status_register(13)
0311            1 /'CS PARITY ERROR GROUP 1*'/
0312
0313            data            v2cpu_error_status_register(14)
0314            1 /'CS PARITY ERROR GROUP 2*'/
0315
0316            data            v2cpu_error_status_register(15)
0317            1 /'CS PARITY ERROR SUMMARY*'/
0318
0319            data            v2cpu_error_status_register(16)
0320            1 /'NESTED ERROR*'/
0321
0322            character*25    v1tber0_register(0:0)
0323
0324            data            v1tber0_register(0)
0325            1 /'ENABLE MEMORY MANAGEMENT*'/
0326
0327            character*15    v2tber0_register(6:9)
0328
0329            data            v2tber0_register(6)
0330            1 /'TB HIT GROUP 0*'/
0331
0332            data            v2tber0_register(7)
0333            1 /'TB HIT GROUP 1*'/
0334
0335            data            v2tber0_register(8)
0336            1 /'IB AUTO RELOAD*'/
0337
0338            data            v2tber0_register(9)
0339            1 /'IB 'WCHK''*'/
0340
0341            character*17    v3tber0_register(14:15)
0342
0343            data            v3tber0_register(14)
0344            1 /'MICRO CODE ''ADS''*'/
0345
0346            data            v3tber0_register(15)
0347            1 /'MICRO CODE ''FS''*'/
0348
0349            character*34    v1tber1_register(0:3)
0350
0351            data            v1tber1_register(0)
0352            1 /'AUTOMATIC HARDWARE INITIATED LOAD*'/
0353
0354            data            v1tber1_register(1)
0355            1 /'PROTECTION VIOLATION OR MISS*'/
0356
0357            data            v1tber1_register(2)
0358            1 /'TB PARITY ERROR*'/
0359
```

```
0360            data            v1tber1_register(3)
0361            1 /'TB MISS ON LOAD*'/
0362
0363            character*24    v2tber1_register(8:20)
0364
0365            data            v2tber1_register(8)
0366            1 /'CP TB PARITY ERROR*'/
0367
0368            data            v2tber1_register(9)
0369            1 /'GROUP 0, ADDRESS BYTE 0*'/
0370
0371            data            v2tber1_register(10)
0372            1 /'GROUP 0, ADDRESS BYTE 1*'/
0373
0374            data            v2tber1_register(11)
0375            1 /'GROUP 0, ADDRESS BYTE 2*'/
0376
0377            data            v2tber1_register(12)
0378            1 /'GROUP 1, ADDRESS BYTE 0*'/
0379
0380            data            v2tber1_register(13)
0381            1 /'GROUP 1, ADDRESS BYTE 1*'/
0382
0383            data            v2tber1_register(14)
0384            1 /'GROUP 1, ADDRESS BYTE 2*'/
0385
0386            data            v2tber1_register(15)
0387            1 /'GROUP 0, DATA BYTE 0*'/
0388
0389            data            v2tber1_register(16)
0390            1 /'GROUP 0, DATA BYTE 1*'/
0391
0392            data            v2tber1_register(17)
0393            1 /'GROUP 0, DATA BYTE 2*'/
0394
0395            data            v2tber1_register(18)
0396            1 /'GROUP 1, DATA BYTE 0*'/
0397
0398            data            v2tber1_register(19)
0399            1 /'GROUP 1, DATA BYTE 1*'/
0400
0401            data            v2tber1_register(20)
0402            1 /'GROUP 1, DATA BYTE 2*'/
0403
0404            character*32    v1cache_parity_error_register(0:15)
0405
0406            data            v1cache_parity_error_register(0)
0407            1 /'PARITY OK "CAM" GROUP 1, BYTE 2*'/
0408
0409            data            v1cache_parity_error_register(1)
0410            1 /'PARITY OK "CAM" GROUP 1, BYTE 1*'/
0411
0412            data            v1cache_parity_error_register(2)
0413            1 /'PARITY OK "CAM" GROUP 1, BYTE 0*'/
0414
0415            data            v1cache_parity_error_register(3)
0416            1 /'PARITY OK "CAM" GROUP 0, BYTE 2*'/
```

```
0417
0418            data                vicache_parity_error_register(4)
0419            1 /'PARITY OK "CAM" GROUP 0, BYTE 1*'/
0420
0421            data                vicache_parity_error_register(5)
0422            1 /'PARITY OK "CAM" GROUP 0, BYTE 0*'/
0423
0424            data                vicache_parity_error_register(6)
0425            1 /'PARITY OK "CDM" GROUP 0, BYTE 3*'/
0426
0427            data                vicache_parity_error_register(7)
0428            1 /'PARITY OK "CDM" GROUP 0, BYTE 2*'/
0429
0430            data                vicache_parity_error_register(8)
0431            1 /'PARITY OK "CDM" GROUP 0, BYTE 1*'/
0432
0433            data                vicache_parity_error_register(9)
0434            1 /'PARITY OK "CDM" GROUP 0, BYTE 0*'/
0435
0436            data                vicache_parity_error_register(10)
0437            1 /'PARITY OK "CDM" GROUP 1, BYTE 3*'/
0438
0439            data                vicache_parity_error_register(11)
0440            1 /'PARITY OK "CDM" GROUP 1, BYTE 2*'/
0441
0442            data                vicache_parity_error_register(12)
0443            1 /'PARITY OK "CDM" GROUP 1, BYTE 1*'/
0444
0445            data                vicache_parity_error_register(13)
0446            1 /'PARITY OK "CDM" GROUP 1, BYTE 0*'/
0447
0448            data                vicache_parity_error_register(14)
0449            1 /'CP ERROR*'/
0450
0451            data                vicache_parity_error_register(15)
0452            1 /'ANY ERROR*'/
0453
0454            integer*4           cache_group_disabled
0455
0456            integer*4           mnemonic
0457
0458            integer*4           ast_level
0459
0460            integer*4           arithmetic_trap_code
0461
0462            integer*4           last_tb_write_pulse
0463
0464            equivalence         (cache_group_disabled,mnemonic,ast_level,
0465            1 arithmetic_trap_code,last_tb_write_pulse)
0466
0467            logical*1           diagnostic_mode
0468
0469            integer*4           lib$extzv
0470
0471            integer*4           compress4
0472
0473            integer*4           compressc
```

```
0474
0475
0476
0477            call movc3 (%val(48),start_of_cpu_specific_stuff,buffer)
0478
0479            diagnostic_mode = .false.
0480
0481            if (iand(tber0_register,'001f001e'x) .ne. 0) diagnostic_mode = .true.
0482
0483            call linchk (lun,2)
0484
0485            write(lun,10) 'EXCEPTION PC',exception_pc
0486     10     format(/' ',t8,a,t24,z8.8)
0487
0488            if (.not. diagnostic_mode) then
0489
0490            call vaxpsl (lun,exception_psl)
0491            else
0492
0493            call linchk (lun,1)
0494
0495            write(lun,10) 'ERROR PSL',exception_psl
0496            endif
0497
0498            call linchk (lun,1)
0499
0500            write(lun,10) 'SUMMARY CODE',summary_code
0501
0502            if (.not. diagnostic_mode) then
0503
0504            call linchk (lun,1)
0505
0506            if (summary_parameter .eq. '00'x) then
0507
0508            call linchk (lun,1)
0509
0510            write(lun,15) 'CP READ TIMEOUT'
0511     15     format(' ',t40,a,/,' ',t40,' - OR ERROR CONFIRMATION FAULT')
0512
0513            else if (summary_parameter .eq. '02'x) then
0514
0515            write(lun,20) 'CP TB'
0516     20     format(' ',t40,a,' PARITY ERROR FAULT')
0517
0518            else if (summary_parameter .eq. '03'x) then
0519
0520            write(lun,20) 'CP CACHE'
0521
0522            else if (summary_parameter .eq. '05'x) then
0523
0524            write(lun,25) 'CP'
0525     25     format(' ',t40,a,' READ DATA SUBSTITUTE FAULT')
0526
0527            else if (summary_parameter .eq. '0a'x) then
0528
0529            write(lun,20) 'IB TB'
0530
```

```
0531            else if (summary_parameter .eq. '0c'x) then
0532
0533            write(lun,25) 'IB'
0534
0535            else if (summary_parameter .eq. '0d'x) then
0536
0537            call linchk (lun,1)
0538
0539            write(lun,15) 'IB READ TIMEOUT'
0540
0541            else if (summary_parameter .eq. '0f'x) then
0542
0543            write(lun,20) 'IB CACHE'
0544
0545            else if (summary_parameter .eq. 'f1'x) then
0546
0547            write(lun,30) 'CONTROL STORE'
0548    30      format(' ',t40,a,' PARITY ERROR ABORT')
0549
0550            else if (summary_parameter .eq. 'f2'x) then
0551
0552            write(lun,30) 'CP TB'
0553
0554            else if (summary_parameter .eq. 'f3'x) then
0555
0556            write(lun,30) 'CP CACHE'
0557
0558            else if (summary_parameter .eq. 'f4'x) then
0559
0560            call linchk (lun,1)
0561
0562            write(lun,35) 'CP READ TIMEOUT',' - OR ERROR CONFIRMATION ABORT'
0563    35      format(' ',t40,a,/,' ',t40,a)
0564
0565            else if (summary_parameter .eq. 'f5'x) then
0566
0567            write(lun,40) 'CP READ DATA SUBSTITUTE ABORT'
0568    40      format(' ',t40,a)
0569
0570            else if (summary_parameter .eq. 'f6'x) then
0571
0572            write(lun,35) 'MICROCODE ''NOT SUPPOSED',' - TO BE HERE'' ABORT'
0573            else
0574
0575            write(lun,40) 'UNKNOWN SUMMARY CODE'
0576            endif
0577
0578            if (summary_parameter .eq. 'f1'x) then
0579
0580            mnemonic = lib$extzv(0,8,cs_parity_error_opcode)
0581
0582            call linchk (lun,1)
0583
0584            write(lun,42) opcodes(mnemonic)
0585    42      format(' ',t40,'OPCODE MNEMONIC = ',a<compressc (opcodes(mnemonic))>)
0586            endif
0587
```

```
0588            if (summary_parameter .eq. '03'x
0589            1 .or.
0590            1 summary_parameter .eq. '0f'x
0591            1 .or.
0592            1 summary_parameter .eq. 'f3'x) then
0593
0594            cache_group_disabled = lib$extzv(0,8,disabled_cache_group_flags)
0595
0596            if (cache_group_disabled .ne. 0) then
0597
0598            call linchk (lun,1)
0599
0600            write(lun,45) 'CACHE GROUP ',cache_group_disabled,'. DISABLED BY VMS'
0601    45      format(' ',t40,a,i<compress4 (cache_group_disabled)>,a)
0602            endif
0603            endif
0604            endif
0605
0606            call linchk 'lun,1)
0607
0608            write(lun,50) 'CES',cpu_error_status_register
0609    50      format(' ',t8,a,t24,z8.8)
0610
0611            if (.not. diagnostic_mode) then
0612
0613            ast_level = lib$extzv(1,2,cpu_error_status_register)
0614
0615            call linchk (lun,1)
0616
0617            write(lun,55) modes(ast_level)
0618    55      format(' ',t40,a<compressc (modes(ast_level))>,' AST PENDING')
0619
0620            call output (lun,cpu_error_status_register,v1cpu_error_status_register,
0621            1 3,3,3,'0')
0622
0623            arithmetic_trap_code = lib$extzv(4,3,cpu_error_status_register)
0624
0625            if (arithmetic_trap_code .ne. 0) then
0626
0627            call linchk (lun,1)
0628
0629            write(lun,60) traps(arithmetic_trap_code)
0630    60      format(' ',t40,a<compressc (traps(arithmetic_trap_code))>)
0631            endif
0632
0633            call output (lun,cpu_error_status_register,v2cpu_error_status_register,
0634            1 7,7,16,'0')
0635            endif
0636
0637            call linchk (lun,6)
0638
0639            write(lun,65) 'MICRO PC',trapped_upc,'VA/VIBA',va_viba_register,
0640            1 'D REGISTER',d_register
0641    65      format(3(' ',t8,a,t24,z8.8,:/))
0642
0643            call linchk (lun,1)
0644
```

```
0645            write(lun,50) 'TBER0',tber0_register
0646
0647            if (.not. diagnostic_mode) then
0648
0649            call output (lun,tber0_register,v1tber0_register,0,0,0,'0')
0650
0651            call output (lun,tber0_register,v2tber0_register,6,6,9,'0')
0652
0653            call linchk (lun,1)
0654
0655            write(lun,70) 'MICRO CODE "MCT" FIELD = ',
0656           1 lib$extzv(10,4,tber0_register)
0657     70     format(' ',t40,a,Z2.2)
0658
0659            call output (lun,tber0_register,v3tber0_register,14,14,15,'0')
0660            else
0661
0662            call linchk (lun,1)
0663
0664            write(lun,40) 'DIAGNOSTIC MODE'
0665            endif
0666
0667            call linchk (lun,1)
0668
0669            write(lun,50) 'TBER1',tber1_register
0670
0671            if (.not. diagnostic_mode) then
0672
0673            if (iand(tber1_register,'00000010'x) .eq. 0) then
0674
0675            call output (lun,tber1_register,v1tber1_register,0,0,3,'0')
0676            else
0677
0678            write(lun,40) 'BAD "IPA"'
0679            endif
0680
0681            last_tb_write_pulse = lib$extzv(5,2,tber1_register)
0682
0683            call linchk (lun,1)
0684
0685            if (last_tb_write_pulse .eq. 1) then
0686
0687            write(lun,75) 'TO GROUP 0'
0688     75     format(' ',t40,'LAST TB WRITE PULSE ',a)
0689
0690            else if (last_tb_write_pulse .eq. 2) then
0691
0692            write(lun,75) 'TO GROUP 1'
0693
0694            else if (last_tb_write_pulse .eq. 3) then
0695
0696            write(lun,75) 'UNPREDICTABLE'
0697            endif
0698
0699            call output (lun,tber1_register,v2tber1_register,8,8,20,'0')
0700            endif
0701
```

```
0702            if (.not. diagnostic_mode) then
0703
0704            if (iand(sbi_error_register,'00001000'x) .ne. 0) then
0705
0706            call sbi_timeout (lun,sbi_timeout_address_register)
0707
0708            goto 80
0709            endif
0710            endif
0711
0712            call linchk (lun,1)
0713
0714            write(lun,50) 'SBITA',sbi_timeout_address_register
0715
0716    80      call linchk (lun,1)
0717
0718            write(lun,50) 'CACHE PE REG',cache_parity_error_register
0719
0720            if (.not. diagnostic_mode) then
0721
0722            call output (lun,cache_parity_error_register,
0723            1 vlcache_parity_error_register,0,0,15,'0')
0724            endif
0725
0726            if (.not. diagnostic_mode) then
0727
0728            call sbi_error (lun,sbi_error_register)
0729            else
0730
0731            call linchk (lun,1)
0732
0733            write(lun,50) 'SBIER',sbi_error_register
0734            endif
0735
0736            return
0737
0738            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	TBER0_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	32	(0:15)
2-00000034	CHAR	V1CPU_ERROR_STATUS_REGISTER	27	(3:3)
2-000001CB	CHAR	V1TBER0_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	V2TBER0_REGISTER	60	(6:9)
2-000002CA	CHAR	V2TBER1_REGISTER	312	(8:20)
2-00000220	CHAR	V3TBER0_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						

```
0001
0002
0003
0004
0005     c++
0006     c       Functional description:
0007     c
0008     c       This routine displays machine checks for the 11/750.  The format
0009     c       of the entry is as follows.
0010     c
0011     c       +-----------------------------------+
0012     c       |           error code              |
0013     c       +-----------------------------------+
0014     c       |          va last fetch            |
0015     c       +-----------------------------------+
0016     c       |           pc at error             |
0017     c       +-----------------------------------+
0018     c       |     memory data last reference    |
0019     c       +-----------------------------------+
0020     c       |        saved mode register        |
0021     c       +-----------------------------------+
0022     c       |       write vector register       |
0023     c       +-----------------------------------+
0024     c       |              tber                 |
0025     c       +-----------------------------------+
0026     c       |              caer                 |
0027     c       +-----------------------------------+
0028     c       |         bus error register        |
0029     c       +-----------------------------------+
0030     c       |           error summary           |
0031     c       +-----------------------------------+
0032     c       |           exception pc            |
0033     c       +-----------------------------------+
0034     c       |           exception psl           |
0035     c       +-----------------------------------+
0036     c--
0037
0038
0039
0040
0041             subroutine mcheck_750 (lun,start_of_cpu_specific_stuff)
0042
0043
0044
0045             include 'src$:opcodes.for /nolist'
0083
0084
0085
0086             byte            lun
0087
0088             integer*4       start_of_cpu_specific_stuff
0089
0090             integer*4       buffer(12)
0091
0092             integer*4       error_code
0093
0094             integer*4       va_last_fetch
```

```
0095
0096            integer*4       pc_at_error
0097
0098            integer*4       memory_data_last_reference
0099
0100            integer*4       saved_mode_register
0101
0102            integer*4       write_vector_reg'ster
0103
0104            integer*4       tber_register
0105
0106            integer*4       caer_register
0107
0108            integer*4       bus_error_register
0109
0110            integer*4       machine_check_error_summary
0111
0112            integer*4       exception_pc
0113
0114            integer*4       exception_psl
0115
0116            equivalence     (buffer(1),error_code)
0117
0118            equivalence     (buffer(2),va_last_fetch)
0119
0120            equivalence     (buffer(3),pc_at_error)
0121
0122            equivalence     (buffer(4),memory_data_last_reference)
0123
0124            equivalence     (buffer(5),saved_mode_register)
0125
0126            equivalence     (buffer(6),write_vector_register)
0127
0128            equivalence     (buffer(7),tber_register)
0129
0130            equivalence     (buffer(8),caer_register)
0131
0132            equivalence     (buffer(9),bus_error_register)
0133
0134            equivalence     (buffer(10),machine_check_error_summary)
0135
0136            equivalence     (buffer(11),exception_pc)
0137
0138            equivalence     (buffer(12),exception_psl)
0139
0140            integer*4       mnemonic
0141
0142            integer*4       compressc
0143
0144            logical*1       diagnostic_mode
0145
0146
0147
0148
0149            call movc3 (%val(48),start_of_cpu_specific_stuff,buffer)
0150
0151            diagnostic_mode = .false.
```

```
0152
0153            call linchk (lun,2)
0154
0155            write(lun,10) 'EXCEPTION PC',exception_pc
0156    10      format(/' ',t8,a,t24,z8.8)
0157
0158            call vaxpsl (lun,exception_psl)
0159
0160            call linchk (lun,3)
0161
0162            write(lun,15) 'SUMMARY CODE',error_code
0163    15      format(/' ',t8,a,t24,z8.8)
0164
0165            if (.not. diagnostic_mode) then
0166
0167            if (error_code .eq. 1) then
0168
0169            write(lun,20) 'CONTROL STORE PARITY ERROR'
0170    20      format(' ',t40,a)
0171
0172            else if (error_code .eq. 2) then
0173
0174            write(lun,20) 'TRANSLATION BUFFER OR BUS ERROR'
0175
0176            else if (error_code .eq. 3) then
0177
0178            write(lun,20) 'CACHE PARITY ERROR'
0179
0180            else if (error_code .eq. 6) then
0181
0182            write(lun,20) 'MICRO CODE LOST'
0183
0184            else if (error_code .eq. 7) then
0185
0186            write(lun,20) 'IRD ROM LOST'
0187            else
0188
0189            write(lun,20) 'UNKNOWN SUMMARY CODE'
0190            endif
0191            endif
0192
0193            call linchk (lun,4)
0194
0195            write(lun,25) 'VA LAST REF',va_last_fetch,'PC AT ERROR',pc_at_error,
0196           1 'MDR',memory_data_last_reference,'SMR',saved_mode_register
0197    25      format(4(' ',t8,a,t24,z8.8,:/))
0198
0199            if (.not. diagnostic_mode) then
0200
0201            call cmier_1916 (lun,ishft(iand(saved_mode_register,'0000000f'x),16))
0202            endif
0203
0204            call linchk (lun,1)
0205
0206            write(lun,30) 'RLTO',write_vector_register
0207    30      format(' ',t8,a,t24,z8.8)
0208
```

```
0209            if (.not. diagnostic_mode) then
0210
0211            call cmier_1512 (lun,ishft(iand(write_vector_register,'00000001'x),12))
0212            endif
0213
0214            call linchk (lun,1)
0215
0216            write(lun,30) 'TBER',tber_register
0217
0218            if (.not. diagnostic_mode) then
0219
0220            call cmier_118 (lun,ishft(iand(tber_register,'0000000f'x),8))
0221
0222            if (iand(tber_register,'00010000'x) .ne. 0) then
0223
0224            call linchk (lun,1)
0225
0226            write(lun,35) 'TB GROUP 0 DISABLED (VMS)'
0227    35      format(' ',t40,a)
0228
0229            else if (iand(tber_register,'00020000'x) .ne. 0) then
0230
0231            write(lun,35) 'TB GROUP 1 DISABLED (VMS)'
0232            endif
0233            else
0234
0235            call linchk (lun,1)
0236
0237            write(lun,37) 'DIAGNOSTIC MODE'
0238    37      format(' ',t40,a)
0239            endif
0240
0241            if (.not. diagnostic_mode) then
0242
0243            call comet_caer (lun,iand(caer_register,'fffcffff'x))
0244
0245            if (error_code .eq. 3) then
0246
0247            if (iand(caer_register,'00010000'x) .ne. 0) then
0248
0249            write(lun,35) 'CACHE DISABLED (VMS)'
0250            endif
0251            endif
0252            else
0253
0254            call linchk (lun,1)
0255
0256            write(lun,10) 'CAER',caer_register
0257            endif
0258
0259            call linchk (lun,1)
0260
0261            write(lun,30) 'BER',bus_error_register
0262
0263            if (.not. diagnostic_mode) then
0264
0265            call cmier_30 (lun,bus_error_register)
```

```
0266
0267            call comet_mcesr (lun,machine_check_error_summary)
0268
0269            if (error_code .eq. 1
0270            1 .or.
0271            1 error_code .eq. 6
0272            1 .or.
0273            1 error_code .eq. 7) then
0274
0275            mnemonic = lib$extzv(16,8,machine_check_error_summary)
0276
0277            call linchk (lun,1)
0278
0279            write(lun,40) opcodes(mnemonic)
0280    40      format(' ',t40,'OPCODE MNEMONIC = ',a<compressc (opcodes(mnemonic))>)
0281            endif
0282            else
0283
0284            call linchk (lun,1)
0285
0286            write(lun,10) 'MCESR',machine_check_error_summary
0287            endif
0288
0289            return
0290
0291            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-00000008@ 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                  VAXPSL
```

```
0001
0002
0003
0004     c++
0005     c       Functional description:
0006     c
0007     c       This routine displays machine checks for the 11/7zz.  The format
0008     c       of the entry is as follows.
0009     c
0010     c       +-----------------------------------+
0011     c       |      machine check type code      |
0012     c       +-----------------------------------+
0013     c       |           first parameter         |
0014     c       +-----------------------------------+
0015     c       |          second parameter         |
0016     c       +-----------------------------------+
0017     c       |            exception pc           |
0018     c       +-----------------------------------+
0019     c       |           exception psl           |
0020     c       +-----------------------------------+
0021     c--
0022
0023
0024
0025
0026             subroutine mcheck_7zz (lun,start_of_cpu_specific_stuff)
0027
0028
0029
0030
0031             byte            lun
0032
0033             integer*4       start_of_cpu_specific_stuff
0034
0035             integer*4       buffer(5)
0036
0037             integer*4       machine_check_type_code
0038
0039             integer*4       first_parameter
0040
0041             integer*4       second_parameter
0042
0043             integer*4       exception_pc
0044
0045             integer*4       exception_psl
0046
0047             equivalence     (buffer(1),machine_check_type_code)
0048
0049             equivalence     (buffer(2),first_parameter)
0050
0051             equivalence     (buffer(3),second_parameter)
0052
0053             equivalence     (buffer(4),exception_pc)
0054
0055             equivalence     (buffer(5),exception_psl)
0056
0057             integer*4       error_pfn
```

```
0058
0059            integer*4       compress4
0060
0061            character*21    vlfpa_error_bits(0:2)
0062
0063            data            vlfpa_error_bits(0)
0064            1 /'PARITY ERROR*'/
0065
0066            data            vlfpa_error_bits(1)
0067            1 /'GROUP 0 PARITY ERROR*'/
0068
0069            data            vlfpa_error_bits(2)
0070            1 /'GROUP 1 PARITY ERROR*'/
0071
0072
0073
0074
0075            call movc3 (%val(20),start_of_cpu_specific_stuff,buffer)
0076
0077            call linchk (lun,2)
0078
0079            write(lun,10) 'EXCEPTION PC',exception_pc
0080    10      format(/' ',t8,a,t24,z8.8)
0081
0082            call vaxpsl (lun,exception_psl)
0083
0084            call linchk (lun,3)
0085
0086            write(lun,10) 'SUMMARY CODE',machine_check_type_code
0087
0088            if (machine_check_type_code .eq. 0) then
0089
0090            write(lun,20) 'MICRO CODE SHOULDN''T BE HERE'
0091    20      format(' ',t40,a)
0092
0093            if (first_parameter .eq. 2) then
0094
0095            call linchk (lun,2)
0096
0097            write(lun,25) first_parameter,'CANNOT WRITE BACK PTE <M> BIT'
0098    25      format(' ',t8,'1ST PARAMETER',t24,z8.8,:/,:' ',:t40,:a)
0099
0100            else if (first_parameter .eq. 3) then
0101
0102            call linchk (lun,2)
0103
0104            write(lun,25) first_parameter,'BAD 8085 INTERRUPT'
0105            else
0106
0107            goto 50
0108            endif
0109
0110            goto 60
0111
0112            else if (machine_check_type_code .eq. 1) then
0113
0114            write(lun,20) 'TRANSLATION BUFFER PARITY ERROR'
```

```
0115
0116            call linchk (lun,2)
0117
0118            write(lun,25) first_parameter,'PTE FROM TRANSLATION BUFFER'
0119
0120            call linchk (lun,2)
0121
0122            write(lun,30) second_parameter,'PTE FROM MEMORY'
0123    30      format(' ',t8,'2ND PARAMETER',t24,z8.8,:/,:t40,:a)
0124
0125            return
0126
0127            else if (machine_check_type_code .eq. 3) then
0128
0129            write(lun,20) 'IMPOSSIBLE VALUE IN MEMORY CSR'
0130
0131            call linchk (lun,2)
0132
0133            write(lun,25) first_parameter,'VIRTUAL ADDRESS REFERENCED'
0134
0135            goto 60
0136
0137            else if (machine_check_type_code .eq. 4) then
0138
0139            write(lun,20) 'FAST INTERRUPT WITHOUT SUPPORT'
0140
0141            if (first_parameter .ne. 0
0142            1 .or.
0143            1 second_parameter .ne. 0) goto 50
0144
0145            return
0146
0147            else if (machine_check_type_code .eq. 5) then
0148
0149            write(lun,20) 'FPA PARITY ERROR'
0150
0151            call output (lun,first_parameter,v1fpa_error_bits,0,0,2,'0')
0152
0153            goto 60
0154
0155            else if (machine_check_type_code .eq. 6) then
0156
0157            write(lun,20) 'ERROR ON "SPTE" READ'
0158
0159            call linchk (lun,2)
0160
0161            write(lun,25) first_parameter,'PHYSICAL ADDRESS OF "SPTE"'
0162
0163            goto 60
0164
0165            else if (machine_check_type_code .eq. 7) then
0166
0167            write(lun,20) 'UNCORRECTABLE ECC ERROR'
0168
0169            call linchk (lun,2)
0170
0171            write(lun,35) first_parameter
```

```
0172     35       format(' ',t8,'1ST PARAMETER',t24,z8.8)
0173
0174              error_pfn = lib$extzv(9,15,first_parameter)
0175
0176              write(lun,37) error_pfn
0177     37       format(' ',t40,'PAGE #',i<compress4 (error_pfn)>,'. IN ERROR')
0178
0179              goto 60
0180
0181              else if (machine_check_type_code .eq. 8) then
0182
0183              write(lun,20) 'NON-EXISTENT MEMORY'
0184
0185              call linchk (lun,1)
0186
0187              write(lun,40) first_parameter
0188     40       format(' ',t8,'1ST PARAMETER',t24,z8.8,/,
0189              1 t40,'PHYSICAL ADDRESS REFERENCED')
0190
0191              goto 60
0192
0193              else if (machine_check_type_code .eq. 9
0194              1 .or.
0195              1 machine_check_type_code .eq. 10
0196              1 .or.
0197              1 machine_check_type_code .eq. 11) then
0198
0199              if (machine_check_type_code .ne. 11) then
0200
0201              write(lun,20) 'ILLEGAL I/O SPACE REFERENCE'
0202              else
0203
0204              write(lun,20) 'ILLEGAL UNIBUS REFERENCE'
0205              endif
0206
0207              call linchk (lun,1)
0208
0209              write(lun,40) first_parameter
0210
0211              goto 60
0212              else
0213
0214              write(lun,20) 'UNKNOWN SUMMARY CODE'
0215              endif
0216
0217     50       continue
0218
0219              call linchk (lun,2)
0220
0221              write(lun,55) first_parameter,second_parameter
0222     55       format(' ',t8,'1ST PARAMETER',t24,z8.8,/,' ',t8,'2ND PARAMETER',t24,
0223              1 z8.8)
0224
0225              return
0226
0227     60       continue
0228
```

```
0229            if (second_parameter .ne. 0) then
0230
0231            call linchk (lun,1)
0232
0233            write(lun,30) second_parameter
0234            endif
0235
0236            return
0237
0238            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

```
0001
0002
0003     c++
0004     c       Functional description:
0005     c
0006     c       This routine displays machine checks for the UVAX1.  The format
0007     c       of the entry is as follows.
0008     c
0009     c       +-----------------------------------+
0010     c       |        machine check code         |
0011     c       +-----------------------------------+
0012     c       |           first parameter         |
0013     c       +-----------------------------------+
0014     c       |          second parameter         |
0015     c       +-----------------------------------+
0016     c       |            exception pc           |
0017     c       +-----------------------------------+
0018     c       |            exception psl          |
0019     c       +-----------------------------------+
0020     c--
0021
0022
0023
0024
0025             subroutine mcheck_uv1 (lun,start_of_cpu_specific_stuff)
0026
0027
0028
0029
0030             implicit        none
0031
0032             byte            lun
0033
0034             integer*4       start_of_cpu_specific_stuff
0035
0036             integer*4       buffer(5)
0037
0038             integer*4       machine_check_code
0039
0040             integer*4       parameter_1
0041
0042             integer*4       parameter_2
0043
0044             integer*4       exception_pc
0045
0046             integer*4       exception_psl
0047
0048             equivalence     (buffer(1),machine_check_code)
0049
0050             equivalence     (buffer(2),parameter_1)
0051
0052             equivalence     (buffer(3),parameter_2)
0053
0054             equivalence     (buffer(4),exception_pc)
0055
0056             equivalence     (buffer(5),exception_psl)
0057
```

```
0058            integer*4       compress4
0059
0060            character*27    address_previous_function
0061
0062            data            address_previous_function
0063            1 /'PHYS ADDR PREVIOUS FUNCTION'7
0064
0065            character*26    address_current_function
0066
0067            data            address_current_function
0068            1 /'PHYS ADDR CURRENT FONCTION'7
0069
0070
0071            character*25    physical_pte_address
0072
0073            data            physical_pte_address
0074            1 /'PHYSICAL ADDRESS OF "PTE"'/
0075
0076            character*25    virtual_pte_address
0077
0078            data            virtual_pte_address
0079            1 /'VIRTUAL ADDRESS FOR "PTE"'/
0080
0081
0082
0083            call movc3 (%val(20),start_of_cpu_specific_stuff,buffer)
0084
0085            call linchk (lun,2)
0086
0087            write(lun,5) 'EXCEPTION PC',exception_pc
0088    5       format(' ',t8,a,t24,z8.8)
0089
0090            call vaxpsl (lun,exception_psl)
0091
0092            call linchk (lun,3)
0093
0094            write(lun,5) 'MACHINE CHECK',machine_check_code
0095
0096            if (machine_check_code .eq. 1) then
0097
0098            write(lun,10) 'MEMORY CONTROLLER BUGCHECK'
0099    10      format(' ',t40,a)
0100
0101            call parameter_one (lun,parameter_1,.true.)
0102
0103            call linchk (lun,1)
0104
0105            write(lun,10) address_previous_function
0106
0107            call parameter_two (lun,parameter_2,.true.)
0108
0109            write(lun,10) address_current_function
0110
0111            else if (machine_check_code .eq. 2) then
0112
0113            write(lun,10) 'UNRECOVERABLE READ ERROR'
0114
```

```
0115            call parameter_one (lun,parameter_1,.true.)
0116
0117            call linchk (lun,1)
0118
0119            write(lun,10) 'PHYSICAL PAGE ADDRESS OF ERROR'
0120
0121            call parameter_two (lun,parameter_2,.true.)
0122
0123            call linchk (lun,1)
0124
0125            write(lun,10) address_current_function
0126
0127            else if (machine_check_code .eq. 3) then
0128
0129            write(lun,10) 'NON-EXISTENT MEMORY'
0130
0131            call parameter_one (lun,parameter_1,.true.)
0132
0133            call linchk (lun,1)
0134
0135            write(lun,10) 'PHYSICAL ADDRESS OF ERROR'
0136
0137            call parameter_two (lun,parameter_2,.true.)
0138
0139            write(lun,10) address_current_function
0140
0141            else if (machine_check_code .eq. 4) then
0142
0143            write(lun,10) 'ILLEGAL OPERATION'
0144
0145            call parameter_one (lun,parameter_1,.true.)
0146
0147            call linchk (lun,1)
0148
0149            write(lun,10) 'UNALIGNED I/O REFERENCE'
0150
0151            call parameter_two (lun,parameter_2,.true.)
0152
0153            call linchk (lun,1)
0154
0155            write(lun,10) address_current_function
0156
0157            else if (machine_check_code .eq. 5) then
0158
0159            write(lun,10) 'FATAL PAGE TABLE READ ERROR'
0160
0161            call parameter_one (lun,parameter_1,.true.)
0162
0163            call linchk (lun,1)
0164
0165            write(lun,10) physical_pte_address
0166
0167            call parameter_two (lun,parameter_2,.true.)
0168
0169            call linchk (lun,1)
0170
0171            write(lun,10) virtual_pte_address
```

```
0172
0173            else if (machine_check_code .eq. 6) then
0174
0175            write(lun,10) 'FATAL PAGE TABLE WRITE ERROR'
0176
0177            call parameter_one (lun,parameter_1,.true.)
0178
0179            call linchk (lun,1)
0180
0181            write(lun,10) physical_pte_address
0182
0183            call parameter_two (lun,parameter_2,.true.)
0184
0185            call linchk (lun,1)
0186
0187            write(lun,10) virtual_pte_address
0188
0189            else if (machine_check_code .eq. 7) then
0190
0191            write(lun,10) 'CONTROL STORE PARITY ERROR'
0192
0193            call parameter_one (lun,parameter_1,.false.)
0194
0195            call parameter_two (lun,parameter_2,.false.)
0196
0197            else if (machine_check_code .eq. 8) then
0198
0199            write(lun,10) 'MICRO MACHINE BUGCHECK'
0200
0201            call parameter_one (lun,parameter_1,.false.)
0202
0203            call parameter_two (lun,parameter_2,.false.)
0204
0205            else if (machine_check_code .eq. 9) then
0206
0207            write(lun,10) ''Q-BUS'' VECTOR READ ERROR'
0208
0209            call parameter_one (lun,parameter_1,.false.)
0210
0211            call parameter_two (lun,parameter_2,.false.)
0212
0213            else if (machine_check_code .eq. 10) then
0214
0215            write(lun,10) 'WRITE PARAMETER ERROR'
0216
0217            call parameter_one (lun,parameter_1,.true.)
0218
0219            call linchk (lun,1)
0220
0221            write(lun,10) 'VIRTUAL ADDRESS BEING WRITTEN'
0222
0223            call parameter_two (lun,parameter_2,.false.)
0224
0225        c
0226        c       The IF-THEN-ELSE should be expanded at this point to add
0227        c       additional mcheck types for uv1.
0228        c
```

```
0229
0230            else
0231
0232            write(lun,15) 'TYPE CODE #',machine_check_code
0233    15      format(' ',t40,a,i<compress4 (machine_check_code)>,'.')
0234
0235            call parameter_one (lun,parameter_1,.true.)
0236
0237            call parameter_two (lun,parameter_2,.true.)
0238            endif
0239
0240            return
0241
0242            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)

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

000
000
000
000
000
000
000
000
000
001
001
001
001
001
001
001
001
001
001
001
002
002
002
002
002
002
002
002
002
002
002
002
002
003
003
003
003
003
003
003
003
004
004
004
004
004
004
004
004
004
004
004
004
005
005
005
005
005
005
005
005
005
005

```
0001
0002
0003
0004
0005            subroutine parameter_one (lun,parameter_1,expand)
0006
0007
0008
0009
0010            implicit        none
0011
0012            byte            lun
0013
0014            integer*4       parameter_1
0015
0016            logical*1       expand
0017
0018
0019
0020
0021            if (.not. expand) then
0022
0023            if (parameter_1 .eq. 0) goto 20
0024            endif
0025
0026            call linchk (lun,1)
0027
0028            write(lun,10) parameter_1
0029    10      format(' ',t8,'PARAMETER 1',t24,z8.8)
0030
0031    20      return
0032
0033            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

011
011
011
011
011
012
012
012
012
012
012
012
018
031
031
031
031
031
031
031
031
032
032
032
032
032
032
032
032
032
032
033
033
033
033
033
033
033
033
033
034
034
034
034
034
034
034
034
034
034
034
035
035
035
035
035
035
035
035
035

```
0001
0002
0003
0004
0005            subroutine parameter_two (lun,parameter_2,expand)
0006
0007
0008
0009
0010            implicit        none
0011
0012            byte            lun
0013
0014            integer*4       parameter_2
0015
0016            logical*1       expand
0017
0018
0019
0020
0021            if (.not. expand) then
0022
0023            if (parameter_2 .eq. 0) goto 20
0024            endif
0025
0026            call linchk (lun,1)
0027
0028            write(lun,10) parameter_2
0029    10      format(' ',t8,'PARAMETER 2',t24,z8.8)
0030
0031    20      return
0032
0033            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

MOUVXX.LIS

MCHK_DISP.LIS