```
EEEEEEEEEEEEEEE   RRRRRRRRRRR      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
EEEEEEEEEEEEEEE   RRR          RRR   FFF
```

FILEID**ML11

```
MM      MM  LL              11          11
MM      MM  LL              11          11
MMMM  MMMM  LL            1111        1111
MMMM  MMMM  LL            1111        1111
MM  MM  MM  LL              11          11
MM  MM  MM  LL              11          11
MM      MM  LL              11          11
MM      MM  LL              11          11
MM      MM  LL              11          11
MM      MM  LL              11          11
MM      MM  LL              11          11          ....
MM      MM  LL              11          11          ....
MM      MM  LLLLLLLLLL    111111      111111        ....
MM      MM  LLLLLLLLLL    111111      111111        ....

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
```

ML11

ARRA

LABE

1-
1-
1-
1-

FUNC

Ty

```
0001            SUBROUTINE ML11 (lun)
0002    C
0003    C Version:      'V04-000'
0004    C
0005    C*****************************************************************************
0006    C*                                                                           *
0007    C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0008    C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0009    C*  ALL RIGHTS RESERVED.                                                     *
0010    C*                                                                           *
0011    C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0012    C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0013    C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0014    C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0015    C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0016    C*  TRANSFERRED.                                                             *
0017    C*                                                                           *
0018    C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0019    C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0020    C*  CORPORATION.                                                             *
0021    C*                                                                           *
0022    C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0023    C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0024    C*                                                                           *
0025    C*                                                                           *
0026    C*****************************************************************************
0027    C
0028    C
0029    C
0030    C       Author: Sharon Reynolds                 Creation date:  12-Dec-80
0031    C
0032    C
0033    C++
0034    C       Functional description:
0035    C
0036    C       This module builds the error log report for the ML11 solid-state
0037    C       disk.
0038    C
0039    C       Modified by:
0040    C
0041    C       V03-003 SAR0227         Sharon A. Reynolds,     28-Mar-1984
0042    C               Changed the call to UCB$L_OWNUIC to ORB$L_OWNER.
0043    C
0044    C       V03-002 SAR0114         Sharon A. Reynolds,     23-Jun-1983
0045    C               Changed the carriage control in the 'format' statements
0046    C               for use with ERF.
0047    C
0048    C       V03-001 SAR0048         Sharon A. Reynolds,     13-Jun-1983
0049    C               Removed brif/cryptic support.
0050    C
0051    C       v02-003 BP0003          Brian Porter,           18-NOV-1981
0052    C               Added new mba code. Minor edit.
0053    C
0054    C       v02-002 BP0002          Brian Porter,           05-NOv-1981
0055    C               Added 'device attention' support.
0056    C
0057    C       v02-001 BP0001          Brian Porter,           01-JUL-1981
```

```
0058    C               Added call to DHEAD and LOGGER.  Added DIAGNOSTIC_MODE.
0059    C--
0060    C**
0061
0062
0063            Include 'SRC$:MSGHDR.FOR /NOLIST'
0122            Include 'SRC$:DEVERR.FOR /NOLIST'
0223
0224
0225            Byte            lun
0226
0227
0228            Parameter       timeout = 96
0229            Parameter       xfer_cmd = 20
0230
0231
0232            Integer*4       COMPRESSC
0233            Integer*4       COMPRESS4
0234
0235            Integer*4       field
0236
0237            integer*4       adapter_registers(7)
0238
0239            integer*4       selected_map_register
0240
0241            Integer*4       irp_flag
0242            Integer*4       mlcs1
0243            Integer*4       mlds
0244            Integer*4       mler
0245            Integer*4       mlmr
0246            Integer*4       mlas
0247            Integer*4       mlda
0248            Integer*4       mldt
0249            Integer*4       mlsn
0250            Integer*4       mle1
0251            Integer*4       mle2
0252            Integer*4       mlee
0253            Integer*4       mlel
0254            Integer*4       prom_dis
0255            Integer*4       prom_rw
0256            Integer*4       drv_func
0257            Integer*4       type
0258            Integer*4       cards
0259            Integer*4       chan
0260            Integer*4       err_func
0261            Integer*4       crc_err
0262            Integer*4       sngl_err
0263            Integer*4       unc_err
0264            Integer*4       wrd
0265
0266            logical*1       diagnostic_mode
0267
0268            Character*7     mlcs1_1(0:0)
0269            Character*16    mlcs1_2(11:11)
0270            Character*14    mlds_1(6:8)
0271            Character*23    mlds_2(10:10)
0272            Character*17    mlds_3(14:15)
```

```
0273            Character*16    mlds_mol(0:1)
0274            Character*24    mler_1(0:3)
0275            Character*15    mler_2(5:6)
0276            Character*23    mler_3(9:10)
0277            Character*21    mler_4(13:15)
0278            Character*18    mlmr_1(1:2)
0279            Character*30    mlmr_2(7:7)
0280            Character*17    xfer_rate(0:3)
0281            Character*16    aray_typ(0:1)
0282
0283    C
0284    C       Make the contents of the register save area in the error log
0285    C       buffer available to this module
0286    C
0287
0288            EQUIVALENCE     (adapter_registers,emb$l_dv_regsav(0))
0289            Equivalence     (irp_flag,EMB$L_DV_REGSAV(7))
0290            Equivalence     (mlcs1,EMB$L_DV_REGSAV(8))
0291            Equivalence     (mlds,EMB$L_DV_REGSAV(9))
0292            Equivalence     (mler,EMB$L_DV_REGSAV(10))
0293            Equivalence     (mlmr,EMB$L_DV_REGSAV(11))
0294            Equivalence     (mlas,EMB$L_DV_REGSAV(12))
0295            Equivalence     (mlda,EMB$L_DV_REGSAV(13))
0296            Equivalence     (mldt,EMB$L_DV_REGSAV(14))
0297            Equivalence     (mlsn,EMB$L_DV_REGSAV(16))
0298            Equivalence     (mle1,EMB$L_DV_REGSAV(17))
0299            Equivalence     (mle2,EMB$L_DV_REGSAV(18))
0300            Equivalence     (mlee,EMB$L_DV_REGSAV(21))
0301            Equivalence     (mlel,EMB$L_DV_REGSAV(22))
0302
0303    C
0304    C       Define text for bits in the ML11 control/status register
0305    C
0306
0307            Data    mlcs1_1(0)      /'GO BIT*'/
0308
0309            Data    mlcs1_2(11)     /'DRIVE AVAILABLE*'/
0310
0311
0312    C
0313    C       Define text for bits in the ML11 drive status register
0314    C
0315
0316            Data    mlds_1(6)       /'VOLUME VALID*'/
0317            Data    mlds_1(7)       /'DRIVE READY*'/
0318            Data    mlds_1(8)       /'DRIVE PRESENT*'/
0319
0320            Data    mlds_2(10)      /'LAST BLOCK TRANSFERRED*'/
0321
0322            Data    mlds_3(14)      /'ERROR*'/
0323            Data    mlds_3(15)      /'ATTENTION ACTIVE*'/
0324
0325            Data    mlds_mol(0)     /'MEDIUM OFF LINE*'/
0326            Data    mlds_mol(1)     /'MEDIUM ON LINE*'/
0327
0328    C
0329    C       Define text for bits in the ML11 error register
```

```
0330    C
0331
0332            Data    mler_1(0)       /'ILLEGAL FUNCTION*'/
0333            Data    mler_1(1)       /'ILLEGAL REGISTER*'/
0334            Data    mler_1(2)       /'REGISTER MODIFY REFUSED*'/
0335            Data    mler_1(3)       /'CONTROL PARITY*'/
0336
0337            Data    mler_2(5)       /'DATA PARITY*'/
0338            Data    mler_2(6)       /'ECC HARD ERROR*'/
0339
0340            Data    mler_3(9)       /'ADDRESS OVERFLOW ERROR*'/
0341            Data    mler_3(10)      /'INVALID ADDRESS ERROR*'/
0342
0343            Data    mler_4(13)      /'OPERATION INCOMPLETE*'/
0344            Data    mler_4(14)      /'DRIVE UNSAFE*'/
0345            Data    mler_4(15)      /'DATA CHECK*'/
0346
0347
0348    C
0349    C       Define text for bits in the ML11 maintenance register
0350    C
0351
0352            Data    mlmr_1(1)       /'ECC DISABLED*'/
0353            Data    mlmr_1(2)       /'DATA CHECK ENABLE*'/
0354
0355            Data    mlmr_2(7)       /'REFRESH RATE DECREASED BY 50%*'/
0356
0357            Data    xfer_rate(0)    /'2.0 MBYTE/SECOND*'/
0358            Data    xfer_rate(1)    /'1.0 MBYTE/SECOND*'/
0359            Data    xfer_rate(2)    /'.5 MBYTE/SECOND*'/
0360            Data    xfer_rate(3)    /'.25 MBYTE/SECOND*'/
0361
0362            Data    aray_typ(0)     /'16K CHIP ARRAYS*'/
0363            Data    aray_typ(1)     /'64K CHIP ARRAYS*'/
0364
0365
0366    C
0367    C       Construct the report header
0368    C
0369
0370            Call FRCTOF (lun)
0371
0372            call dhead1 (lun,'MASSBUS')
0373
0374    C
0375    C       Establish if it was a data transfer type command and get the
0376    C       massbus registers if so
0377    C
0378
0379            diagnostic_mode = .false.
0380
0381            if (lib$extzv(0,1,mlmr) .eq. 1) diagnostic_mode = .true.
0382
0383            if (lib$extzv(3,1,mlmr) .eq. 1) diagnostic_mode = .true.
0384
0385            if (lib$extzv(5,2,mlmr) .ne. 0) diagnostic_mode = .true.
0386
```

```
0387            Drv_func = LIB$EXTZV (1,5,mlcs1)
0388
0389            If (
0390            1 EMB$W_DV_BCNT .ne. 0
0391            1 .and.
0392            1 drv_func .ge. xfer_cmd
0393            1 .and.
0394            1 emb$w_hd_entry .ne. 98
0395            1 ) then
0396
0397            call mba_control_registers (lun,5,adapter_registers,
0398            1 selected_map_register)
0399
0400            call mba_mapping_register (lun,selected_map_register,
0401            1 adapter_registers(6))
0402
0403            if (selected_map_register .gt. 0) then
0404
0405            call mba_mapping_register (lun,(selected_map_register - 1),
0406            1 adapter_registers(7))
0407            endif
0408            Endif
0409
0410
0411    C
0412    C       Decode and output the bits in the control and status register
0413    C
0414
0415            Call LINCHK (lun,2)
0416            Write (lun,20) mlcs1
0417    20      Format (/' ',T8,'MLCS1',T24,Z8.8)
0418
0419            if (.not. diagnostic_mode) then
0420
0421            call mba_status_register16_31 (lun,mlcs1,mlcs1,0)
0422
0423            Call OUTPUT (lun,mlcs1,mlcs1_1,0,0,0,'0')
0424
0425            Call LINCHK (lun,1)
0426            If (drv_func .eq. 0) then
0427            Write (lun,22)
0428    22      Format (' ',T40,'NO-OPERATION')
0429
0430            Else if (drv_func .eq. 4) then
0431            Write (lun,24)
0432    24      Format (' ',T40,'DRIVE CLEAR')
0433
0434            Else if (drv_func .eq. 8) then
0435            Write (lun,26)
0436    26      Format (' ',T40,'READ IN PRESET')
0437
0438            Else if (drv_func .eq. 12) then
0439            Write (lun,28)
0440    28      Format (' ',T40,'SEARCH')
0441
0442            Else if (drv_func .eq. 20) then
0443            Write (lun,30)
```

```
0444     30       Format (' ',T40,'WRITE CHECK')
0445
0446              Else if (dry_func .eq. 24) then
0447              Write (lun,32)
0448     32       Format (' ',T40,'WRITE DATA')
0449
0450              Else if (dry_func .eq. 28) then
0451              Write (lun,34)
0452     34       Format (' ',T40,'READ DATA')
0453              Endif
0454
0455              Call OUTPUT (lun,mlcs1,mlcs1_2,11,11,11,'0')
0456              endif
0457
0458     C
0459     C        Decode and output the bits in the drive status register
0460     C
0461
0462              Call LINCHK (lun,1)
0463              Write (lun,40) mlds
0464     40       Format (' ',T8,'MLDS',T24,Z8.8)
0465
0466              if (.not. diagnostic_mode) then
0467
0468              call mba_status_register16_31 (lun,mlcs1,mlds,1)
0469
0470              Call OUTPUT (lun,mlds,mlds_1,6,6,8,'0')
0471
0472              Call OUTPUT (lun,mlds,mlds_2,10,10,10,'0')
0473
0474              Field = LIB$EXTZV (12,1,mlds)
0475
0476              Call LINCHK (lun,1)
0477              Write (lun,45) mlds_mol(field)
0478     45       Format (' ',T40,A<COMPRESSC (mlds_mol(field))>)
0479
0480              Call OUTPUT (lun,mlds,mlds_3,14,14,15,'0')
0481              endif
0482
0483     C
0484     C        Decode and output the bits in the error register
0485     C
0486
0487              Call LINCHK (lun,1)
0488              Write (lun,50) mler
0489     50       Format (' ',T8,'MLER',T24,Z8.8)
0490
0491              if (.not. diagnostic_mode) then
0492
0493              call mba_status_register16_31 (lun,mlds,mler,1)
0494
0495              Call OUTPUT (lun,mler,mler_1,0,0,3,'0')
0496
0497              Call OUTPUT (lun,mler,mler_2,5,5,6,'0')
0498
0499              Call OUTPUT (lun,mler,mler_3,9,9,10,'0')
0500
```

```
0501            Call OUTPUT (lun,mler,mler_4,13,13,15,'0')
0502            endif
0503
0504    C
0505    C       Decode and output the bits in the maintenance register
0506    C
0507
0508            Call LINCHK (lun,1)
0509            Write (lun,60) mlmr
0510    60      Format (' ',T8,'MLMR',T24,Z8.8)
0511
0512            if (.not. diagnostic_mode) then
0513
0514            call mba_status_register16_31 (lun,mler,mlmr,1)
0515
0516            call output (lun,mlmr,mlmr_1,1,1,2,'0')
0517
0518            Call OUTPUT (lun,mlmr,mlmr_2,7,7,7,'0')
0519
0520            Field = LIB$EXTZV (8,2,mlmr)
0521
0522            Call LINCHK (lun,1)
0523            Write (lun,75) xfer_rate(field)
0524    75      Format (' ',T40,'XFER RATE = ',
0525            1 A<COMPRESSC (xfer_rate(field))>)
0526
0527            Type = LIB$EXTZV (10,1,mlmr)
0528            Cards = LIB$EXTZV (11,5,mlmr)
0529
0530            Call LINCHK (lun,2)
0531
0532            Write (lun,80) aray_typ(type)
0533    80      Format (' ',T40,A<COMPRESSC (aray_typ(type))>)
0534
0535            Write (lun,85) cards
0536    85      Format (' ',T40,I<COMPRESS4 (cards)>,
0537            1 '. ARRAY MODULES')
0538            else
0539
0540            Call LINCHK (lun,1)
0541            Write (lun,70)
0542    70      Format (' ',T40,'DIAGNOSTIC MODE')
0543            Endif
0544
0545    C
0546    C       Decode and output the bits in the attention summary register
0547    C
0548
0549            Call LINCHK (lun,1)
0550            Write (lun,100) mlas
0551    100     Format (' ',T8,'MLAS',T24,Z8.8)
0552
0553            if (.not. diagnostic_mode) then
0554
0555            call mba_status_register16_31 (lun,mlmr,mlas,1)
0556
0557            Do 106, I=0,7
```

```
0558
0559            If (JIAND(mlas,2**I) .ne. 0) then
0560
0561            Call LINCHK (lun,1)
0562            Write (lun,105) I
0563    105     Format (' ',T40,'ATTENTION UNIT ',I1,'.')
0564            Endif
0565
0566    106     Continue
0567            endif
0568
0569
0570    C
0571    C       Decode and output the bits in the desired address register
0572    C
0573
0574            Call LINCHK (lun,1)
0575            Write (lun,120) mlda
0576    120     Format (' ',T8,'MLDA',T24,Z8.8)
0577
0578            if (.not. diagnostic_mode) then
0579
0580            call mba_status_register16_31 (lun,mlas,mlda,1)
0581
0582            Field = LIB$EXTZV (0,16,mlda)
0583
0584            Call LINCHK (lun,1)
0585            Write (lun,130) field
0586    130     Format (' ',T40,'SECTOR = ',I<COMPRESS4 (field)>,'.')
0587            endif
0588
0589
0590    C
0591    C       Decode and output the bits in the drive type register
0592    C
0593
0594            Call LINCHK (lun,1)
0595
0596            Write (lun,140) mldt
0597    140     Format (' ',T8,'MLDT',T24,Z8.8)
0598
0599            if (.not. diagnostic_mode) then
0600
0601            call mba_status_register16_31 (lun,mlda,mldt,1)
0602
0603            Field = LIB$EXTZV (0,9,mldt)
0604
0605            If (field .eq. 110) then
0606
0607            Call LINCHK (lun,1)
0608            Write (lun,150)
0609    150     Format (' ',T40,'DRIVE TYPE = ML11')
0610            Endif
0611            endif
0612
0613    C
0614    C       Decode and output bits in the serial number register
```

```
0615    C
0616
0617            Call LINCHK (lun,1)
0618            Write (lun,160) mlsn
0619    160     Format (' ',T8,'MLSN',T24,Z8.8)
0620
0621            if (.not. diagnostic_mode) then
0622
0623            call mba_status_register16_31 (lun,mldt,mlsn,1)
0624            endif
0625
0626    C
0627    C       Decode and output the bits in the ECC CRC word register 1
0628    C
0629
0630            Call LINCHK (lun,1)
0631            Write (lun,170) mle1
0632    170     Format (' ',T8,'MLE1',T24,Z8.8)
0633
0634            if (.not. diagnostic_mode) then
0635
0636            call mba_status_register16_31 (lun,mlsn,mle1,1)
0637            endif
0638
0639    C
0640    C       Decode and output the bits in the ECC CRC word register 2
0641    C
0642
0643            Call LINCHK (lun,1)
0644            Write (lun,180) mle2
0645    180     Format (' ',T8,'MLE2',T24,Z8.8)
0646
0647            if (.not. diagnostic_mode) then
0648
0649            call mba_status_register16_31 (lun,mle1,mle2,1)
0650            endif
0651
0652    C
0653    C       Decode and output bits in the ECC error register
0654    C
0655
0656            Call LINCHK (lun,1)
0657            Write (lun,190) mlee
0658    190     Format (' ',T8,'MLEE',T24,Z8.8)
0659
0660            if (.not. diagnostic_mode) then
0661
0662            call mba_status_register16_31 (lun,mle2,mlee,1)
0663
0664            Crc_err = LIB$EXTZV (13,1,mlee)
0665            Sngl_err = LIB$EXTZV (14,1,mlee)
0666            Unc_err = LIB$EXTZV (15,1,mlee)
0667            Chan = LIB$EXTZV (6,6,mlee)
0668            Err_func = LIB$EXTZV (0,6,mlee)
0669
0670            Do 194, I=0,5
0671            If (JIAND(err_func,2**I) .ne. 0) then
```

```
0672            Wrd = I+1
0673            Endif
0674
0675    194     Continue
0676
0677
0678            If (crc_err .eq. 1) then
0679
0680            Call LINCHK (lun,2)
0681            Write (lun,195)
0682    195     Format (' ',T40,'CRC ERROR')
0683
0684            Write (lun,196) wrd,chan
0685    196     Format (' ',T40,'WORD ',I1,', BIT ',
0686           1 I<COMPRESS$4 (chan)>, ' IN ERROR')
0687
0688            Else if (sngl_err .eq. 1) then
0689
0690            Call LINCHK (lun,2)
0691            Write (lun,197)
0692    197     Format (' ',T40,'SINGLE ERROR')
0693
0694            Write (lun,198) wrd,chan
0695    198     Format (' ',T40,'WORD ',I1,', BIT ',
0696           1 I<COMPRESS$4 (chan)>,' IN ERROR')
0697
0698            Else if (unc_err .eq. 1) then
0699
0700            Call LINCHK (lun,1)
0701            Write (lun,199)
0702    199     Format (' ',T40,'UNCORRECTABLE ERROR')
0703
0704            Endif
0705            endif
0706
0707    C
0708    C       Decode and output the bits in the ECC error location register
0709    C
0710
0711            Call LINCHK (lun,1)
0712
0713            Write (lun,200) mlel
0714    200     Format (' ',T8,'MLEL',T24,Z8.8)
0715
0716            if (.not. diagnostic_mode) then
0717
0718            call mba_status_register16_31 (lun,mlee,mlel,1)
0719            endif
0720
0721
0722    C
0723    C       Get software information and return to ERRPRT
0724    C
0725
0726            call linchk (lun,1)
0727
0728            write(lun,205)
```

```
0729     205      format(' ',:)
0730
0731              if (
0732              1 emb$w_hd_entry .ne. 98
0733              1 .and.
0734              1 irp_flag .ne. 0
0735              1 ) then
0736
0737              call ucb$b_ertcnt (lun,emb$b_dv_ertcnt)
0738
0739              call ucb$b_ertmax (lun,emb$b_dv_ertmax)
0740              endif
0741
0742              call orb$l_owner (lun,emb$l_dv_ownuic)
0743
0744              call ucb$l_char (lun,emb$l_dv_char)
0745
0746              call ucb$w_sts (lun,emb$w_dv_sts)
0747
0748              call ucb$l_opcnt (lun,emb$l_dv_opcnt)
0749
0750              call ucb$w_errcnt (lun,emb$w_dv_errcnt)
0751
0752              if (
0753              1 emb$w_hd_entry .ne. 98
0754              1 .and.
0755              1 irp_flag .ne. 0
0756              1 ) then
0757
0758              call linchk (lun,1)
0759
0760              write(lun,205)
0761
0762              call ml11_qio (lun,emb$w_dv_func)
0763
0764              call irp$w_bcnt (lun,emb$w_dv_bcnt)
0765
0766              call irp$w_boff (lun,emb$w_dv_boff)
0767
0768              call irp$l_pid (lun,emb$l_dv_rqpid)
0769
0770              call irp$q_iosb (lun,emb$l_dv_iosb1)
0771              endif
0772
0773              Return
0774              End
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	2709	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	739	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	2116	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	6076	

ENTRY POINTS

Address	Type	Name
0-00000000		ML11

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000244	I*4	CARDS	2-00000248	I*4	CHAN
2-00000250	I*4	CRC_ERR	2-0000022B	L*1	DIAGNOSTIC_MODE
2-0000023C	I*4	DRV_FUNC	3-0000001C	L*1	EMB$B_DV_CLASS
3-00000010	L*1	EMB$B_DV_ERTCNT	3-00000011	L*1	EMB$B_DV_ERTMAX
3-0000003E	L*1	EMB$B_DV_NAMLNG	3-0000003A	L*1	EMB$B_DV_SLAVE
3-0000001D	L*1	EMB$B_DV_TYPE	3-00000036	I*4	EMB$L_DV_CHAR
3-00000012	I*4	EMB$L_DV_IOSB1	3-00000016	I*4	EMB$L_DV_IOSB2
3-00000026	I*4	EMB$L_DV_MEDIA	3-0000004E	I*4	EMB$L_DV_NUMREG
3-0000002E	I*4	EMB$L_DV_OPCNT	3-00000032	I*4	EMB$L_DV_OWNUIC
3-0000001E	I*4	EMB$L_DV_RQPID	3-00000000	I*4	EMB$L_HD_SID
3-0000003F	CHAR	EMB$T_DV_NAME	3-00000024	I*2	EMB$W_DV_BCNT
3-00000022	I*2	EMB$W_DV_BOFF	3-0000002C	I*2	EMB$W_DV_ERRCNT
3-0000003C	I*2	EMB$W_DV_FUNC	3-0000001A	I*2	EMB$W_DV_STS
3-0000002A	I*2	EMB$W_DV_UNIT	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	2-0000024C	I*4	ERR_FUNC
2-0000022C	I*4	FIELD	2-00000260	I*4	I
3-0000006E	I*4	IRP_FLAG	AP-00000004@	L*1	LUN
3-00000082	I*4	MLAS	3-00000072	I*4	MLCS1
3-00000086	I*4	MLDA	3-00000076	I*4	MLDS
3-0000008A	I*4	MLDT	3-00000096	I*4	MLE1
3-0000009A	I*4	MLE2	3-000000A6	I*4	MLEE
3-000000AA	I*4	MLEL	3-0000007A	I*4	MLER
3-0000007E	I*4	MLMR	3-00000092	I*4	MLSN
2-00000234	I*4	PROM_DIS	2-00000238	I*4	PROM_RW
2-00000230	I*4	SELECTED_MAP_REGISTER	2-00000254	I*4	SNGL_ERR
2-00000240	I*4	TYPE	2-00000258	I*4	UNC_ERR
2-0000025C	I*4	WRD			

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000052	I*4	ADAPTER_REGISTERS	28	(7)
2-0000020B	CHAR	ARAY_TYP	32	(0:1)
3-00000000	L*1	EMB	512	(0:511)
3-00000052	I*4	EMB$L_DV_REGSAV	420	(0:104)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)
2-00000000	CHAR	MLCS1_1	7	(0:0)
2-00000007	CHAR	MLCS1_2	16	(11:11)
2-00000017	CHAR	MLDS_1	42	(6:8)
2-00000041	CHAR	MLDS_2	23	(10:10)
2-00000058	CHAR	MLDS_3	34	(14:15)
2-0000007A	CHAR	MLDS_MOL	32	(0:1)
2-0000009A	CHAR	MLER_1	96	(0:3)
2-000000FA	CHAR	MLER_2	30	(5:6)
2-00000118	CHAR	MLER_3	46	(9:10)
2-00000146	CHAR	MLER_4	63	(13:15)
2-00000185	CHAR	MLMR_1	36	(2)
2-000001A9	CHAR	MLMR_2	30	(7:7)
2-000001C7	CHAR	XFER_RATE	68	(0:3)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-0000004A	20'	1-0000005D	22'	1-00000071	24'	1-00000084	26'	1-0000009A	28'	1-000000A8	30'
1-000000BB	32'	1-000000CD	34'	1-000000DE	40'	1-000000EF	45'	1-000000FB	50'	1-0000010C	60'
1-00000160	70'	1-0000011D	75'	1-00000137	80'	1-00000143	85'	1-00000177	100'	1-00000188	105'
**	106	1-000001A4	120'	1-000001B5	130'	1-000001CF	140'	1-000001E0	150'	1-000001F9	160'
1-0000020A	170'	1-0000021B	180'	1-0000022C	190'	**	194	1-0000023D	195'	1-0000024E	196'
1-00000276	197'	1-0000028A	198'	1-000002B2	199'	1-000002CD	200'	1-000002DE	205'		

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	COMPRESSC		DHEAD1
	FRCTOF		IRP$L_PID		IRP$Q_IOSB
	IRP$W_BCNT		IRP$W_BOFF	I*4	LIB$EXTZV
	LINCHR		MBA_CONTROL_REGISTERS		MBA_MAPPING_REGISTER
	MBA_STATUS_REGISTER16_31		ML1T_QIO		ORB$L_OWNER
	OUTPUT		UCB$B_ERTCNT		UCB$B_ERTMAX
	UCB$L_CHAR		UCB$L_OPCNT		UCB$W_ERRCNT
	UCB$W_STS				

MOUN

0400
0401
0402
0403
0404
0405
0406
0407
0408
0409
0410
0411
0412
0413
0414
0415
0416
0417
0418
0419
0420
0421
0422
0423
0424
0425
0426
0427
0428
0429
0430
0431
0432
0433
0434
0435
0436
0437
0438
0439
0440
0441
0442
0443
0444
0445
0446
0447
0448
0449
0450
0451
0452
0453
0454
0455
0456

```
0001
0002
0003
0004          Subroutine ML11_QIO (lun,emb$w_dv_func)
0005          include 'src$:qiocommon.for /nolist'
0270
0271
0272          byte            lun
0273
0274          integer*2       emb$w_dv_func
0275
0276          integer*4       qiocode(0:1,0:63)
0277
0278
0279          if (qiocode(0,0) .eq. 0) then
0280
0281          qiocode(1,00) = %loc(io$_nop)
0282
0283          qiocode(1,01) = %loc(io$_unload)
0284
0285          qiocode(1,02) = %loc(io$_seek)
0286
0287          qiocode(1,03) = %loc(io$_recal)
0288
0289          qiocode(1,04) = %loc(io$_drvclr)
0290
0291          qiocode(1,05) = %loc(io$_release)
0292
0293          qiocode(1,06) = %loc(io$_offset)
0294
0295          qiocode(1,07) = %loc(io$_retcenter)
0296
0297          qiocode(1,08) = %loc(io$_packack)
0298
0299          qiocode(1,09) = %loc(io$_search)
0300
0301          qiocode(1,10) = %loc(io$_writecheck)
0302
0303          qiocode(1,11) = %loc(io$_writepblk)
0304
0305          qiocode(1,12) = %loc(io$_readpblk)
0306
0307          qiocode(1,13) = %loc(io$_writehead)
0308
0309          qiocode(1,14) = %loc(io$_readhead)
0310
0311          qiocode(1,24) = %loc(io$_writecheckh)
0312
0313          qiocode(1,25) = %loc(io$_readpreset)
0314
0315          qiocode(1,26) = %loc(io$_setchar)
0316
0317          qiocode(1,27) = %loc(io$_sensechar)
0318
0319          qiocode(1,32) = %loc(io$_writelblk)
0320
```

MOUN

0457 0458 0459 0460 0461 0462 0463 0464 0465 0466 0467 0468 0469 0470 0471 0472 0473 0474 0475 0476 0477 0478 0479 0480 0481 0482 0483 0484 0485 0486 0487 0488 0489 0490 0491 0492 0493 0494 0495 0496 0497 0498 0499 0500 0501 0502 0503 0504 0505 0506 0507 0508 0509 0510 0511 0512 0513

```
0321            qiocode(1,33) = %loc(io$_readlblk)
0322
0323            qiocode(1,35) = %loc(io$_setmode)
0324
0325            qiocode(1,39) = %loc(io$_sensemode)
0326
0327            qiocode(1,48) = %loc(io$_writevblk)
0328
0329            qiocode(1,49) = %loc(io$_readvblk)
0330
0331            qiocode(1,50) = %loc(io$_access)
0332
0333            qiocode(1,51) = %loc(io$_create)
0334
0335            qiocode(1,52) = %loc(io$_deaccess)
0336
0337            qiocode(1,53) = %loc(io$_delete)
0338
0339            qiocode(1,54) = %loc(io$_modify)
0340
0341            qiocode(1,56) = %loc(io$_acpcontrol)
0342
0343            qiocode(1,57) = %loc(io$_mount)
0344
0345            do 10,i = 0,63
0346
0347            qiocode(0,i) = 33
0348
0349            if (qiocode(1,i) .eq. 0) then
0350
0351            qiocode(1,i) = %loc(qio_string)
0352            endif
0353
0354      10    continue
0355            endif
0356
0357            call irp$w_func (lun,emb$w_dv_func,
0358            1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))
0359
0360            return
0361
0362            end
```

```
MOUN

0514
0515
0516
0517
0518
0519
0520
0521
0522
0523
0524
0525
0526
0527
0528
0529

PROG

   0
   1
   2

ENTR

   0-
   0-

VARI
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	309	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	8	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	548	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	QIOCOMMON	1247	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	2112	

ENTRY POINTS

Address	Type	Name
0-00000000		ML11_QIO

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008@	I*2	EMB$W_DV_FUNC	2-00000200	I*4	I
3-00000442	CHAR	IO$_ABORT	3-0000034D	CHAR	IO$_ACCESS
3-000003C2	CHAR	IO$_ACPCONTROL	3-000004B3	CHAR	IO$_AVAILABLE
3-00000297	CHAR	IO$_CLEAN	3-00000369	CHAR	IO$_CREATE
3-00000385	CHAR	IO$_DEACCESS	3-00000393	CHAR	IO$_DELETE
3-0000026D	CHAR	IO$_DIAGNOSE	3-00000065	CHAR	IO$_DRVCLR
3-000004CB	CHAR	IO$_DSE	3-000000A9	CHAR	IO$_ERASETAPE
3-00000276	CHAR	IO$_FORMAT	3-00000071	CHAR	IO$_INITIALIZE
3-00000014	CHAR	IO$_LOADMCODE	3-000003A1	CHAR	IO$_MODIFY
3-000003E2	CHAR	IO$_MOUNT	3-00000000	CHAR	IO$_NOP
3-0000009D	CHAR	IO$_OFFSET	3-000000EB	CHAR	IO$_PACKACK
3-000000E0	CHAR	IO$_QSTOP	3-000003EF	CHAR	IO$_RDSTATS
3-00000421	CHAR	IO$_READCSR	3-00000169	CHAR	IO$_READHEAD
3-000002B6	CHAR	IO$_READLBLK	3-0000013F	CHAR	IO$_READPBLK
3-00000200	CHAR	IO$_READPRESET	3-00000195	CHAR	IO$_READTRACKD
3-0000033A	CHAR	IO$_READVBLK	3-0000045A	CHAR	IO$_READWTHBUF
3-00000484	CHAR	IO$_READWTHXBUF	3-0000004D	CHAR	IO$_RECAL
3-0000007C	CHAR	IO$_RELEASE	3-000001AB	CHAR	IO$_REREADN
3-000001B8	CHAR	IO$_REREADP	3-000000CA	CHAR	IO$_RETCENTER
3-000002E6	CHAR	IO$_REWIND	3-000002C9	CHAR	IO$_REWINDOFF
3-000000FC	CHAR	IO$_SEARCH	3-00000024	CHAR	IO$_SEEK
3-00000231	CHAR	IO$_SENSECHAR	3-00000309	CHAR	IO$_SENSEMODE
3-0000021D	CHAR	IO$_SETCHAR	3-000003B8	CHAR	IO$_SETCLOCK
3-00000088	CHAR	IO$_SETCLOCKP	3-000002DD	CHAR	IO$_SETMODE
3-000002ED	CHAR	IO$_SKIPFILE	3-000002FA	CHAR	IO$_SKIPRECORD
3-00000029	CHAR	IO$_SPACEFILE	3-0000010E	CHAR	IO$_SPACERECORD
3-000003D7	CHAR	IO$_STARTDATA	3-000000B4	CHAR	IO$_STARTDATAP
3-00000037	CHAR	IO$_STARTMPROC	3-0000020F	CHAR	IO$_STARTSPNDL
3-00000059	CHAR	IO$_STOP	3-0000000D	CHAR	IO$_UNLOAD
3-0000046B	CHAR	IO$_WRITEBUFNCRC	3-0000011E	CHAR	IO$_WRITECHECK
3-000001E4	CHAR	IO$_WRITECHECKH	3-000003FF	CHAR	IO$_WRITECSR
3-00000153	CHAR	IO$_WRITEHEAD	3-000002A2	CHAR	IO$_WRITELBLK
3-00000247	CHAR	IO$_WRITEMARK	3-00000314	CHAR	IO$_WRITEOF
3-0000012A	CHAR	IO$_WRITEPBLK	3-000001C9	CHAR	IO$_WRITERET

```
  3-0000017E  CHAR IO$_WRITETRACKD          3-00000326  CHAR IO$_WRITEVBLK
  3-00000448  CHAR IO$_WRITEWTHBUF          3-00000257  CHAR IO$_WRTTMKR
 AP-0000004a  L*1  LUN                      3-000004A1  CHAR QIO_STRING

ARRAYS

    Address  Type  Name                Bytes  Dimensions

  2-00000000  I*4  QIOCODE               512  (0:1, 0:63)

LABELS

    Address  Label

      **     10

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name             Type  Name

        IRP$W_FUNC        I*4  LIB$EXTZV

COMMAND QUALIFIERS

  FORTRAN /LIS=LIS$:ML11/OBJ=OBJ$:ML11 MSRC$:ML11
  /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:              10.27 seconds
  Elapsed Time:          23.17 seconds
  Page Faults:           269
  Dynamic Memory:        240 pages
```

0001
0002
0003
0004
0005
0006
0007
0008
0067
0068
0132
0133
0134
0135
0136
0137
0138
0139
0140
0141
0142
0143
0144
0145
0146
0147
0148
0149
0150
0151
0152
0153
0154
0155
0156
0157
0158
0159
0160
0161
0162
0163
0164
0165
0166
0167
0168
0169
0170
0171
0172
0173
0174
0175
0176
0177
0178

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