```
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
```

```
_$

Va
--
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
00
```

FILEID**DQDISKS

```
DDDDDDDD      QQQQQQ    DDDDDDDD      IIIIII      SSSSSSSS  KK      KK    SSSSSSSS
DDDDDDDD      QQQQQQ    DDDDDDDD      IIIIII      SSSSSSSS  KK      KK    SSSSSSSS
DD      DD  QQ      QQ  DD      DD      II      SS          KK      KK  SS
DD      DD  QQ      QQ  DD      DD      II      SS          KK      KK  SS
DD      DD  QQ      QQ  DD      DD      II      SS          KK    KK    SS
DD      DD  QQ      QQ  DD      DD      II      SS          KK    KK    SS
DD      DD  QQ      QQ  DD      DD      II        SSSSSS    KKKKKK        SSSSSS
DD      DD  QQ      QQ  DD      DD      II        SSSSSS    KKKKKK        SSSSSS
DD      DD  QQ  QQ  QQ  DD      DD      II              SS  KK    KK            SS
DD      DD  QQ  QQ  QQ  DD      DD      II              SS  KK    KK            SS
DD      DD  QQ    QQ    DD      DD      II              SS  KK      KK          SS
DD      DD  QQ    QQ    DD      DD      II              SS  KK      KK          SS    ....
DDDDDDDD      QQQQ  QQ  DDDDDDDD      IIIIII    SSSSSSSS    KK      KK  SSSSSSSS      ....
DDDDDDDD      QQQQ  QQ  DDDDDDDD      IIIIII    SSSSSSSS    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
```

```
DQD
ARR
3
3
3
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
2
LAB
1
1
FUN
T
```

```
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   20-JUL-1981

c++
c       Functional description:
c
c       This module is used to display error log entries logged for the
c       11/7zz IDC.  The format of the device specific portion of the record
c       is as follows.
c
c       +----------------------------------+
c       :               csr                :
c       +----------------------------------+
c       :               bar                :
c       +----------------------------------+
c       :               bcr                :
c       +----------------------------------+
c       :               dar                :
c       +----------------------------------+
c       :               mpr                :
c       +----------------------------------+
c       :               ecc1               :
c       +----------------------------------+
c       :               ecc2               :
c       +----------------------------------+
c       :        data path number          :
c       +----------------------------------+
c       .     data path reg (always 0)     :
c       +----------------------------------+
c       :           final uba map          :
c       +----------------------------------+
c       |        previous uba map           |
c       +-----------------------------------+
c       |     vec$l_mapreg (from crb)       |
c       +-----------------------------------+
c
c       Modified by:
c
C       V03-003 SAR0217         Sharon A. Reynolds,     28-Mar-1984
C               Changed the call to UCB$L_OWNUIC to ORB$L_OWNER.
C
C       V03-002 SAR0069         Sharon A. Reynolds,     20-Jun-1983
C               Changed the carriage control in the 'format' statements
C               for use with ERF.
C
C       v03-001 SAR0046         Sharon A. Reynolds,     9-Jun-1983
C               Removed brief/cryptic support.
C
c       v02-008 BP0008          Brian Porter,           23-JAN-1982
c               Corrected polarity of 'plug valid' for the r80.
c
c       v02-007 BP0007          Brian Porter,           23-NOV-1981
c               Minor edit.
c
c       v02-006 BP0006          Brian Porter,           04-NOV-1981
c               Corrected 'DAR' output error.  Added 'device attention'
c               support.
c
c       v02-005 BP0005          Brian Porter,           30-SEP-1981
c               Corrected random problems.
c
c       v02-004 BP0004          Brian Porter,           29-SEP-1981
c               Added 'DAR' decoding functionality.
c
c       v02-003 BP0003          Brian Porter,           14-SEP-1981
c               Corrected problem in attention logic.  Added CSR
c               functionality.
c
c       v02-002 BP0002          Brian Porter,           31-AUG-1981
c               Corrected call to calc_map.
c
c       v02-001 BP0001          Brian Porter,           24-AUG-1981
c               Changed record format to conform to other drivers.
c**
c--



        Subroutine DQDISKS (lun)


        include 'src$:msghdr.for /nolist'
        include 'src$:deverr.for /nolist'

        byte            lun

        integer*4       control_status_register
        integer*4       bus_address_register
        integer*4       byte_control_register
        integer*4       disk_address_register
        integer*4       multi_purpose_register
        integer*4       ecc_position_register
        integer*4       ecc_pattern_register
        integer*4       data_path_number
        integer*4       data_path_register
        integer*4       final_map_register
        integer*4       previous_map_register
        integer*4       vec$l_mapreg

        equivalence     (emb$l_dv_regsav(0),control_status_register)
        equivalence     (emb$l_dv_regsav(1),bus_address_register)
        equivalence     (emb$l_dv_regsav(2),byte_control_register)
        equivalence     (emb$l_dv_regsav(3),disk_address_register)
        equivalence     (emb$l_dv_regsav(4),multi_purpose_register)
        equivalence     (emb$l_dv_regsav(5),ecc_position_register)
        equivalence     (emb$l_dv_regsav(6),ecc_pattern_register)
        equivalence     (emb$l_dv_regsav(7),data_path_number)
        equivalence     (emb$l_dv_regsav(8),data_path_register)
        equivalence     (emb$l_dv_regsav(9),final_map_register)
        equivalence     (emb$l_dv_regsav(10),previous_map_register)
        equivalence     (emb$l_dv_regsav(11),vec$l_mapreg)

        character*12    v1csr(0:0)
        data            v1csr(0)        /'DRIVE READY*'/

        character*17    v2csr(6:7)
        data            v2csr(6)        /'INTERRUPT ENABLE*'/
        data            v2csr(7)        /'CONTROLLER READY*'/

        character*21    v3csr(10:10)
        data            v3csr(10)       /'OPERATION INCOMPLETE*'/

        character*20    v4csr(13:15)
        data            v4csr(13)       /'NON-EXISTENT MEMORY*'/
        data            v4csr(14)       /'DRIVE ERROR*'/
        data            v4csr(15)       /'COMPOSITE ERROR*'/

        character*22    v5csr(22:24)
        data            v5csr(22)       /'R80 SKIP SECTOR ERROR*'/
        data            v5csr(23)       /'R80 SKIP SECTOR ERROR*'/
        data            v5csr(24)       /'INTERRUPT REQUEST*'/

        character*30    v6csr(26:28)
        data            v6csr(26)       /'R80*'/
        data            v6csr(27)       /'AUTOMATIC SKIP SECTOR INHIBIT*'/
        data            v6csr(28)       /'TIMEOUT INHIBIT*'/

        character*11    v1rl02_mpr(3:5)
        data            v1rl02_mpr(3)   /'BRUSH HOME*'/
        data            v1rl02_mpr(4)   /'HEADS OUT*'/
        data            v1rl02_mpr(5)   /'COVER OPEN*'/

        character*19    v2rl02_mpr(8:15)
        data            v2rl02_mpr(8)   /'DRIVE SELECT ERROR*'/
        data            v2rl02_mpr(9)   /'VOLUME CHECK*'/
        data            v2rl02_mpr(10)  /'WRITE GATE ERROR*'/
        data            v2rl02_mpr(11)  /'SPINDLE ERROR*'/
        data            v2rl02_mpr(12)  /'SEEK TIMEOUT*'/
        data            v2rl02_mpr(13)  /'WRITE LOCK*'/
        data            v2rl02_mpr(14)  /'HEAD CURRENT ERROR*'/
        data            v2rl02_mpr(15)  /'WRITE DATE ERROR*'/

        character*14    v1r80_mpr(8:13)
        data            v1r80_mpr(8)    /'FAULT*'/
        data            v1r80_mpr(9)    /'PLUG VALID*'/
        data            v1r80_mpr(10)   /'SEEK ERROR*'/
        data            v1r80_mpr(11)   /'ON CYLINDER*'/
        data            v1r80_mpr(12)   /'DRIVE READY*'/
        data            v1r80_mpr(13)   /'WRITE PROTECT*'/

        integer*4       compress4
        integer*4       compressc
        integer*4       field

        character*27    idc_command(0:7)
        data            idc_command(0)  /'NO DRIVE OPERATION*'/
        data            idc_command(1)  /'WRITE CHECK DATA*'/
        data            idc_command(2)  /'GET STATUS*'/
        data            idc_command(3)  /'SEEK*'/
        data            idc_command(4)  /'READ HEADER*'/
        data            idc_command(5)  /'WRITE DATA*'/
        data            idc_command(6)  /'READ DATA*'/
        data            idc_command(7)  /'READ DATA W/O HEADER CHECK*'/

        logical*1       diagnostic_mode

        integer*4       lib$extzv
        integer*4       data_check_and_opi_bits
        integer*4       data_late_and_opi_bits
        integer*4       sector_count
        integer*4       ecc_status_bits
        integer*4       rl02_status_bits

        character*20    v1rl02_status_bits(0:7)
        data    v1rl02_status_bits(0)   /'LOAD STATE*'/
        data    v1rl02_status_bits(1)   /'SPIN UP*'/
        data    v1rl02_status_bits(2)   /'BRUSH CYCLE*'/
        data    v1rl02_status_bits(3)   /'LOAD HEADS*'/
        data    v1rl02_status_bits(4)   /'SEEK TRACK COUNTING*'/
        data    v1rl02_status_bits(5)   /'SEEK LINEAR MODE*'/
        data    v1rl02_status_bits(6)   /'UNLOAD HEADS*'/
        data    v1rl02_status_bits(7)   /'SPIN DOWN*'/

        integer*4       device_function
        integer*4       device_type
        integer*4       sector
        integer*4       cylinder
        integer*4       tag
        integer*4       head

        character*11    v1dar(0:1)
        data            v1dar(0)        /'MARKER*'/
        data            v1dar(1)        /'GET STATUS*'/

        character*6     v2dar(3:3)
        data            v2dar(3)        /'RESET*'/

        character*8     v4dar(2:2,0:1)
        data            v4dar(2,0)      /'REVERSE*'/
        data            v4dar(2,1)      /'FORWARD*'/

        character*18    v6dar(4:4,0:1)
        data            v6dar(4,0)      /'SELECT LOWER HEAD*'/
        data            v6dar(4,1)      /'SELECT UPPER HEAD*'/

        character*15    v7dar(6:6)
        data            v7dar(6)        /'RETURN-TO-ZERO*'/


        call frctof (lun)

        call dhead1 (lun,'RB730')

        diagnostic_mode = .false.

        if (lib$extzv(25,1,control_status_register) .eq. 1)
        1 diagnostic_mode = .true.

        device_function = lib$extzv (1,3,control_status_register)

        device_type = lib$extzv (26,1,control_status_register)

        call linchk (lun,2)

        write(lun,5) 'RB CSR',control_status_register
5       format(/' ',t8,a,t24,z8.8)

        if (.not. diagnostic_mode) then

        call output (lun,control_status_register,v1csr,0,0,0,'0')

        call linchk (lun,1)

        if (lib$extzv(29,1,control_status_register) .eq. 1) then

        write(lun,10) 'R80 WRITE FORMAT FUNCTION'
10      format(' ',t40,a)
        else

        idc_function = lib$extzv(1,3,control_status_register)

        write(lun,15) idc_command(idc_function)
15      format(' ',t40,a<compressc (idc_command(idc_function))>)
        endif

        call output (lun,control_status_register,v2csr,6,6,7,'0')

        call linchk (lun,1)

        write(lun,20) 'DRIVE #',lib$extzv(8,2,control_status_register),
        1 '. SELECTED'
20      format(' ',t40,a,i1.1,a)

        call output (lun,control_status_register,v3csr,10,10,10,'0')

        data_check_and_opi_bits = lib$extzv(10,2,control_status_register)

        if (
        1 data_check_and_opi_bits .eq. 2
        1 .or.
        1 data_check_and_opi_bits .eq. 3
        1 ) then

        call linchk (lun,1)
        endif

        if (data_check_and_opi_bits .eq. 2) then

        write(lun,25) 'DATA CHECK ERROR'
25      format(' ',t40,a)

        else if (data_check_and_opi_bits .eq. 3) then

        write(lun,25) 'HEADER CRC ERROR'
        endif

        data_late_and_opi_bits = lib$extzv(10,3,control_status_register)

        if (
        1 data_late_and_opi_bits .eq. 4
        1 .or.
        1 data_late_and_opi_bits .eq. 5
        1 ) then

        call linchk (lun,1)
        endif

        if (data_late_and_opi_bits .eq. 4) then

        write(lun,25) 'DATA LATE'

        else if (data_late_and_opi_bits .eq. 5) then

        write(lun,25) 'HEADER NOT FOUND'
        endif

        call output (lun,control_status_register,v4csr,13,13,15,'0')

        do 35,i = 16,19

        if (lib$extzv(i,1,control_status_register) .eq. 1) then

        call linchk (lun,1)

        write(lun,30) 'ATTENTION DRIVE #',i-16,'.'
30      format(' ',t40,a,i1.1,a)
```

```
        endif

35      continue

        if (lib$extzv (26,1,control_status_register) .eq. 1) then

        ecc_status_bits = lib$extzv (20,2,control_status_register)

        if (ecc_status_bits .ne. 0) then

        call linchk (lun,1)

        if (ecc_status_bits .eq. 1) then

        write(lun,40) 'DATA ERROR'
40      format(' ',t40,a)

        else if (ecc_status_bits .eq. 2) then

        write(lun,40) 'HARD ERROR'

        else if (ecc_status_bits .eq. 3) then

        write(lun,40) 'CORRECTABLE ERROR'
        endif
        endif
        endif

        call output (lun,control_status_register,v5csr,22,22,24,'0')

        if (lib$extzv (26,1,control_status_register) .eq. 1) then

        call output (lun,control_status_register,v6csr,26,26,28,'0')
        endif
        else

        call linchk (lun,1)

        write(lun,40) 'DIAGNOSTIC MODE'
        endif

        call linchk (lun,1)

        write(lun,45) 'RB BAR',bus_address_register
45      format(' ',t8,a,t24,z8.8)

        if (.not. diagnostic_mode) then

        if (
        1 device_function .eq. 1
        1 .or.
        1 device_function .eq. 5
        1 .or.
        1 device_function .eq. 6
        1 .or.
        1 device_function .eq. 7
        1 ) then
```

```

            call calc_map (lun,16,bus_address_register,bus_address_register)
            endif
            endif

            call linchk (lun,1)

            write(lun,45) 'RB BCR',byte_control_register

            call linchk (lun,1)

            write(lun,45) 'RB DAR',disk_address_register

            if (.not. diagnostic_mode) then

            if (
            1 device_function .eq. 1
            1 .or.
            1 device_function .eq. 5
            1 .or.
            1 device_function .eq. 6
            1 .or.
            1 device_function .eq. 7
            1 ) then

            if (device_type .eq. 0) then

            sector = lib$extzv (0,6,disk_address_register)

            cylinder = lib$extzv (7,9,disk_address_register)

            else if (device_type .eq. 1) then

            sector = lib$extzv (0,5,disk_address_register)

            cylinder = lib$extzv (9,10,disk_address_register)
            endif

            call linchk (lun,2)

            write(lun,46) sector,cylinder
      46    format(' ',t40,'SECTOR #',i<compress4 (sector)>,'.',/,
            1 t40,'CYLINDER #',i<compress4 (cylinder)>,'.')

            else if (device_function .eq. 2) then

            if (device_type .eq. 0) then

            call output (lun,disk_address_register,v1dar,0,0,1,'0')

            call output (lun,disk_address_register,v2dar,3,3,3,'0')
            endif

            else if (device_function .eq. 3) then

            if (device_type .eq. 0) then

            call output (lun,disk_address_register,v1dar,0,0,1,'0')

            call output (lun,disk_address_register,v4dar,2,2,2,'2')

            call output (lun,disk_address_register,v2dar,3,3,3,'0')

            call output (lun,disk_address_register,v6dar,4,4,4,'2')

            cylinder = lib$extzv (7,9,disk_address_register)

            call linchk (lun,1)

            write(lun,47) cylinder
    47      format(' ',t40,i<compress4 (cylinder)>,'. CYLINDER(S) TO MOVE')

            else if (device_type .eq. 1) then

            tag = lib$extzv (13,3,disk_address_register)

            call linchk (lun,1)

            if (tag .eq. 1) then

            cylinder = lib$extzv (0,10,disk_address_register)

            write(lun,48) 'CYLINDER #',cylinder
    48      format(' ',t40,a,i<compress4 (cylinder)>,'. SELECTED')

            else if (tag .eq. 2) then

            head = lib$extzv (0,4,disk_address_register)

            write(lun,48) 'HEAD #',head

            else if (tag .eq. 4) then

            call output (lun,disk_address_register,v7dar,6,6,6,'0')
            endif
            endif
            endif
            endif

            call linchk (lun,1)

            write(lun,50) 'RB MPR',multi_purpose_register
    50      format(' ',t8,a,t24,z8.8)

            if (.not. diagnostic_mode) then

            if (lib$extzv (26,1,control_status_register) .eq. 1) then

            sector_count = lib$extzv (0,5,multi_purpose_register)

            call linchk (lun,1)

            write(lun,55) 'SECTOR COUNT ',sector_count,'.'
    55      format(' ',t40,a,i<compress4 (sector_count)>,a)
```

```

            call output (lun,multi_purpose_register,v1r80_mpr,8,8,13,'0')
            else

            rl02_status_bits = lib$extzv (0,3,multi_purpose_register)

            call linchk (lun,1)

            write(lun,60) v1rl02_status_bits(rl02_status_bits)
    60      format(' ',t40,a<compressc (v1rl02_status_bits(rl02_status_bits))>)

            call output (lun,multi_purpose_register,v1rl02_mpr,3,3,5,'0')

            call linchk (lun,1)

            if (lib$extzv (6,1,multi_purpose_register) .eq. 1) then

            write(lun,65) 'LOWER HEAD SELECTED'
    65      format(' ',t40,a)
            else

            write(lun,65) 'UPPER HEAD SELECTED'
            endif

            call output (lun,multi_purpose_register,v2rl02_mpr,8,8,15,'0')
            endif
            endif

            call linchk (lun,2)

            write(lun,70) 'RB ECC1',ecc_position_register,
            1 'RB ECC2',ecc_pattern_register
    70      format(' ',t8,a,t24,z8.8,/,t8,a,t24,z8.8)

            if (
            1 (device_function .eq. 1
            1 .or.
            1 device_function .eq. 5
            1 .or.
            1 device_function .eq. 6
            1 .or.
            1 device_function .eq. 7)
            1 .and.
            1 emb$w_hd_entry .ne. 98
            1 ) then

            call uba_datapath (lun,iand(data_path_number,'0000007f'x),
            1 data_path_register)

            call calc_map2 (16,bus_address_register,bus_address_register,field)

            call uba_mapping (lun,field,final_map_register)

            if (
            1 lib$extzv (16,16,emb$l_dv_iosb1) .gt. 512
            1 .and.
            1 field .ne. 0
```

```
        1 ) then

        call uba_mapping (lun,(field-1),previous_map_register)
        endif

        call vecmapreg (lun,vec$l_mapreg)
        endif

        call linchk (lun,1)

        write(lun,75)
75      format(' ',:)

        if (emb$w_hd_entry .ne. 98) then

        call ucb$b_ertcnt (lun,emb$b_dv_ertcnt)

        call ucb$b_ertmax (lun,emb$b_dv_ertmax)
        endif

        call orb$l_owner (lun,emb$l_dv_ownuic)

        call ucb$l_char (lun,emb$l_dv_char)

        call ucb$w_sts (lun,emb$w_dv_sts)

        call ucb$l_opcnt (lun,emb$l_dv_opcnt)

        call ucb$w_errcnt (lun,emb$w_dv_errcnt)

        if (emb$w_hd_entry .ne. 98) then

        call ucb$l_media (lun,emb$l_dv_media)

        call linchk (lun,1)

        write(lun,75)

        call dqdisks_qio (lun,emb$w_dv_func)

        call irp$w_bcnt (lun,emb$w_dv_bcnt)

        call irp$w_boff (lun,emb$w_dv_boff)

        call irp$l_pid (lun,emb$l_dv_rqpid)

        call irp$q_iosb (lun,emb$l_dv_iosb1)
        endif

        return
        end
```

```
DQDISKS                                                   L  7
                                                    16-Sep-1984 00:02:07     VAX-11 FORTRAN V3.4-56                     Page  12
                                                     5-Sep-1984 13:52:37     DISK$VMSMASTER:[ERF.SRC]DQDISKS.FOR;1

PROGRAM SECTIONS

    Name                                    Bytes   Attributes

  0 $CODE                                    2861   PIC CON REL LCL   SHR   EXE    RD NOWRT LONG
  1 $PDATA                                    634   PIC CON REL LCL   SHR NOEXE    RD NOWRT LONG
  2 $LOCAL                                   2972   PIC CON REL LCL NOSHR NOEXE    RD   WRT LONG
  3 EMB                                       512   PIC OVR REL GBL   SHR NOEXE    RD   WRT LONG

    Total Space Allocated                    6979

ENTRY POINTS

    Address   Type  Name

  0-00000000        DQDISKS

VARIABLES

    Address   Type  Name                                     Address   Type  Name

  3-00000056  I*4   BUS_ADDRESS_REGISTER                   3-0000005A  I*4   BYTE_CONTROL_REGISTER
  3-00000052  I*4   CONTROL_STATUS_REGISTER                2-00000424  I*4   CYLINDER
  2-00000404  I*4   DATA_CHECK_AND_OPI_BITS                2-00000408  I*4   DATA_LATE_AND_OPI_BITS
  3-0000006E  I*4   DATA_PATH_NUMBER                       3-00000072  I*4   DATA_PATH_REGISTER
  2-00000418  I*4   DEVICE_FUNCTION                        2-0000041C  I*4   DEVICE_TYPE
  2-000003FF  L*1   DIAGNOSTIC_MODE                        3-0000005E  I*4   DISK_ADDRESS_REGISTER
  3-0000006A  I*4   ECC_PATTERN_REGISTER                   3-00000066  I*4   ECC_POSITION_REGISTER
  2-00000410  I*4   ECC_STATUS_BITS                        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                        2-00000400  I*4   FIELD
  3-00000076  I*4   FINAL_MAP_REGISTER                     2-0000042C  I*4   HEAD
  2-00000434  I*4   I                                      2-00000430  I*4   IDC_FUNCTION
 AP-00000004@ L*1   LUN                                    3-00000062  I*4   MULTI_PURPOSE_REGISTER
  3-0000007A  I*4   PREVIOUS_MAP_REGISTER                  2-00000414  I*4   RL02_STATUS_BITS
  2-00000420  I*4   SECTOR                                 2-0000040C  I*4   SECTOR_COUNT
  2-00000428  I*4   TAG                                    3-0000007E  I*4   VEC$L_MAPREG
```

```
                                                          M  7
DQDISKS                                                 16-Sep-1984 00:02:07    VAX-11 FORTRAN V3.4-56                Page  13
                                                         5-Sep-1984 13:52:37    DISK$VMSMASTER:[ERF.SRC]DQDISKS.FOR;1

ARRAYS

    Address   Type  Name                                        Bytes  Dimensions

  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)
  2-00000228  CHAR  IDC_COMMAND                                   216  (0:7)
  2-00000000  CHAR  V1CSR                                          12  (0:0)
  2-000003A0  CHAR  V1DAR                                          22  (0:1)
  2-000001D4  CHAR  V1R80_MPR                                      84  (8:13)
  2-0000011B  CHAR  V1RL02_MPR                                     33  (3:5)
  2-00000300  CHAR  V1RL02_STATUS_BITS                            160  (0:7)
  2-0000000C  CHAR  V2CSR                                          34  (6:7)
  2-000003B6  CHAR  V2DAR                                           6  (3:3)
  2-0000013C  CHAR  V2RL02_MPR                                    152  (8:15)
  2-0000002E  CHAR  V3CSR                                          21  (10:10)
  2-00000043  CHAR  V4CSR                                          60  (13:15)
  2-000003BC  CHAR  V4DAR                                          16  (2:2, 0:1)
  2-0000007F  CHAR  V5CSR                                          66  (22:24)
  2-000000C1  CHAR  V6CSR                                          90  (26:28)
  2-000003CC  CHAR  V6DAR                                          36  (4:4, 0:1)
  2-000003F0  CHAR  V7DAR                                          15  (6:6)

LABELS

    Address   Label       Address   Label       Address   Label       Address   Label       Address   Label       Address   Label

  1-00000176  5'        1-00000183  10'       1-0000018A  15'       1-00000196  20'       1-000001A1  25'       1-000001A8  30'
      **      35        1-000001B3  40'       1-000001BA  45'       1-000001C6  46'       1-000001F7  47'       1-0000021A  48'
  1-00000233  50'       1-0000023F  55'       1-0000024D  60'       1-00000259  65'       1-00000260  70'       1-00000275  75'

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name                Type  Name                Type  Name                Type  Name                Type  Name                Type  Name

        CALC_MAP                  CALC_MAP2           I*4   COMPRESS4           I*4   COMPRESSC                 DHEAD1                    DQDISKS_QIO
        FRCTOF                    IRP$L_PID                 IRP$Q_IOSB                IRP$W_BCNT                IRP$W_BOFF          I*4   LIB$EXTZV
        LINCHK                    ORB$L_OWNER               OUTPUT                    UBA_DATAPATH              UBA_MAPPING               UCB$B_ERTCNT
        UCB$B_ERTMAX              UCB$L_CHAR                UCB$L_MEDIA               UCB$L_OPCNT               UCB$W_ERRCNT              UCB$W_STS
        VECMAPREG
```

```
	Subroutine DQDISKS_QIO (lun,emb$w_dv_func)


	include 'src$:qiocommon.for /nolist'


	byte		lun
	integer*2	emb$w_dv_func
	integer*4	qiocode(0:1,0:63)


	if (qiocode(0,0) .eq. 0) then
	qiocode(1,00) = %loc(io$_nop)
	qiocode(1,02) = %loc(io$_seek)
	qiocode(1,03) = %loc(io$_recal)
	qiocode(1,04) = %loc(io$_drvclr)
	qiocode(1,08) = %loc(io$_packack)
	qiocode(1,10) = %loc(io$_writecheck)
	qiocode(1,11) = %loc(io$_writepblk)
	qiocode(1,12) = %loc(io$_readpblk)
	qiocode(1,14) = %loc(io$_readhead)
	qiocode(1,26) = %loc(io$_setchar)
	qiocode(1,27) = %loc(io$_sensechar)
	qiocode(1,32) = %loc(io$_writelblk)
	qiocode(1,33) = %loc(io$_readlblk)
	qiocode(1,35) = %loc(io$_setmode)
	qiocode(1,39) = %loc(io$_sensemode)
	qiocode(1,48) = %loc(io$_writevblk)
```

```
        qiocode(1,49) = %loc(io$_readvblk)

        qiocode(1,50) = %loc(io$_access)

        qiocode(1,51) = %loc(io$_create)

        qiocode(1,52) = %loc(io$_deaccess)

        qiocode(1,53) = %loc(io$_delete)

        qiocode(1,54) = %loc(io$_modify)

        qiocode(1,56) = %loc(io$_acpcontrol)

        qiocode(1,57) = %loc(io$_mount)

        do 10,i = 0,63

        qiocode(0,i) = 33

        if (qiocode(1,i) .eq. 0) then

        qiocode(1,i) = %loc(qio_string)
        endif

10      continue
        endif

        call irp$w_func (lun,emb$w_dv_func,
        1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))

        return

        end
```

```
                                                        C  8
DQDISKS_QIO                                        16-Sep-1984 00:02:07    VAX-11 FORTRAN V3.4-56          Page  16
                                                    5-Sep-1984 13:52:37    DISK$VMSMASTER:[ERF.SRC]DQDISKS.FOR;1

PROGRAM SECTIONS

    Name                                  Bytes   Attributes

  0 $CODE                                   255   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                  2058

ENTRY POINTS

    Address   Type  Name

  0-00000000        DQDISKS_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$_READHEAD
  3-00000286  CHAR  IO$_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-00000004@ 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
```

```
0001
0002


COMMAND QUALIFIERS

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

  /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:           10.94 seconds
  Elapsed Time:       25.02 seconds
  Page Faults:        269
  Dynamic Memory:     248 pages
```

0147 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY