```
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**DUTUDRIVR

```
DDDDDDDD    UU      UU  TTTTTTTTTT  UU      UU  DDDDDDDD    RRRRRRRR     IIIIII    VV      VV  RRRRRRRR
DDDDDDDD    UU      UU  TTTTTTTTTT  UU      UU  DDDDDDDD    RRRRRRRR     IIIIII    VV      VV  RRRRRRRR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RR      RR     II      VV      VV  RR      RR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RR      RR     II      VV      VV  RR      RR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RR      RR     II      VV      VV  RR      RR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RR      RR     II      VV      VV  RR      RR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RRRRRRRR       II      VV      VV  RRRRRRRR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RRRRRRRR       II      VV      VV  RRRRRRRR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RR  RR         II      VV      VV  RR  RR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RR  RR         II      VV      VV  RR  RR
DD      DD  UU      UU      TT      UU      UU  DD      DD  RR    RR       II        VV  VV    RR    RR      ....
DD      DD  UU      UU      TT      UU      UU  DD      DD  RR    RR       II        VV  VV    RR    RR      ....
DDDDDDDD    UUUUUUUUUU      TT      UUUUUUUUUU  DDDDDDDD    RR      RR   IIIIII      VV      RR      RR    ....
DDDDDDDD    UUUUUUUUUU      TT      UUUUUUUUUU  DDDDDDDD    RR      RR   IIIIII      VV      RR      RR    ....
```

```
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   10-FEB-1982
0029
0030     c++
0031     c       Functional description:
0032     c
0033     c       This module displays entries logged by MSCP disks (dudriver) and
0034     c       and MSCP tapes (tudriver).
0035     c
0036     c       Modified by:
0037     c
0038     C       V03-007 EAD0200         Elliott A. Drayton              23-Jul-1984
0039     C               Added code to mark the begining of the intervening entries.
0040     C
0041     C       V03-006 SAR0272         Sharon A. Reynolds      18-Jun-1984
0042     C               - Re-structured and re-named the routines in this
0043     C               module to handle disk or tape MSCP entries for the
0044     C               addition of TMSCP support.
0045     C
0046     C       V03-005 SAR0197         Sharon A. Reynolds,     20-Feb-1984
0047     C               Added an SYE update that:
0048     C               - Removed 'invalid mscp command end message'.
0049     C
0050     C       V03-004 SAR0157         Sharon A. Reynolds,     12-Oct-1983
0051     C               Added an SYE update that:
0052     C               - adds an extra arguement to the 'dudriver_mscp_dispatcher'
0053     C                 routine.
0054     C               - adds an extra arguement to the call for the
0055     C                 'dudriver_mscp_dispatcher' routine.
0056     C               - adds an extra arguement to the calls for several
0057     C                 routines that reside in 'mscp.for'.
```

```
0058     C
0059     C       V03-003 SAR0072         Sharon A. Reynolds,     20-Jun-1983
0060     C               Changed the carriage control in the 'format' statements
0061     C               for use with ERF.
0062     C
0063     c       v03-002 BP0002          Brian Porter,           08-FEB-1983
0064     c               Corrected argument list to erllogmsg2.
0065     c
0066     c       v03-001 BP0001          Brian Porter,           19-APR-1982
0067     c               Made changes to accomodate invalid command mscp messages.
0068     c**
0069     c--
0070
0071             Subroutine DISK_TAPE_DRVR_MSCP_DISPATCHER (lun,option,reccnt,
0072             1 mount_flag_and_label,record_length,queue_count)
0073
0074
0075             include 'src$:msghdr.for /nolist'
0134             include 'src$:emblmdef.for /nolist'
0203             include 'src$:embspdef.for /nolist'
0316
0317
0318             byte            lun
0319
0320             character*1     option
0321
0322     c  This value RECCNT is not the record number of the entry just read from the
0323     c  errlog.sys file it is the value which was saved in the queue when this
0324     c  routine is called by _DQ.
0325             integer*4       reccnt
0326     c
0327             integer*4       mount_flag_and_label
0328             integer*4       record_length
0329             Integer*4       queue_count
0330             Integer*4       packet_length
0331
0332             byte            mslg$b_format
0333             equivalence     (emb(46),mslg$b_format)
0334
0335
0336             if (emb$w_hd_entry .eq. 100) then               ! Logmessage entry
0337
0338     C
0339     C Determine whether to output the long or short header and call
0340     C the appropriate routine.
0341     C
0342             If (queue_count .EQ. 1) then
0343
0344             Call FRCTOF (lun)
0345             Call HEADER2 (lun,reccnt)
0346             Else
0347
0348             Call HEADER3 (lun,reccnt)
0349             Endif
0350
0351             Call LOGGER (lun,'ERL$LOGMESSAGE ENTRY')
0352
```

```
0353            Call DHEAD3 (lun,'I/O',emb$b_lm_naming,emb$t_lm_name,emb$w_lm_unit,
0354            1 mount_flag_and_label)
0355
0356            Packet_length = record_length - 39
0357
0358            if (mslg$b_format .eq. 0) then          ! Controller error
0359
0360            if (option .eq. 'S') then
0361            Call MSLG$K_CNT_ERR (lun,packet_length)
0362            endif
0363
0364            else if (mslg$b_format .eq. 1) then     ! Memory access error
0365
0366            if (option .eq. 'S') then
0367            Call MSLG$K_BUS_ADDR (lun,packet_length)
0368            endif
0369
0370            else if (               ! Disk/tape transfer error
0371            1 mslg$b_format .eq. 2  ! mslg$k_disk_trn
0372            1 .OR.
0373            1 mslg$b_format .EQ. 5  ! mslg$k_tape_trn
0374            1 ) then
0375
0376            if (option .eq. 'S') then
0377            Call DISK_TAPE_TRANSFER_ERRORS (lun,packet_length)
0378            endif
0379
0380            else if (               ! SDI/STI errors
0381            1 mslg$b_format .eq. 3  ! Disk SDI comm error - mslg$k_sdi
0382            1 .OR.
0383            1 mslg$b_format .EQ. 6  ! Tape STI comm or cmd failure - mslg$k_sti_err
0384            1 .OR.
0385            1 mslg$b_format .EQ. 7  ! Tape STI Drive Error Log - mslg$k_sti_del
0386            1 .OR.
0387            1 mslg$b_format .EQ. 8  ! Tape STI Formatter Error Log - mslg$k_sti_fel
0388            1 ) then
0389
0390            if (option .eq. 'S') then
0391            Call SDI_STI_ERRORS (lun,packet_length)
0392            endif
0393
0394            else if (mslg$b_format .eq. 4) then     ! Small Disk error
0395
0396            if (option .eq. 'S') then
0397            Call MSLG$K_SML_DSK (lun,packet_length)
0398            endif
0399
0400            else
0401    C
0402    C Unknown format type, call a routine that will decode/output the header
0403    C information and dump the rest of the packet in a hex longword format.
0404    C
0405            Call ERLLOGMSG2 (lun,record_length)
0406            endif
0407
0408            else if (emb$w_hd_entry .eq. 99) then   ! Logstatus entry
0409
```

```
0410     C
0411     C Determine whether to output the long or short header and call
0412     C the appropriate routine.
0413     C
0414             If (queue_count .EQ. 1) then
0415             Call FRCTOF (lun)
0416             Call HEADER2 (lun,reccnt)
0417
0418             Else
0419             Call HEADER3 (lun,reccnt)
0420             Endif
0421
0422             Call LOGGER (lun,'ERL$LOGSTATUS ENTRY')
0423
0424             Call DHEAD3 (lun,'I/O',emb$b_sp_namlng,emb$t_sp_name,emb$w_sp_unit,
0425             1 mount_flag_and_label)
0426
0427             Call ERLLOGSTS2 (lun)
0428             endif
0429
0430             return
0431             end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	426	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	45	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	164	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	1147	

ENTRY POINTS

Address	Type	Name
0-00000000		DISK_TAPE_DRVR_MSCP_DISPATCHER

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000010	L*1	EMB$B_LM_CLASS	3-00000014	L*1	EMB$B_LM_NAMLNG
3-00000011	L*1	EMB$B_LM_TYPE	3-00000010	L*1	EMB$B_SP_CLASS
3-00000040	L*1	EMB$B_SP_NAMLNG	3-00000011	L*1	EMB$B_SP_TYPE
3-00000000	I*4	EMB$L_HD_SID	3-00000014	I*4	EMB$L_SP_BCNT
3-00000038	I*4	EMB$L_SP_CHAR	3-0000003C	I*4	EMB$L_SP_CMDREF
3-00000020	I*4	EMB$L_SP_IOSB1	3-00000024	I*4	EMB$L_SP_IOSB2
3-00000018	I*4	EMB$L_SP_MEDIA	3-0000002C	I*4	EMB$L_SP_OPCNT
3-00000034	I*4	EMB$L_SP_OWNUIC	3-0000001C	I*4	EMB$L_SP_RQPID

```
  3-00000015  CHAR EMB$T_LM_NAME
  3-00000004  I*2  EMB$W_HD_ENTRY
  3-00000024  I*2  EMB$W_LM_MSGTYP
  3-00000012  I*2  EMB$W_SP_BOFF
  3-00000028  I*2  EMB$W_SP_FUNC
  3-0000002A  I*2  EMB$W_SP_UNIT
 AP-00000010@ I*4  MOUNT_FLAG_AND_LABEL
 AP-00000008@ CHAR OPTION
 AP-00000018@ I*4  QUEUE_COUNT
 AP-0000014@ I*4  RECORD_LENGTH
  3-00000041  CHAR EMB$T_SP_NAME
  3-0000000E  I*2  EMB$W_HD_ERRSEQ
  3-00000012  I*2  EMB$W_LM_UNIT
  3-00000030  I*2  EMB$W_SP_ERRCNT
  3-00000032  I*2  EMB$W_SP_STS
 AP-00000004@ L*1  LUN
  3-0000002E  L*1  MSLG$B_FORMAT
  2-00000000  I*4  PACKET_LENGTH
 AP-0000000C@ I*4  RECCNT
```

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	L*1	EMB	512	(0:511)
3-00000026	L*1	EMB$B_LM_MSGTXT	460	(460)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	DHEAD3		DISK_TAPE_TRANSFER_ERRORS		ERLLOGMSG2
	ERLLOGSTS2		FRCTOF		HEADER2
	HEADER3		LOGGER		MSLG$K_BUS_ADDR
	MSLG$K_CNT_ERR		MSLG$K_SML_DSK		SDI_STI_ERRORS

```
0001
0002
0003
0004    c++
0005    c       Since mscp error log entries are delivered from the port via
0006    c       the datagram service it is possible for them to be delivered
0007    c       out of sequence or duplicated.  It is the responsibility of
0008    c       this queue to collect all entries containing the same command
0009    c       reference for a given cpu together.  They are placed in order
0010    c       of error log entry type.
0011    c
0012    c       The format of the elements are as follows
0013    c
0014    c       +-----------------------------------+
0015    c       |              flink1               |
0016    c       +-----------------------------------+
0017    c       |              blink1               |
0018    c       +-----------------------------------+
0019    c       |           logging sid             |
0020    c       +-----------------------------------+
0021    c       |   root command reference flink    |
0022    c       +-----------------------------------+
0023    c       |   root command reference blink    |
0024    c       +-----------------------------------+
0025    c       |   command reference entry count   |
0026    c       +-----------------------------------+
0027    c
0028    c       +-----------------------------------+
0029    c       |              flink2               |
0030    c       +-----------------------------------+
0031    c       |              blink2               |
0032    c       +-----------------------------------+
0033    c       |     command reference number      |
0034    c       +-----------------------------------+
0035    c       |    root_emb$$w_hd_entry_flink     |
0036    c       +-----------------------------------+
0037    c       |    root_emb$$w_hd_entry_blink     |
0038    c       +-----------------------------------+
0039    c       |      emb$$w_hd_entry count        |
0040    c       +-----------------------------------+
0041    c
0042    c       +-----------------------------------+
0043    c       |              flink3               |
0044    c       +-----------------------------------+
0045    c       |              blink3               |
0046    c       +-----------------------------------+
0047    c       |          emb$$w_hd_entry          |
0048    c       +-----------------------------------+
0049    c       |      error log record number      |
0050    c       +-----------------------------------+
0051    c       |   error log record size (bytes)   |
0052    c       +-----------------------------------+
0053    c       |                                   |
0054    c       +--                               --+
0055    c       .                                   .
0056    c       .        error log record           .
0057    c       .                                   .
```

```
0058     c        +--                                  --+
0059     c        ¦                                      ¦
0060     c        +--------------------------------------+
0061     c        ¦          mounted flag                ¦
0062     c        +--------------------------------------+
0063     c        ¦                                      ¦
0064     c        +--                                  --+
0065     c        ¦   mounted volume label (if any)      ¦
0066     c        +--                                  --+
0067     c        ¦                                      ¦
0068     c        +--------------------------------------+
0069     c--
0070
0071
0072              Subroutine DISK_TAPE_DRIVERS_MSCP_Q (record_length,reccnt,
0073              1 search_command_reference_number)
0074
0075
0076              include 'src$:msghdr.for /nolist'
0077              include 'src$:emblmdef.for /nolist'
0136              include 'src$:embspdef.for /nolist'
0205
0318
0319
0320              byte              lun
0321
0322              character*1       option
0323
0324              integer*4         record_length
0325              integer*4         reccnt
0326              integer*4         search_command_reference_number
0327              integer*4         buffer(2)
0328              logical*4         lib$get_vm
0329              integer*4         insert_blink
0330              integer*4         logging_sid_entry_address
0331              integer*4         command_reference_entry_address
0332              integer*4         emb$$w_hd_entry_address
0333              integer*4         root_logging_sid_flink
0334              integer*4         root_logging_sid_blink
0335
0336              equivalence       (buffer(1),root_logging_sid_flink)
0337              equivalence       (buffer(2),root_logging_sid_blink)
0338
0339              integer*4         logging_sid_entry_count
0340              data              logging_sid_entry_count /0/
0341
0342              integer*4         buffer1(6)
0343              integer*4         flink1
0344              integer*4         blink1
0345              integer*4         logging_sid
0346              integer*4         root_command_reference_flink
0347              integer*4         root_command_reference_blink
0348              integer*4         command_reference_entry_count
0349
0350              equivalence       (buffer1(1),flink1)
0351              equivalence       (buffer1(2),blink1)
0352
```

```
0353            equivalence     (buffer1(3),logging_sid)
0354            equivalence     (buffer1(4),root_command_reference_flink)
0355            equivalence     (buffer1(5),root_command_reference_blink)
0356            equivalence     (buffer1(6),command_reference_entry_count)
0357
0358            integer*4       buffer2(6)
0359            integer*4       flink2
0360            integer*4       blink2
0361            integer*4       command_reference_number
0362            integer*4       root_emb$$w_hd_entry_flink
0363            integer*4       root_emb$$w_hd_entry_blink
0364            integer*4       emb$$w_hd_entry_count
0365
0366            equivalence     (buffer2(1),flink2)
0367            equivalence     (buffer2(2),blink2)
0368            equivalence     (buffer2(3),command_reference_number)
0369            equivalence     (buffer2(4),root_emb$$w_hd_entry_flink)
0370            equivalence     (buffer2(5),root_emb$$w_hd_entry_blink)
0371            equivalence     (buffer2(6),emb$$w_hd_entry_count)
0372
0373            integer*4       buffer3(5)
0374            integer*4       flink3
0375            integer*4       blink3
0376            integer*4       emb$$w_hd_entry
0377            integer*4       error_log_record_number
0378            integer*4       error_log_record_length
0379
0380            equivalence     (buffer3(1),flink3)
0381            equivalence     (buffer3(2),blink3)
0382            equivalence     (buffer3(3),emb$$w_hd_entry)
0383            equivalence     (buffer3(4),error_log_record_number)
0384            equivalence     (buffer3(5),error_log_record_length)
0385
0386
0387            if (logging_sid_entry_count .eq. 0) then
0388
0389            root_logging_sid_flink = %loc(root_logging_sid_flink)
0390            root_logging_sid_blink = root_logging_sid_flink
0391            endif
0392
0393            logging_sid_entry_address = root_logging_sid_flink
0394
0395            do 100,i = 1,logging_sid_entry_count
0396
0397            call movc3 (%val(24),%val(logging_sid_entry_address),buffer1)
0398
0399            if (logging_sid .eq. emb$l_hd_sid) then
0400
0401    10      command_reference_entry_address = root_command_reference_flink
0402
0403            do 90,j = 1,command_reference_entry_count
0404
0405            call movc3 (%val(24),%val(command_reference_entry_address),buffer2)
0406
0407            if (command_reference_number .eq. search_command_reference_number)
0408            1 then
0409
```

```
0410     25       insert_blink = root_emb$$w_hd_entry_blink
0411
0412              if (emb$$w_hd_entry_count .ne. 0) then
0413
0414              call movc3 (%val(12),%val(root_emb$$w_hd_entry_blink),buffer3)
0415
0416              if (emb$$w_hd_entry .lt. emb$w_hd_entry) then
0417
0418              insert_blink = blink3
0419              endif
0420              endif
0421
0422              call movc5 (%val(0),,%val(0),%val(20),buffer3)
0423
0424              if (lib$get_vm((20+record_length+16),emb$$w_hd_entry_address)) then
0425
0426              call insque (%val(emb$$w_hd_entry_address),%val(insert_blink))
0427
0428              emb$$w_hd_entry = emb$w_hd_entry
0429
0430              error_log_record_number = reccnt
0431
0432              error_log_record_length = record_length
0433
0434              call movc3 (%val(12),emb$$w_hd_entry,
0435             1 %val(emb$$w_hd_entry_address + 8))
0436
0437              call movc3 (%val(record_length),emb,%val(emb$$w_hd_entry_address + 20))
0438
0439              call movl (-1,%val(emb$$w_hd_entry_address+20+record_length))
0440
0441              if (emb$w_hd_entry .eq. 100) then
0442
0443              call get_current_label (3,emb$l_hd_sid,emb$b_lm_namlng,emb$t_lm_name,
0444             1 emb$w_lm_unit,%val(emb$$w_hd_entry_address+20+record_length+4),*30)
0445
0446              else if (emb$w_hd_entry .eq. 99) then
0447
0448              call get_current_label (3,emb$l_hd_sid,emb$b_sp_namlng,emb$t_sp_name,
0449             1 emb$w_sp_unit,%val(emb$$w_hd_entry_address+20+record_length+4),*30)
0450              endif
0451
0452              call movl (emb$$w_hd_entry_address+20+record_length+4,
0453             1 %val(emb$$w_hd_entry_address+20+record_length))
0454
0455     30       emb$$w_hd_entry_count = emb$$w_hd_entry_count + 1
0456
0457              call movl (emb$$w_hd_entry_count,
0458             1 %val(command_reference_entry_address + 20))
0459              endif
0460
0461              return
0462              endif
0463
0464              command_reference_entry_address = flink2
0465
0466     90       continue
```

```
0467
0468            call movc5 (%val(0),,%val(0),%val(24),buffer2)
0469
0470            if (lib$get_vm(24,command_reference_entry_address)) then
0471
0472            call insque (%val(command_reference_entry_address),
0473            1 %val(root_command_reference_blink))
0474
0475            command_reference_number = search_command_reference_number
0476
0477            root_emb$$w_hd_entry_flink = command_reference_entry_address + 12
0478
0479            root_emb$$w_hd_entry_blink = root_emb$$w_hd_entry_flink
0480
0481            call movc3 (%val(16),command_reference_number,
0482            1 %val(command_reference_entry_address + 8))
0483
0484            command_reference_entry_count = command_reference_entry_count + 1
0485
0486            call movl (command_reference_entry_count,
0487            1 %val(logging_sid_entry_address + 20))
0488
0489            goto 25
0490            endif
0491
0492            return
0493            endif
0494
0495            logging_sid_entry_address = flink1
0496
0497    100     continue
0498
0499            call movc5 (%val(0),,%val(0),%val(24),buffer1)
0500
0501            if (lib$get_vm(24,logging_sid_entry_address)) then
0502
0503            call insque (%val(logging_sid_entry_address),
0504            1 %val(root_logging_sid_blink))
0505
0506            logging_sid = emb$l_hd_sid
0507
0508            root_command_reference_flink = logging_sid_entry_address + 12
0509
0510            root_command_reference_blink = root_command_reference_flink
0511
0512            call movc3 (%val(16),logging_sid,%val(logging_sid_entry_address + 8))
0513
0514            logging_sid_entry_count = logging_sid_entry_count + 1
0515
0516            goto 10
0517            endif
0518
0519            return
0520
0521
0522
0523            entry DISK_TAPE_DRIVERS_MSCP_DQ (lun,option)
```

```
0524
0525
0526            logging_sid_entry_address = root_logging_sid_flink
0527
0528            If (logging_sid_entry_count .GT. 0) then
0529                Write (lun, 9000)
0530    9000        Format (////////////////////
0531            1'      B E G I N I N G   O F   I N T E R V E N I N G   E N T R I E S')
0532            Endif
0533
0534            do 150,i = 1,logging_sid_entry_count
0535
0536            call movc3 (%val(24),%val(logging_sid_entry_address),buffer1)
0537
0538            command_reference_entry_address = root_command_reference_flink
0539
0540            do 200,j = 1,command_reference_entry_count
0541
0542            call movc3 (%val(24),%val(command_reference_entry_address),buffer2)
0543
0544            emb$$w_hd_entry_address = root_emb$$w_hd_entry_flink
0545
0546            do 250,k = 1,emb$$w_hd_entry_count
0547
0548            call movc3 (%val(20),%val(emb$$w_hd_entry_address),buffer3)
0549
0550            call movc5 (%val(0),,%val(0),%val(512),emb)
0551
0552            call movc3 (%val(error_log_record_length),
0553            1 %val(emb$$w_hd_entry_address + 20),emb)
0554
0555            call DISK_TAPE_DRVR_MSCP_DISPATCHER (lun,option,
0556            1 error_log_record_number,
0557            1 %val(emb$$w_hd_entry_address+20+error_log_record_length),
0558            1 error_log_record_length,k)
0559
0560            emb$$w_hd_entry_address = flink3
0561
0562    250     continue
0563
0564            command_reference_entry_address = flink2
0565
0566    200     continue
0567
0568            logging_sid_entry_address = flink1
0569
0570    150     continue
0571
0572            return
0573
0574            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	900	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	98	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	548	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	2058	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-00000290		DISK_TAPE_DRIVERS_MSCP_DQ	0-00000000		DISK_TAPE_DRIVERS_MSCP_Q

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000030	I*4	BLINK1	2-00000018	I*4	BLINK2
2-00000004	I*4	BLINK3	2-00000054	I*4	COMMAND_REFERENCE_ENTRY_ADDRESS
2-00000040	I*4	COMMAND_REFERENCE_ENTRY_COUNT	2-0000001C	I*4	COMMAND_REFERENCE_NUMBER
2-00000008	I*4	EMB$$W_AD_ENTRY	2-00000058	I*4	EMB$$W_AD_ENTRY_ADDRESS
2-00000028	I*4	EMB$$W_HD_ENTRY_COUNT	3-00000010	L*1	EMB$B_LM_CLASS
3-00000014	L*1	EMB$B_LM_NAMLNG	3-00000011	L*1	EMB$B_LM_TYPE
3-00000010	L*1	EMB$B_SP_CLASS	3-00000040	L*1	EMB$B_SP_NAMLNG
3-00000011	L*1	EMB$B_SP_TYPE	3-00000000	I*4	EMB$L_HD_SID
3-00000014	I*4	EMB$L_SP_BCNT	3-00000038	I*4	EMB$L_SP_CHAR
3-0000003C	I*4	EMB$L_SP_CMDREF	3-00000020	I*4	EMB$L_SP_IOSB1
3-00000024	I*4	EMB$L_SP_IOSB2	3-00000018	I*4	EMB$L_SP_MEDIA
3-0000002C	I*4	EMB$L_SP_OPCNT	3-00000034	I*4	EMB$L_SP_OWNUIC
3-0000001C	I*4	EMB$L_SP_RQPID	3-00000015	CHAR	EMB$T_LM_NAME
3-00000041	CHAR	EMB$T_SP_NAME	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	3-00000024	I*2	EMB$W_LM_MSGTYP
3-00000012	I*2	EMB$W_LM_UNIT	3-00000012	I*2	EMB$W_SP_BOFF
3-00000030	I*2	EMB$W_SP_ERRCNT	3-00000028	I*2	EMB$W_SP_FUNC
3-00000032	I*2	EMB$W_SP_STS	3-0000002A	I*2	EMB$W_SP_UNIT
2-00000010	I*4	ERROR_LOG_RECORD_LENGTH	2-0000000C	I*4	ERROR_LOG_RECORD_NUMBER
2-0000002C	I*4	FLINK1	2-00000014	I*4	FLINK2
2-00000000	I*4	FLINK3	2-00000060	I*4	I
2-0000004C	I*4	INSERT_BLINK	2-00000064	I*4	J
2-00000068	I*4	K	2-00000034	I*4	LOGGING_SID
2-00000050	I*4	LOGGING_SID_ENTRY_ADDRESS	2-0000005C	I*4	LOGGING_SID_ENTRY_COUNT
AP-0000004@	L*1	LUN	AP-0000008@	CHAR	OPTION
AP-0000008@	I*4	RECCNT	AP-0000004@	I*4	RECORD_LENGTH
2-0000003C	I*4	ROOT_COMMAND_REFERENCE_BLINK	2-00000038	I*4	ROOT_COMMAND_REFERENCE_FLINK
2-00000024	I*4	ROOT_EMB$$W_AD_ENTRY_BLINK	2-00000020	I*4	ROOT_EMB$$W_AD_ENTRY_FLINK
2-00000048	I*4	ROOT_LOGGING_SID_BLINK	2-00000044	I*4	ROOT_LOGGING_SID_FLINK
AP-0000000C@	I*4	SEARCH_COMMAND_REFERENCE_NUMBER			

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000044	I*4	BUFFER	8	(2)
2-0000002C	I*4	BUFFER1	24	(6)
2-00000014	I*4	BUFFER2	24	(6)
2-00000000	I*4	BUFFER3	20	(5)
3-00000000	L*1	EMB	512	(0:511)
3-00000026	L*1	EMB$B_LM_MSGTXT	460	(460)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
0-0000004C	10	0-0000007D	25	0-0000019D	30	**	90	**	100	**	150
**	200	**	250	1-0000000C	9000'						

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	DISK_TAPE_DRVR_MSCP_DISPATCHER		GET_CURRENT_LABEL		INSQUE
L*4	LIB$GET_VM		MOVC3		MOVC5
	MOVL				

```
0001
0002
0003
0004            Subroutine DUDRIVER_QIO (lun,emb$w_dv_func)
0005
0006
0007            include 'src$:qiocommon.for /nolist'
0271
0272
0273
0274            byte            lun
0275
0276            integer*2       emb$w_dv_func
0277
0278            integer*4       qiocode(0:1,0:63)
0279
0280
0281
0282            if (qiocode(0,0) .eq. 0) then
0283
0284            qiocode(1,00) = %loc(io$_nop)
0285            qiocode(1,01) = %loc(io$_unload)
0286            qiocode(1,08) = %loc(io$_packack)
0287
0288            qiocode(1,10) = %loc(io$_writecheck)
0289            qiocode(1,11) = %loc(io$_writepblk)
0290            qiocode(1,12) = %loc(io$_readpblk)
0291
0292            qiocode(1,17) = %loc(io$_available)
0293            qiocode(1,26) = %loc(io$_setchar)
0294            qiocode(1,27) = %loc(io$_sensechar)
0295
0296            qiocode(1,32) = %loc(io$_writelblk)
0297            qiocode(1,33) = %loc(io$_readlblk)
0298            qiocode(1,35) = %loc(io$_setmode)
0299
0300            qiocode(1,39) = %loc(io$_sensemode)
0301            qiocode(1,48) = %loc(io$_writevblk)
0302            qiocode(1,49) = %loc(io$_readvblk)
0303
0304            qiocode(1,50) = %loc(io$_access)
0305            qiocode(1,51) = %loc(io$_create)
0306            qiocode(1,52) = %loc(io$_deaccess)
0307
0308            qiocode(1,53) = %loc(io$_delete)
0309            qiocode(1,54) = %loc(io$_modify)
0310            qiocode(1,56) = %loc(io$_acpcontrol)
0311            qiocode(1,57) = %loc(io$_mount)
0312
0313            do 10,i = 0,63
0314
0315            qiocode(0,i) = 33
0316
0317            if (qiocode(1,i) .eq. 0) then
0318            qiocode(1,i) = %loc(qio_string)
0319            endif
0320
```

```
0321     10      continue
0322             endif
0323
0324             call cdrp$w_func (lun,emb$w_dv_func,
0325             1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))
0326
0327             return
0328             end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	243	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	2046	

ENTRY POINTS

Address	Type	Name
0-00000000		DUDRIVER_QIO

VARIABLES

Address	Type	Name
AP-00000008a	I*2	EMB$W_DV_FUNC
3-00000442	CHAR	IO$_ABORT
3-000003C2	CHAR	IO$_ACPCONTROL
3-00000297	CHAR	IO$_CLEAN
3-00000385	CHAR	IO$_DEACCESS
3-0000026D	CHAR	IO$_DIAGNOSE
3-000004CB	CHAR	IO$_DSE
3-00000276	CHAR	IO$_FORMAT
3-00000014	CHAR	IO$_LOADMCODE
3-000003E2	CHAR	IO$_MOUNT
3-0000009D	CHAR	IO$_OFFSET
3-000000E0	CHAR	IO$_QSTOP
3-00000421	CHAR	IO$_READCSR
3-000002B6	CHAR	IO$_READLBLK
3-00000200	CHAR	IO$_READPRESET
3-0000033A	CHAR	IO$_READVBLK
3-00000484	CHAR	IO$_READWTHXBUF
3-0000007C	CHAR	IO$_RELEASE
3-000001B8	CHAR	IO$_REREADP
3-000002E6	CHAR	IO$_REWIND
3-000000FC	CHAR	IO$_SEARCH
3-00000231	CHAR	IO$_SENSECHAR
2-00000200	I*4	I
3-0000034D	CHAR	IO$_ACCESS
3-000004B3	CHAR	IO$_AVAILABLE
3-00000369	CHAR	IO$_CREATE
3-00000393	CHAR	IO$_DELETE
3-00000065	CHAR	IO$_DRVCLR
3-000000A9	CHAR	IO$_ERASETAPE
3-00000071	CHAR	IO$_INITIALIZE
3-000003A1	CHAR	IO$_MODIFY
3-00000000	CHAR	IO$_NOP
3-000000EB	CHAR	IO$_PACKACK
3-000003EF	CHAR	IO$_RDSTATS
3-00000169	CHAR	IO$_READHEAD
3-0000013F	CHAR	IO$_READPBLK
3-00000195	CHAR	IO$_READTRACKD
3-0000045A	CHAR	IO$_READWTHBUF
3-0000004D	CHAR	IO$_RECAL
3-000001AB	CHAR	IO$_REREADN
3-000000CA	CHAR	IO$_RETCENTER
3-000002C9	CHAR	IO$_REWINDOFF
3-00000024	CHAR	IO$_SEEK
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

        CDRP$W_FUNC        I*4  LIB$EXTZV
```

```
0001
0002
0003
0004          Subroutine TUDRIVER_QIO (lun,emb$w_dv_func)
0005
0006          include 'src$:qiocommon.for /nolist'
0270
0271
0272          byte            lun
0273
0274          integer*2       emb$w_dv_func
0275
0276          integer*4       qiocode(0:1,0:63)
0277
0278
0279
0280          if (qiocode(0,0) .eq. 0) then
0281
0282          qiocode(1,00) = %loc(io$_nop)
0283          qiocode(1,01) = %loc(io$_unload)
0284          qiocode(1,08) = %loc(io$_packack)
0285
0286          qiocode(1,10) = %loc(io$_writecheck)
0287          qiocode(1,11) = %loc(io$_writepblk)
0288          qiocode(1,12) = %loc(io$_readpblk)
0289
0290          qiocode(1,17) = %loc(io$_available)
0291          qiocode(1,21) = %LOC(io$_dse)
0292          qiocode(1,26) = %loc(io$_setchar)
0293
0294          qiocode(1,27) = %loc(io$_sensechar)
0295          qiocode(1,32) = %loc(io$_writelblk)
0296          qiocode(1,33) = %loc(io$_readlblk)
0297
0298          qiocode(1,35) = %loc(io$_setmode)
0299          qiocode(1,39) = %loc(io$_sensemode)
0300          qiocode(1,48) = %loc(io$_writevblk)
0301
0302          qiocode(1,49) = %loc(io$_readvblk)
0303          qiocode(1,50) = %loc(io$_access)
0304          qiocode(1,51) = %loc(io$_create)
0305
0306          qiocode(1,52) = %loc(io$_deaccess)
0307          qiocode(1,53) = %loc(io$_delete)
0308          qiocode(1,54) = %loc(io$_modify)
0309
0310          qiocode(1,56) = %loc(io$_acpcontrol)
0311          qiocode(1,57) = %loc(io$_mount)
0312
0313          do 10,i = 0,63
0314
0315          qiocode(0,i) = 33
0316
0317          if (qiocode(1,i) .eq. 0) then
0318          qiocode(1,i) = %loc(qio_string)
0319          endif
0320
```

```
0321     10      continue
0322             endif
0323
0324             call cdrp$w_func (lun,emb$w_dv_func,
0325             1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))
0326
0327             return
0328             end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	250	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	2053	

ENTRY POINTS

Address	Type	Name
0-00000000		TUDRIVER_QIO

VARIABLES

Address	Type	Name
AP-00000008@	I*2	EMB$W_DV_FUNC
3-00000442	CHAR	IO$_ABORT
3-000003C2	CHAR	IO$_ACPCONTROL
3-00000297	CHAR	IO$_CLEAN
3-00000385	CHAR	IO$_DEACCESS
3-0000026D	CHAR	IO$_DIAGNOSE
3-000004CB	CHAR	IO$_DSE
3-00000276	CHAR	IO$_FORMAT
3-00000014	CHAR	IO$_LOADMCODE
3-000003E2	CHAR	IO$_MOUNT
3-0000009D	CHAR	IO$_OFFSET
3-000000E0	CHAR	IO$_QSTOP
3-00000421	CHAR	IO$_READCSR
3-000002B6	CHAR	IO$_READLBLK
3-00000200	CHAR	IO$_READPRESET
3-0000033A	CHAR	IO$_READVBLK
3-00000484	CHAR	IO$_READWTHXBUF
3-0000007C	CHAR	IO$_RELEASE
3-000001B8	CHAR	IO$_REREADP
3-000002E6	CHAR	IO$_REWIND
3-000000FC	CHAR	IO$_SEARCH
3-00000231	CHAR	IO$_SENSECHAR
2-00000200	I*4	I
3-0000034D	CHAR	IO$_ACCESS
3-000004B3	CHAR	IO$_AVAILABLE
3-00000369	CHAR	IO$_CREATE
3-00000393	CHAR	IO$_DELETE
3-00000065	CHAR	IO$_DRVCLR
3-000000A9	CHAR	IO$_ERASETAPE
3-00000071	CHAR	IO$_INITIALIZE
3-000003A1	CHAR	IO$_MODIFY
3-00000000	CHAR	IO$_NOP
3-000000EB	CHAR	IO$_PACKACK
3-000003EF	CHAR	IO$_RDSTATS
3-00000169	CHAR	IO$_READHEAD
3-0000013F	CHAR	IO$_READPBLK
3-00000195	CHAR	IO$_READTRACKD
3-0000045A	CHAR	IO$_READWTHBUF
3-0000004D	CHAR	IO$_RECAL
3-000001AB	CHAR	IO$_REREADN
3-000000CA	CHAR	IO$_RETCENTER
3-000002C9	CHAR	IO$_REWINDOFF
3-00000024	CHAR	IO$_SEEK
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

        CDRP$W_FUNC        I*4  LIB$EXTZV

COMMAND QUALIFIERS

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

  /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:              11.00 seconds
  Elapsed Time:          28.53 seconds
  Page Faults:           373
  Dynamic Memory:        199 pages
```

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CLASSIFY
LIS

DR750
LIS

CSTRING
LIS

DHEADS
LIS

DR780
LIS

DR11W
LIS

COMPRESS
LIS

DTAILS
LIS

DECODECC
LIS

DUMPREG
LIS

DUTUDRIVR
LIS

CALCMAP
LIS

DUP3271
LIS

CRYPTK
LIS

DQDISKS
LIS

DUP11
LIS

ERFCOM
LIS

ERFMSG
LIS

ERF
LIS

ERF5VECT
LIS

ERF2VECT
LIS

ERFOUTPUT
LIS

ERFCOMVEC
LIS

ERFBRIVEC
LIS

ERF3VECT
LIS

ERFDISKVE
LIS

ERFPARSER
LIS

ERFBUSVEC
LIS

ERFINIVEC
LIS

ERF4VECT
LIS

ERFINICOM
LIS

ERF1VECT
LIS