```
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**MSCP

```
MM      MM    SSSSSSSS    CCCCCCCC  PPPPPPPP
MM      MM    SSSSSSSS    CCCCCCCC  PPPPPPPP
MMMM  MMMM  SS          CC          PP      PP
MMMM  MMMM  SS          CC          PP      PP
MM  MM  MM  SS          CC          PP      PP
MM  MM  MM  SS          CC          PP      PP
MM      MM    SSSSSS    CC          PPPPPPPP
MM      MM    SSSSSS    CC          PPPPPPPP
MM      MM          SS  CC          PP
MM      MM          SS  CC          PP
MM      MM          SS  CC          PP              ....
MM      MM          SS  CC          PP              ....
MM      MM  SSSSSSSS      CCCCCCCC  PP              ....
MM      MM  SSSSSSSS      CCCCCCCC  PP              ....

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
```

MSL

PRO

0
1
2
3

ENT

0

VAR

AP

ARR

LAB

1

```
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 pudriver and dudriver.
0034    c
0035    c       Modified by:
0036    c
0037    C       V03-016 SAR0291         Sharon A. Reynolds      7-Aug-1984
0038    C               Fixed a bug in the starting buffer location for the
0039    C               call the dumpreg in SDI_STI_ERRORS routine.
0040    C
0041    C       V03-015 SAR0287         Sharon A. Reynolds      6-Jul-1984
0042    C               Added call for TU81 device dependent information.
0043    C
0044    C       V03-014 SAR0277         Sharon A. Reynolds      20-Jun-1984
0045    C               Added TMSCP support.
0046    C
0047    C       V03-013 SAR0211         Sharon A. Reynolds,     21-Mar-1984
0048    C               Fixed a bug in the conversion of the volume serial
0049    C               number in the MSLG$$L_VOL_SER routine.
0050    C
0051    C       V03-012 SAR0200         Sharon A. Reynolds,     20-Feb-1984
0052    C               Added an SYE update that:
0053    C               - Corrected formatting problems in AZTEC host access
0054    C               errors.
0055    C               - Added mscp 1.2 ECO. This added controller support for
0056    C               ??????????????????????????????.
0057    C
```

```
0058     C        V03-011 SAR0156         Sharon A. Reynolds,     11-Oct-1983
0059     C        Added an SYE update that:
0060     C                - Corrects the failure code output.
0061     C                - Adds extra controllers to 'last fail' dispatching.
0062     C                - Adds unknown device/controller specific code
0063     C                  extensions.
0064     C                - Adds AZTEC extensions.
0065     C
0066     C        V03-010 SAR0087         Sharon A. Reynolds,     20-Jun-1983
0067     C        Changed the carriage control in the 'format' statements
0068     C        for use with ERF.
0069     C
0070     c        v03-009 BP0009          Brian Porter,           28-FEB-1983
0071     c        Added ra60 byte 15 descriptions.
0072     c
0073     c        v03-008 BP0008          Brian Porter,           03-JAN-1982
0074     c        Corrected spelling error which resulted in the first
0075     c        longword of the sdi status being displayed as zero's.
0076     c
0077     c        v03-007 BP0007          Brian Porter,           15-DEC-1982
0078     c        Corrected symbol error value for data errors and
0079     c        sdi data being incorrectly reported.
0080     c
0081     c        v03-006 BP0006          Brian Porter,           15-NOV-1982
0082     c        Added ra60 etc.  Added sdi stuff to disk transfer error.
0083     c
0084     c        v03-005 BP0005          Brian Porter,           08-NOV-1982
0085     c        Corrected symbol error value for data errors.
0086     c
0087     c        v03-004 BP0004          Brian Porter,           20-AUG-1982
0088     c        Corrected fortran error.
0089     c
0090     c        v03-003 BP0003          Brian Porter,           04-MAY-1982
0091     c        Corrected group bug.
0092     c
0093     c        v03-002 BP0002          Brian Porter,           22-APR-1982
0094     c        Modified invalid command.
0095     c
0096     c        v03-001 BP0001          Brian Porter,           12-APR-1982
0097     c        Added unit available, duplicate unit and invalid command
0098     c        message support.
0099     c**
0100     c--
0101
0102              Subroutine MSCP$$K_OP_AVATN (lun,expand)
0103
0104              include 'src$:msghdr.for /nolist'
0163
0164
0165              byte            lun
0166
0167              logical*1       expand
0168
0169              integer*4       reserved1
0170
0171              integer*2       mscp$w_unit
0172              integer*2       reserved2
```

```
0173
0174            byte            mscp$b_opcode
0175            byte            reserved3
0176
0177            integer*2       reserved4
0178            integer*2       mscp$w_mult_unt
0179            integer*2       mscp$w_unt_flags
0180
0181            integer*4       undefined
0182            integer*4       mscp$l_unit_id1
0183            integer*4       mscp$l_unit_id2
0184            integer*4       mscp$l_media_id
0185
0186            equivalence     (emb(38),reserved1)
0187            equivalence     (emb(42),mscp$w_unit)
0188            equivalence     (emb(44),reserved2)
0189            equivalence     (emb(46),mscp$b_opcode)
0190            equivalence     (emb(47),reserved3)
0191            equivalence     (emb(48),reserved4)
0192            equivalence     (emb(50),mscp$w_mult_unt)
0193            equivalence     (emb(52),mscp$w_unt_flags)
0194            equivalence     (emb(54),undefined)
0195            equivalence     (emb(58),mscp$l_unit_id1)
0196            equivalence     (emb(62),mscp$l_unit_id2)
0197            equivalence     (emb(66),mscp$l_media_id)
0198
0199
0200
0201            call linchk (lun,1)
0202
0203            write(lun,10)
0204    10      format(' ',:)
0205
0206            call mscp_first_twelve_bytes2 (lun,expand)
0207
0208
0209
0210            Entry INVALID_MSCP$$K_OP_AVATN (lun,expand)
0211
0212
0213
0214            call mscp$$w_mult_unt (lun,mscp$w_mult_unt)
0215
0216            call mscp$$w_unt_flgs (lun,mscp$w_unt_flgs,expand)
0217
0218            call linchk (lun,1)
0219
0220            write(lun,15) undefined
0221    15      format(' ',t8,'UNDEFINED',t24,z8.8)
0222
0223            call mscp$$q_unit_id (lun,mscp$l_unit_id1,mscp$l_unit_id2,expand)
0224
0225            call mscp$$l_media_id (lun,mscp$l_media_id,expand)
0226
0227            return
0228
0229            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	193	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	31	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	92	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	828	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-00000042		INVALID_MSCP$$K_OP_AVATN	0-00000000		MSCP$$K_OP_AVATN

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000000	I*4	EMB$L_HD_SID	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	AP-00000008@	L*1	EXPAND
AP-0000004@	L*1	LUN	3-0000002E	L*1	MSCP$B_OPCODE
3-00000042	I*4	MSCP$L_MEDIA_ID	3-0000003A	I*4	MSCP$L_UNIT_ID1
3-0000003E	I*4	MSCP$L_UNIT_ID2	3-00000032	I*2	MSCP$W_MULT_UNT
3-0000002A	I*2	MSCP$W_UNIT	3-00000034	I*2	MSCP$W_UNT_FLAGS
2-00000000	I*4	MSCP$W_UNT_FLGS	3-00000026	I*4	RESERVED1
3-0000002C	I*2	RESERVED2	3-0000002F	L*1	RESERVED3
3-00000030	I*2	RESERVED4	3-00000036	I*4	UNDEFINED

ARRAYS

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

LABELS

Address	Label	Address	Label
1-00000004	10'	1-00000009	15'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	LINCHK		MSCP$$L_MEDIA_ID		MSCP$$Q_UNIT_ID
	MSCP$$W_MULT_UNT		MSCP$$W_UNT_FLGS		MSCP_FIRST_TWELVE_BYTES2

```
0001
0002
0003
0004          Subroutine MSCP$$K_OP_ACPTH (lun,expand)
0005
0006
0007          include 'src$:msghdr.for /nolist'
0066
0067
0068          byte            lun
0069
0070          logical*1       expand
0071
0072          integer*4       reserved1
0073
0074          integer*2       mscp$w_unit
0075          integer*2       reserved2
0076
0077          byte            mscp$b_opcode
0078          byte            reserved3
0079
0080          integer*2       reserved4
0081
0082          equivalence     (emb(38),reserved1)
0083          equivalence     (emb(42),mscp$w_unit)
0084          equivalence     (emb(44),reserved2)
0085          equivalence     (emb(46),mscp$b_opcode)
0086          equivalence     (emb(47),reserved3)
0087          equivalence     (emb(48),reserved4)
0088
0089
0090
0091          Entry MSCP$$K_OP_DUPUN (lun,expand)
0092
0093          call linchk (lun,1)
0094
0095          write(lun,10)
0096    10    format(' ',:)
0097
0098          call mscp_first_twelve_bytes2 (lun,expand)
0099
0100
0101
0102          Entry INVALID_MSCP$$K_OP_ACPTH (lun,expand)
0103
0104          Entry INVALID_MSCP$$K_OP_DUPUN (lun,expand)
0105
0106
0107          return
0108
0109          end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	97	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	9	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	24	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	642	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-0000004C		INVALID_MSCP$$K_OP_ACPTH	0-00000057		INVALID_MSCP$$K_OP_DUPUN
0-00000000		MSCP$$K_OP_ACPTH	0-00000008		MSCP$$K_OP_DUPUN

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000000	I*4	EMB$L_HD_SID	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	AP-00000008@	L*1	EXPAND
AP-0000004@	L*1	LUN	3-0000002E	L*1	MSCP$B_OPCODE
3-0000002A	I*2	MSCP$W_UNIT	3-00000026	I*4	RESERVED1
3-0000002C	I*2	RESERVED2	3-0000002F	L*1	RESERVED3
3-00000030	I*2	RESERVED4			

ARRAYS

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

LABELS

Address	Label
1-00000004	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	LINCHK		MSCP_FIRST_TWELVE_BYTES2

```
0001
0002
0003
0004          Subroutine MSLG$K_CNT_ERR (lun,packet_length)
0005
0006
0007          include 'src$:msghdr.for /nolist'
0066          include 'src$:emblmdef.for /nolist'
0135
0136
0137          byte            lun
0138
0139          Integer*4       packet_length
0140          integer*4       mslg$l_cmd_ref
0141
0142          integer*2       reserved
0143          integer*2       mslg$w_seq_num
0144
0145          byte            mslg$b_format
0146          byte            mslg$b_flags
0147
0148          integer*2       mslg$w_event
0149
0150          integer*4       mslg$l_cnt_id1
0151          integer*4       mslg$l_cnt_id2
0152
0153          byte            mslg$b_cnt_svr
0154          byte            mslg$b_cnt_hvr
0155
0156          integer*2       lastfail_code
0157
0158          equivalence     (emb(38),mslg$l_cmd_ref)
0159          equivalence     (emb(42),reserved)
0160          equivalence     (emb(44),mslg$w_seq_num)
0161          equivalence     (emb(46),mslg$b_format)
0162          equivalence     (emb(47),mslg$b_flags)
0163          equivalence     (emb(48),mslg$w_event)
0164          equivalence     (emb(50),mslg$l_cnt_id1)
0165          equivalence     (emb(54),mslg$l_cnt_id2)
0166          equivalence     (emb(58),mslg$b_cnt_svr)
0167          equivalence     (emb(59),mslg$b_cnt_hvr)
0168          equivalence     (emb(60),lastfail_code)
0169
0170
0171          call linchk (lun,1)
0172
0173          write(lun,10)
0174    10    format(' ',:)
0175
0176          call mslg$$l_cmd_ref (lun,mslg$l_cmd_ref)
0177          call mslg$$w_seq_num (lun,mslg$w_seq_num)
0178          call mslg$$b_format (lun,mslg$b_format)
0179
0180          call mslg$$b_flags (lun,mslg$b_flags)
0181          call mslg$$w_event (lun,mslg$w_event)
0182          call mslg$$q_cnt_id (lun,mslg$l_cnt_id1,mslg$l_cnt_id2)
0183
```

```
0184            call mslg$$b_cnt_svr (lun,mslg$b_cnt_svr)
0185            call mslg$$b_cnt_hvr (lun,mslg$b_cnt_hvr)
0186
0187    C
0188    C       Device dependent extensions...
0189    C
0190            If (packet_length .GT. 22) then
0191
0192            If (
0193            1 (emb$b_lm_class .EQ. '80'X)           ! Bus device class
0194            1 .AND.
0195            1 (emb$b_lm_type .EQ. 3                 ! UDA50, UQPORT
0196            1 .OR.
0197            1 emb$b_lm_type .EQ. 4                  ! UDA50A
0198            1 .OR.
0199            1 emb$b_lm_type .EQ. 6)                 ! TU81P port
0200            1 ) then
0201
0202            Call UDA_LASTFAIL_ERROR (lun,lastfail_code)
0203            Else
0204
0205            Call LINCHK (lun,3)
0206            Write (lun,15) 'CONTROLLER DEPENDENT INFORMATION'
0207    15      Format (/' ',a,/)
0208
0209            Call DUMPREG (lun,(((packet_length - 22) +3)/4),emb(60))
0210            Endif
0211            Endif
0212
0213            return
0214            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	292	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	52	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	160	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	1016	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$K_CNT_ERR

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$B_LM_TYPE	3-00000000	I*4	EMB$L_HD_SID
3-00000015	CHAR	EMB$T_LM_NAME	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	3-00000024	I*2	EMB$W_LM_MSGTYP
3-00000012	I*2	EMB$W_LM_UNIT	3-0000003C	I*2	LASTFAIL_CODE
AP-00000004@	L*1	LUN	3-0000003B	L*1	MSLG$B_CNT_HVR
3-0000003A	L*1	MSLG$B_CNT_SVR	3-0000002F	L*1	MSLG$B_FLAGS
3-0000002E	L*1	MSLG$B_FORMAT	3-00000026	I*4	MSLG$L_CMD_REF
3-00000032	I*4	MSLG$L_CNT_ID1	3-00000036	I*4	MSLG$L_CNT_ID2
3-00000030	I*2	MSLG$W_EVENT	3-0000002C	I*2	MSLG$W_SEQ_NUM
AP-00000008@	I*4	PACKET_LENGTH	3-0000002A	I*2	RESERVED

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)

LABELS

Address	Label	Address	Label
1-00000028	10'	1-0000002D	15'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	DUMPREG		LINCHK		MSLG$$B_CNT_HVR
	MSLG$$B_CNT_SVR		MSLG$$B_FLAGS		MSLG$$B_FORMAT
	MSLG$$L_CMD_REF		MSLG$$Q_CNT_ID		MSLG$$W_EVENT
	MSLG$$W_SEQ_NUM		UDA_LASTFAIL_ERROR		

```
0001
0002
0003
0004
0005          Subroutine MSLG$K_BUS_ADDR (lun,packet_length)
0006
0007
0008          include 'src$:msghdr.for /nolist'
0067          include 'src$:emblmdef.for /nolist'
0136
0137
0138          byte            lun
0139
0140          Integer*4       packet_length
0141          integer*4       mslg$l_cmd_ref
0142
0143          integer*2       reserved1
0144          integer*2       mslg$w_seq_num
0145
0146          byte            mslg$b_format
0147          byte            mslg$b_flags
0148
0149          integer*2       mslg$w_event
0150
0151          integer*4       mslg$l_cnt_id1
0152          integer*4       mslg$l_cnt_id2
0153
0154          byte            mslg$b_cnt_svr
0155          byte            mslg$b_cnt_hvr
0156
0157          integer*2       reserved2
0158
0159          integer*4       mslg$l_bus_addr
0160
0161          equivalence     (emb(38),mslg$l_cmd_ref)
0162          equivalence     (emb(42),reserved1)
0163          equivalence     (emb(44),mslg$w_seq_num)
0164          equivalence     (emb(46),mslg$b_format)
0165          equivalence     (emb(47),mslg$b_flags)
0166          equivalence     (emb(48),mslg$w_event)
0167          equivalence     (emb(50),mslg$l_cnt_id1)
0168          equivalence     (emb(54),mslg$l_cnt_id2)
0169          equivalence     (emb(58),mslg$b_cnt_svr)
0170          equivalence     (emb(59),mslg$b_cnt_hvr)
0171          equivalence     (emb(60),reserved2)
0172          equivalence     (emb(62),mslg$l_bus_addr)
0173
0174
0175          call linchk (lun,1)
0176
0177          write(lun,10)
0178    10    format(' ',:)
0179
0180          call mslg$$l_cmd_ref (lun,mslg$l_cmd_ref)
0181          call mslg$$w_seq_num (lun,mslg$w_seq_num)
0182          call mslg$$b_format (lun,mslg$b_format)
0183
```

```
0184            call mslg$$b_flags (lun,mslg$b_flags)
0185            call mslg$$w_event (lun,mslg$w_event)
0186            call mslg$$q_cnt_id (lun,mslg$l_cnt_id1,mslg$l_cnt_id2)
0187
0188            call mslg$$b_cnt_svr (lun,mslg$b_cnt_svr)
0189            call mslg$$b_cnt_hvr (lun,mslg$b_cnt_hvr)
0190            call mslg$$l_bus_addr (lun,mslg$l_bus_addr)
0191
0192    C
0193    C       Device dependent extensions...
0194    C
0195            If (packet_length .GT. 28) then
0196
0197            Call LINCHK (lun,3)
0198            Write (lun,150) 'CONTROLLER DEPENDENT INFORMATION'
0199    150     Format (/' ',a,/)
0200
0201            If (
0202            1 (emb$b_lm_class .EQ. 1)               ! Disk device class
0203            1 .AND.
0204            1 (emb$b_lm_type .EQ. 23                ! AZTEC removable
0205            1 .OR.
0206            1 emb$b_lm_type .EQ. 24)                ! AZTEC fixed
0207            1 ) then
0208
0209            Call MSLG$K_BUS_ADDR_AZTEC (lun)
0210
0211    C
0212    C       The IF-THEN-ELSE should be expanded at this point to
0213    C       include additional "type" support.
0214    C
0215            Else
0216
0217            Call DUMPREG (lun,(((packet_length - 28) +3)/4),emb(66))
0218            Endif
0219            Endif
0220
0221    200     Continue
0222
0223            return
0224            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	294	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	52	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	168	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL G3L SHR NOEXE RD WRT LONG
	Total Space Allocated	1026	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$K_BUS_ADDR

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$B_LM_TYPE	3-00000000	I*4	EMB$L_HD_SID
3-00000015	CHAR	EMB$T_LM_NAME	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	3-00000024	I*2	EMB$W_LM_MSGTYP
3-00000012	I*2	EMB$W_LM_UNIT	AP-00000004@	L*1	LUN
3-0000003B	L*1	MSLG$B_CNT_HVR	3-0000003A	L*1	MSLG$B_CNT_SVR
3-0000002F	L*1	MSLG$B_FLAGS	3-0000002E	L*1	MSLG$B_FORMAT
3-0000003E	I*4	MSLG$L_BUS_ADDR	3-00000026	I*4	MSLG$L_CMD_REF
3-00000032	I*4	MSLG$L_CNT_ID1	3-00000036	I*4	MSLG$L_CNT_ID2
3-00000030	I*2	MSLG$W_EVENT	3-0000002C	I*2	MSLG$W_SEQ_NUM
AP-00000008@	I*4	PACKET_LENGTH	3-0000002A	I*2	RESERVED1
3-0000003C	I*2	RESERVED2			

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)

LABELS

Address	Label	Address	Label	Address	Label
1-00000028	10'	1-0000002D	150'	**	200

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	DUMPREG		LINCHK		MSLG$$B_CNT_HVR
	MSLG$$B_CNT_SVR		MSLG$$B_FLAGS		MSLG$$B_FORMAT
	MSLG$$L_BUS_ADDR		MSLG$$L_CMD_REF		MSLG$$Q_CNT_ID
	MSLG$$W_EVENT		MSLG$$W_SEQ_NUM		MSLG$K_BUS_ADDR_AZTEC

```
0001
0002
0003
0004          Subroutine DISK_TAPE_TRANSFER_ERRORS (lun,packet_length)
0005
0006          include 'src$:msghdr.for /nolist'
0065          include 'src$:emblmdef.for /nolist'
0134
0135
0136          byte            lun
0137
0138          Integer*4       packet_length
0139          integer*4       mslg$l_cmd_ref
0140
0141          integer*2       mslg$w_unit
0142          integer*2       mslg$w_seq_num
0143
0144          byte            mslg$b_format
0145          byte            mslg$b_flags
0146
0147          integer*2       mslg$w_event
0148
0149          integer*4       mslg$l_cnt_id1
0150          integer*4       mslg$l_cnt_id2
0151
0152          byte            mslg$b_cnt_svr
0153          byte            mslg$b_cnt_hvr
0154
0155          integer*2       mslg$w_mult_unt
0156
0157          integer*4       mslg$l_unit_id1
0158          integer*4       mslg$l_unit_id2
0159
0160          byte            mslg$b_unit_svr
0161          byte            mslg$b_unit_hvr
0162          byte            mslg$b_level
0163          byte            mslg$b_retry
0164
0165          integer*4       mslg$l_vol_ser
0166          integer*4       mslg$l_header
0167
0168          equivalence     (emb(38),mslg$l_cmd_ref)
0169          equivalence     (emb(42),mslg$w_unit)
0170          equivalence     (emb(44),mslg$w_seq_num)
0171          equivalence     (emb(46),mslg$b_format)
0172          equivalence     (emb(47),mslg$b_flags)
0173          equivalence     (emb(48),mslg$w_event)
0174          equivalence     (emb(50),mslg$l_cnt_id1)
0175          equivalence     (emb(54),mslg$l_cnt_id2)
0176          equivalence     (emb(58),mslg$b_cnt_svr)
0177          equivalence     (emb(59),mslg$b_cnt_hvr)
0178          equivalence     (emb(60),mslg$w_mult_unt)
0179          equivalence     (emb(62),mslg$l_unit_id1)
0180          equivalence     (emb(66),mslg$l_unit_id2)
0181          equivalence     (emb(70),mslg$b_unit_svr)
0182          equivalence     (emb(71),mslg$b_unit_hvr)
0183          equivalence     (emb(72),mslg$b_level)
```

```
0184            equivalence     (emb(73),mslg$b_retry)
0185            equivalence     (emb(74),mslg$l_vol_ser)
0186            equivalence     (emb(78),mslg$l_header)
0187
0188
0189            call linchk (lun,1)
0190
0191            write(lun,10)
0192    10      format(' ',:)
0193
0194            call mslg$$l_cmd_ref (lun,mslg$l_cmd_ref)
0195            call mslg$$w_unit (lun,mslg$w_unit,.true.)
0196            call mslg$$w_seq_num (lun,mslg$w_seq_num)
0197
0198            call mslg$$b_format (lun,mslg$b_format)
0199            call mslg$$b_flags (lun,mslg$b_flags)
0200            call mslg$$w_event (lun,mslg$w_event)
0201
0202            call mslg$$q_cnt_id (lun,mslg$l_cnt_id1,mslg$l_cnt_id2)
0203            call mslg$$b_cnt_svr (lun,mslg$b_cnt_svr)
0204            call mslg$$b_cnt_hvr (lun,mslg$b_cnt_hvr)
0205
0206            call mslg$$w_mult_unt (lun,mslg$w_mult_unt)
0207            call mslg$$q_unit_id (lun,mslg$l_unit_id1,mslg$l_unit_id2,.true.)
0208            call mslg$$b_unit_svr (lun,mslg$b_unit_svr)
0209
0210            call mslg$$b_unit_hvr (lun,mslg$b_unit_hvr)
0211            call mslg$$b_level (lun,mslg$b_level)
0212            call mslg$$b_retry (lun,mslg$b_retry)
0213
0214    C
0215    C At this point the disk and tape transfer error entries differ,
0216    C determine whether outputting a disk or tape transfer error and
0217    C call the appropriate routines.
0218    C
0219            If (mslg$b_format .EQ. 2) then                  ! Disk transfer error
0220              call mslg$$l_vol_ser (lun,mslg$l_vol_ser)
0221              call mslg$$l_header (lun,mslg$l_header)
0222
0223            Else if (mslg$b_format .EQ. 5) then             ! Tape transfer error
0224              Call MSLG$$L_GAP_CNT (lun,emb(74))
0225              Call FORMATTER_VERSIONS (lun,emb(78),emb(79))
0226            Endif
0227
0228
0229    C
0230    C       Device dependent extensions...
0231    C
0232            If (packet_length .GT. 44) then
0233
0234    C
0235    C Determine whether this entry applies to a TU81 transfer and
0236    C call the appropriate routine to decode/output the device
0237    C dependent information.
0238    C
0239            If (
0240            1 (mslg$b_format .EQ. 5)                ! Tape transfer error
```

```
0241            1 .AND.
0242            1 (emb$b_lm_type .EQ. 8)                ! TU81 device
0243            1 ) then
0244              Call TU81_SENSE_BYTES_DECODE (lun)
0245              Return
0246            Endif
0247
0248      C
0249      C Output the controller or device dependent information in a
0250      C hex longword format.
0251      C
0252            Call LINCHK (lun,3)
0253            Write (lun,15) 'CONTROLLER OR DEVICE DEPENDENT INFORMATION'
0254      15    Format (/' ',a,/)
0255
0256            Call DUMPREG (lun,(((packet_length - 44) +3)/4),emb(82))
0257            Endif
0258
0259            return
0260            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	455	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	66	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	304	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	1337	

ENTRY POINTS

Address	Type	Name
0-00000000		DISK_TAPE_TRANSFER_ERRORS

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$B_LM_TYPE	3-00000000	I*4	EMB$L_HD_SID
3-00000015	CHAR	EMB$T_LM_NAME	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	3-00000024	I*2	EMB$W_LM_MSGTYP
3-00000012	I*2	EMB$W_LM_UNIT	AP-0000004@	L*1	LUN
3-0000003B	L*1	MSLG$B_CNT_HVR	3-0000003A	L*1	MSLG$B_CNT_SVR
3-0000002F	L*1	MSLG$B_FLAGS	3-0000002E	L*1	MSLG$B_FORMAT
3-00000048	L*1	MSLG$B_LEVEL	3-00000049	L*1	MSLG$B_RETRY
3-00000047	L*1	MSLG$B_UNIT_HVR	3-00000046	L*1	MSLG$B_UNIT_SVR
3-00000026	I*4	MSLG$L_CMD_REF	3-00000032	I*4	MSLG$L_CNT_ID1

```
  3-00000036  I*4  MSLG$L_CNT_ID2          3-0000004E  I*4  MSLG$L_HEADER
  3-0000003E  I*4  MSLG$L_UNIT_ID1         3-00000042  I*4  MSLG$L_UNIT_ID2
  3-0000004A  I*4  MSLG$L_VOL_SER          3-00000030  I*2  MSLG$W_EVENT
  3-0000003C  I*2  MSLG$W_MULT_UNT         3-0000002C  I*2  MSLG$W_SEQ_NUM
  3-0000002A  I*2  MSLG$W_UNIT            AP-00000008@ I*4  PACKET_LENGTH
```

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)
```

LABELS

```
    Address   Label        Address   Label

  1-00000036  10'        1-0000003B  15'
```

FUNCTIONS AND SUBROUTINES REFERENCED

```
  Type  Name                          Type  Name                          Type  Name

        DUMPREG                             FORMATTER_VERSIONS                  LINCHK
        MSLG$$B_CNT_HVR                     MSLG$$B_CNT_SVR                     MSLG$$B_FLAGS
        MSLG$$B_FORMAT                      MSLG$$B_LEVEL                       MSLG$$B_RETRY
        MSLG$$B_UNIT_HVR                    MSLG$$B_UNIT_SVR                    MSLG$$L_CMD_REF
        MSLG$$L_GAP_CNT                     MSLG$$L_HEADER                      MSLG$$L_VOL_SER
        MSLG$$Q_CNT_ID                      MSLG$$Q_UNIT_ID                     MSLG$$W_EVENT
        MSLG$$W_MULT_UNT                    MSLG$$W_SEQ_NUM                     MSLG$$W_UNIT
        TU81_SENSE_BYTES_DECODE
```

MS

01

```
0001
0002
0003
0004          Subroutine SDI_STI_ERRORS (lun,packet_length)
0005
0006
0007          include 'src$:msghdr.for /nolist'
0066          include 'src$:emblmdef.for /nolist'
0135
0136
0137          byte              lun
0138
0139          Integer*4         packet_length
0140          integer*4         mslg$l_cmd_ref
0141
0142          integer*2         mslg$w_unit
0143          integer*2         mslg$w_seq_num
0144
0145          byte              mslg$b_format
0146          byte              mslg$b_flags
0147
0148          integer*2         mslg$w_event
0149
0150          integer*4         mslg$l_cnt_id1
0151          integer*4         mslg$l_cnt_id2
0152
0153          byte              mslg$b_cnt_svr
0154          byte              mslg$b_cnt_hvr
0155
0156          integer*2         mslg$w_mult_unt
0157
0158          integer*4         mslg$l_unit_id1
0159          integer*4         mslg$l_unit_id2
0160
0161          byte              mslg$b_unit_svr
0162          byte              mslg$b_unit_hvr
0163
0164          integer*2         reserved
0165
0166          integer*4         mslg$l_vol_ser
0167          integer*4         mslg$l_header
0168          integer*4         mslg$l_sdi1
0169          integer*4         mslg$l_sdi2
0170          integer*4         mslg$l_sdi3
0171
0172          equivalence       (emb(38),mslg$l_cmd_ref)
0173          equivalence       (emb(42),mslg$w_unit)
0174          equivalence       (emb(44),mslg$w_seq_num)
0175          equivalence       (emb(46),mslg$b_format)
0176          equivalence       (emb(47),mslg$b_flags)
0177          equivalence       (emb(48),mslg$w_event)
0178          equivalence       (emb(50),mslg$l_cnt_id1)
0179          equivalence       (emb(54),mslg$l_cnt_id2)
0180          equivalence       (emb(58),mslg$b_cnt_svr)
0181          equivalence       (emb(59),mslg$b_cnt_hvr)
0182          equivalence       (emb(60),mslg$w_mult_unt)
0183          equivalence       (emb(62),mslg$l_unit_id1)
```

```
0184            equivalence     (emb(66),mslg$l_unit_id2)
0185            equivalence     (emb(70),mslg$b_unit_svr)
0186            equivalence     (emb(71),mslg$b_unit_hvr)
0187            equivalence     (emb(72),reserved)
0188            equivalence     (emb(74),mslg$l_vol_ser)
0189            equivalence     (emb(78),mslg$l_header)
0190            equivalence     (emb(82),mslg$l_sdi1)
0191            equivalence     (emb(86),mslg$l_sdi2)
0192            equivalence     (emb(90),mslg$l_sdi3)
0193
0194
0195            call linchk (lun,1)
0196            write(lun,10)
0197    10      format(' ',:)
0198
0199            call mslg$$l_cmd_ref (lun,mslg$l_cmd_ref)
0200            call mslg$$w_unit (lun,mslg$w_unit,.true.)
0201            call mslg$$w_seq_num (lun,mslg$w_seq_num)
0202
0203            call mslg$$b_format (lun,mslg$b_format)
0204            call mslg$$b_flags (lun,mslg$b_flags)
0205            call mslg$$w_event (lun,mslg$w_event)
0206
0207            call mslg$$q_cnt_id (lun,mslg$l_cnt_id1,mslg$l_cnt_id2)
0208            call mslg$$b_cnt_svr (lun,mslg$b_cnt_svr)
0209            call mslg$$b_cnt_hvr (lun,mslg$b_cnt_hvr)
0210
0211            call mslg$$w_mult_unt (lun,mslg$w_mult_unt)
0212            call mslg$$q_unit_id (lun,mslg$l_unit_id1,mslg$l_unit_id2,.true.)
0213            call mslg$$b_unit_svr (lun,mslg$b_unit_svr)
0214            call mslg$$b_unit_hvr (lun,mslg$b_unit_hvr)
0215
0216            If (mslg$b_format .EQ. 3) then  ! SDI comm error - mslg$k_sdi
0217            call mslg$$l_vol_ser (lun,mslg$l_vol_ser)
0218            call mslg$$l_header (lun,mslg$l_header)
0219            call mslg$$z_sdi (lun,mslg$l_sdi1,mslg$l_sdi2,mslg$l_sdi3)
0220
0221            Else if (
0222            1 mslg$b_format .EQ. 6          ! STI comm error - mslg$k_sti_err
0223            1 .OR.
0224            1 mslg$b_format .EQ. 7          ! STI drive error - mslg$k_sti_del
0225            1 .OR.
0226            1 mslg$b_format .EQ. 8          ! STI formatter error - mslg$k_sti_fel
0227            1 ) then
0228              Call MSLG$$L_GAP_CNT (lun,emb(74))
0229              Call FORMATTER_VERSIONS (lun,emb(78),emb(79))
0230            Endif
0231
0232    C
0233    C       Device dependent extensions...
0234    C
0235            If (packet_length .GT. 42) then
0236
0237    !       If (
0238    !       1 emb$b_lm_class .EQ. 2         ! Tape device class
0239    !       1 .OR.
0240    !       1 emb$b_lm_type .EQ. 6          ! TA78 device
```

```
0241     !          1 ) then
0242     !
0243     !            If (mslg$b_format .EQ. 6) then          ! STI comm or cmd failure
0244     !              Call TA78_UNSUCCESSFUL_RESPONSE (lun)
0245     !
0246     !            Else if (mslg$b_format .EQ. 7) then     ! STI drive error
0247     !              Call TA78_EXTENDED_DRIVE_STS (lun)
0248     !
0249     !            Else if (mslg$b_format .EQ. 8) then     ! STI formatter error
0250     !              Call TA78_EXTENDED_FORMATTER_STS (lun)
0251     !            Endif
0252     !          Endif
0253
0254               Call LINCHK (lun,3)
0255               Write (lun,15) 'DEVICE DEPENDENT INFORMATION'
0256     15        Format (/' ',a,/)
0257
0258               I = 94
0259               If ((mslg$b_format .GE. 6)  .AND.  (mslg$b_format .LE. 8)) then
0260                 I = 82
0261               Endif
0262
0263               Call DUMPREG (lun,(((packet_length - 42) +3)/4),emb(I))
0264               Endif
0265
0266               return
0267               end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	449	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	52	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	296	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	1309	

ENTRY POINTS

Address	Type	Name
0-00000000		SDI_STI_ERRORS

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$B_LM_TYPE	3-00000000	I*4	EMB$L_HD_SID
3-00000015	CHAR	EMB$T_LM_NAME	3-00000004	I*2	EMB$W_HD_ENTRY

```
  3-0000000E  I*2  EMB$W_HD_ERRSEQ
  3-00000012  I*2  EMB$W_LM_UNIT
AP-00000004a  L*1  LUN
  3-0000003A  L*1  MSLG$B_CNT_SVR
  3-0000002E  L*1  MSLG$B_FORMAT
  3-00000046  L*1  MSLG$B_UNIT_SVR
  3-00000032  I*4  MSLG$L_CNT_ID1
  3-0000004E  I*4  MSLG$L_HEADER
  3-00000056  I*4  MSLG$L_SDI2
  3-0000003E  I*4  MSLG$L_UNIT_ID1
  3-0000004A  I*4  MSLG$L_VOL_SER
  3-0000003C  I*2  MSLG$W_MULT_UNT
  3-0000002A  I*2  MSLG$W_UNIT
  3-00000048  I*2  RESERVED
  3-00000024  I*2  EMB$W_LM_MSGTYP
  2-00000000  I*4  I
  3-0000C03B  L*1  MSLG$B_CNT_HVR
  3-0000002F  L*1  MSLG$B_FLAGS
  3-00000047  L*1  MSLG$B_UNIT_HVR
  3-00000026  I*4  MSLG$L_CMD_REF
  3-00000036  I*4  MSLG$L_CNT_ID2
  3-00000052  I*4  MSLG$L_SDI1
  3-0000005A  I*4  MSLG$L_SDI3
  3-00000042  I*4  MSLG$L_UNIT_ID2
  3-00000030  I*2  MSLG$W_EVENT
  3-0000002C  I*2  MSLG$W_SEQ_NUM
AP-00000008a  I*4  PACKET_LENGTH
```

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)

LABELS

Address	Label	Address	Label
1-00000028	10'	1-0000002D	15'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	DUMPREG		FORMATTER_VERSIONS		LINCHK
	MSLG$$B_CNT_HVR		MSLG$$B_CNT_SVR		MSLG$$B_FLAGS
	MSLG$$B_FORMAT		MSLG$$B_UNIT_HVR		MSLG$$B_UNIT_SVR
	MSLG$$L_CMD_REF		MSLG$$L_GAP_CNT		MSLG$$L_HEADER
	MSLG$$L_VOL_SER		MSLG$$Q_CNT_ID		MSLG$$Q_UNIT_ID
	MSLG$$W_EVENT		MSLG$$W_MULT_UNT		MSLG$$W_SEQ_NUM
	MSLG$$W_UNIT		MSLG$$Z_SDI		

```
0001
0002
0003
0004          Subroutine MSLG$K_SML_DSK (lun,packet_length)
0005
0006
0007          include 'src$:msghdr.for /nolist'
0066          include 'src$:emblmdef.for /nolist'
0135
0136
0137          byte            lun
0138
0139          Integer*4       packet_length
0140          integer*4       mslg$l_cmd_ref
0141
0142          integer*2       mslg$w_unit
0143          integer*2       mslg$w_seq_num
0144
0145          byte            mslg$b_format
0146          byte            mslg$b_flags
0147
0148          integer*2       mslg$w_event
0149
0150          integer*4       mslg$l_cnt_id1
0151          integer*4       mslg$l_cnt_id2
0152
0153          byte            mslg$b_cnt_svr
0154          byte            mslg$b_cnt_hvr
0155
0156          integer*2       mslg$w_mult_unt
0157
0158          integer*4       mslg$l_unit_id1
0159          integer*4       mslg$l_unit_id2
0160
0161          byte            mslg$b_unit_svr
0162          byte            mslg$b_unit_hvr
0163
0164          integer*2       mslg$w_sde_cyl
0165
0166          integer*4       mslg$l_vol_ser
0167
0168          equivalence     (emb(38),mslg$l_cmd_ref)
0169          equivalence     (emb(42),mslg$w_unit)
0170          equivalence     (emb(44),mslg$w_seq_num)
0171          equivalence     (emb(46),mslg$b_format)
0172          equivalence     (emb(47),mslg$b_flags)
0173          equivalence     (emb(48),mslg$w_event)
0174          equivalence     (emb(50),mslg$l_cnt_id1)
0175          equivalence     (emb(54),mslg$l_cnt_id2)
0176          equivalence     (emb(58),mslg$b_cnt_svr)
0177          equivalence     (emb(59),mslg$b_cnt_hvr)
0178          equivalence     (emb(60),mslg$w_mult_unt)
0179          equivalence     (emb(62),mslg$l_unit_id1)
0180          equivalence     (emb(66),mslg$l_unit_id2)
0181          equivalence     (emb(70),mslg$b_unit_svr)
0182          equivalence     (emb(71),mslg$b_unit_hvr)
0183          equivalence     (emb(72),mslg$w_sde_cyl)
```

```
0184            equivalence     (emb(74),mslg$l_vol_ser)
0185
0186
0187            call linchk (lun,1)
0188
0189            write(lun,10)
0190    10      format(' ',:)
0191
0192            call mslg$$l_cmd_ref (lun,mslg$l_cmd_ref)
0193
0194            call mslg$$w_unit (lun,mslg$w_unit,.true.)
0195
0196            call mslg$$w_seq_num (lun,mslg$w_seq_num)
0197
0198            call mslg$$b_format (lun,mslg$b_format)
0199
0200            call mslg$$b_flags (lun,mslg$b_flags)
0201
0202            call mslg$$w_event (lun,mslg$w_event)
0203
0204            call mslg$$q_cnt_id (lun,mslg$l_cnt_id1,mslg$l_cnt_id2)
0205
0206            call mslg$$b_cnt_svr (lun,mslg$b_cnt_svr)
0207
0208            call mslg$$b_cnt_hvr (lun,mslg$b_cnt_hvr)
0209
0210            call mslg$$w_mult_unt (lun,mslg$w_mult_unt)
0211
0212            call mslg$$q_unit_id (lun,mslg$l_unit_id1,mslg$l_unit_id2,.true.)
0213
0214            call mslg$$b_unit_svr (lun,mslg$b_unit_svr)
0215
0216            call mslg$$b_unit_hvr (lun,mslg$b_unit_hvr)
0217
0218            call mslg$$w_sde_cyl (lun,mslg$w_sde_cyl)
0219
0220            call mslg$$l_vol_ser (lun,mslg$l_vol_ser)
0221
0222    C
0223    C       Device dependent extensions...
0224    C
0225            If (packet_length .GT. 40) then
0226
0227            Call LINCHK (lun,3)
0228            Write (lun,15) 'CONTROLLER OR DEVICE DEPENDENT INFORMATION'
0229    15      Format (/' ',a,/)
0230
0231            If (
0232            1 emb$b_lm_type .EQ. 23
0233            1 .OR.
0234            1 emb$b_lm_type .EQ. 24
0235            1 ) then
0236
0237            Call MSLG$K_SML_DSK_AZTEC (lun)
0238
0239    C
0240    C       The IF-THEN-ELSE should be expanded at this point to
```

```
0241     C        include additional "type" support.
0242     C
0243              Else
0244
0245              Call DUMPREG (lun,(((packet_length - 40) +3)/4),emb(78))
0246              Endif
0247              Endif
0248
0249              return
0250              end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	376	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	66	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	252	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	1206	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$K_SML_DSK

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$B_LM_TYPE	3-00000000	I*4	EMB$L_HD_SID
3-00000015	CHAR	EMB$T_LM_NAME	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	3-00000024	I*2	EMB$W_LM_MSGTYP
3-00000012	I*2	EMB$W_LM_UNIT	AP-0000004@	L*1	LUN
3-0000003B	L*1	MSLG$B_CNT_HVR	3-0000003A	L*1	MSLG$B_CNT_SVR
3-0000002F	L*1	MSLG$B_FLAGS	3-0000002E	L*1	MSLG$B_FORMAT
3-00000047	L*1	MSLG$B_UNIT_HVR	3-00000046	L*1	MSLG$B_UNIT_SVR
3-00000026	I*4	MSLG$L_CMD_REF	3-00000032	I*4	MSLG$L_CNT_ID1
3-00000036	I*4	MSLG$L_CNT_ID2	3-0000003E	I*4	MSLG$L_UNIT_ID1
3-00000042	I*4	MSLG$L_UNIT_ID2	3-0000004A	I*4	MSLG$L_VOL_SER
3-00000030	I*2	MSLG$W_EVENT	3-0000003C	I*2	MSLG$W_MULT_UNT
3-00000048	I*2	MSLG$W_SDE_CYL	3-0000002C	I*2	MSLG$W_SEQ_NUM
3-0000002A	I*2	MSLG$W_UNIT	AP-00000008@	I*4	PACKET_LENGTH

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)

LABELS

Address	Label	Address	Label
1-00000036	10'	1-0000003B	15'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	DUMPREG		LINCHK		MSLG$$B_CNT_HVR
	MSLG$$B_CNT_SVR		MSLG$$B_FLAGS		MSLG$$B_FORMAT
	MSLG$$B_UNIT_HVR		MSLG$$B_UNIT_SVR		MSLG$$L_CMD_REF
	MSLG$$L_VOL_SER		MSLG$$Q_CNT_ID		MSLG$$Q_UNIT_ID
	MSLG$$W_EVENT		MSLG$$W_MULT_UNT		MSLG$$W_SDE_CYL
	MSLG$$W_SEQ_NUM		MSLG$$W_UNIT		MSLG$K_SML_DSK_AZTEC

```
0001
0002
0003
0004            Subroutine MSCP$$K_OP_ABORT (lun,expand)
0005
0006
0007            include 'src$:msghdr.for /nolist'
0066
0067
0068            byte            lun
0069
0070            logical*1       expand
0071
0072            integer*4       mscp$l_cmd_ref
0073
0074            integer*2       mscp$w_unit
0075            integer*2       reserved1
0076
0077            byte            mscp$b_opcode
0078            byte            reserved2
0079
0080            integer*2       mscp$w_modifier
0081
0082            integer*4       mscp$l_out_ref
0083
0084            equivalence     (emb(38),mscp$l_cmd_ref)
0085            equivalence     (emb(42),mscp$w_unit)
0086            equivalence     (emb(44),reserved1)
0087            equivalence     (emb(46),mscp$b_opcode)
0088            equivalence     (emb(47),reserved2)
0089            equivalence     (emb(48),mscp$w_modifier)
0090            equivalence     (emb(50),mscp$l_out_ref)
0091
0092
0093
0094            Entry MSCP$$K_OP_GTCMD (lun,expand)
0095
0096            Entry MSCP$$K_OP_GTUNT (lun,expand)
0097
0098
0099
0100            call linchk (lun,1)
0101
0102            write(lun,5)
0103    5       format(' ',:)
0104
0105            call mscp_first_twelve_bytes1 (lun,expand)
0106
0107
0108
0109            Entry INVALID_MSCP$$K_OP_ABORT (lun,expand)
0110
0111            Entry INVALID_MSCP$$K_OP_GTCMD (lun,expand)
0112
0113            Entry INVALID_MSCP$$K_OP_GTUNT (lun,expand)
0114
0115
```

```
0116
0117            if (mscp$b_opcode .eq. 3) goto 15
0118
0119            call linchk (lun,1)
0120
0121            write(lun,10) mscp$l_out_ref
0122    10      format(' ',t8,'MSCP$L_OUT_REF',t24,z8.8)
0123
0124    15      return
0125
0126            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	177	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	36	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	24	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	749	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-00000057		INVALID_MSCP$$K_OP_ABORT	0-00000062		INVALID_MSCP$$K_OP_GTCMD
0-0000006D		INVALID_MSCP$$K_OP_GTUNT	0-00000000		MSCP$$K_OP_ABORT
0-0000000B		MSCP$$K_OP_GTCMD	0-00000016		MSCP$$K_OP_GTUNT

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000000	I*4	EMB$L_HD_SID	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	AP-00000008@	L*1	EXPAND
AP-0000004@	L*1	LUN	3-0000002E	L*1	MSCP$B_OPCODE
3-00000026	I*4	MSCP$L_CMD_REF	3-00000032	I*4	MSCP$L_OUT_REF
3-00000030	I*2	MSCP$W_MODIFIER	3-0000002A	I*2	MSCP$W_UNIT
3-0000002C	I*2	RESERVED1	3-0000002F	L*1	RESERVED2

ARRAYS

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

LABELS

Address	Label	Address	Label	Address	Label
1-00000004	5'	1-00000009	10'	0-000000B0	15

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	LINCHK		MSCP_FIRST_TWELVE_BYTES1

```
0001
0002
0003
0004          Subroutine MSCP$$K_OP_ACCES (lun,expand)
0005
0006
0007          include 'src$:msghdr.for /nolist'
0066
0067
0068          byte            lun
0069
0070          logical*1       expand
0071
0072          integer*4       mscp$l_cmd_ref
0073
0074          integer*2       mscp$w_unit
0075          integer*2       reserved1
0076
0077          byte            mscp$b_opcode
0078          byte            reserved2
0079
0080          integer*2       mscp$w_modifier
0081
0082          integer*4       mscp$l_byte_cnt
0083          integer*4       mscp$l_rbn
0084          integer*4       reserved3(3)
0085          integer*4       mscp$z_buffer(3)
0086          integer*4       mscp$l_lbn
0087
0088          equivalence     (emb(38),mscp$l_cmd_ref)
0089          equivalence     (emb(42),mscp$w_unit)
0090          equivalence     (emb(44),reserved1)
0091          equivalence     (emb(46),mscp$b_opcode)
0092          equivalence     (emb(47),reserved2)
0093          equivalence     (emb(48),mscp$w_modifier)
0094
0095          equivalence     (emb(50),mscp$l_byte_cnt)
0096          equivalence     (emb(50),mscp$l_rbn)
0097
0098          equivalence     (emb(54),reserved3)
0099          equivalence     (emb(54),mscp$z_buffer)
0100          equivalence     (emb(66),mscp$l_lbn)
0101
0102          integer*4       compress4
0103
0104
0105          Entry MSCP$$K_OP_CMPCD (lun,expand)
0106
0107          Entry MSCP$$K_OP_COMP (lun,expand)
0108
0109          Entry MSCP$$K_OP_ERASE (lun,expand)
0110
0111          Entry MSCP$$K_OP_FLUSH (lun,expand)
0112
0113          Entry MSCP$$K_OP_READ (lun,expand)
0114
0115          Entry MSCP$$K_OP_REPLC (lun,expand)
```

```
0116
0117            Entry MSCP$$K_OP_WRITE (lun,expand)
0118
0119
0120            call linchk (lun,1)
0121
0122            write(lun,5)
0123    5       format(' ',:)
0124
0125            call mscp_first_twelve_bytes1 (lun,expand)
0126
0127
0128
0129            Entry INVALID_MSCP$$K_OP_ACCES (lun,expand)
0130
0131            Entry INVALID_MSCP$$K_OP_CMPCD (lun,expand)
0132
0133            Entry INVALID_MSCP$$K_OP_COMP (lun,expand)
0134
0135            Entry INVALID_MSCP$$K_OP_ERASE (lun,expand)
0136
0137            Entry INVALID_MSCP$$K_OP_FLUSH (lun,expand)
0138
0139            Entry INVALID_MSCP$$K_OP_READ (lun,expand)
0140
0141            Entry INVALID_MSCP$$K_OP_REPLC (lun,expand)
0142
0143            Entry INVALID_MSCP$$K_OP_WRITE (lun,expand)
0144
0145
0146            call linchk (lun,1)
0147
0148            if (mscp$b_opcode .eq. 20) then
0149
0150            write(lun,7) mscp$l_rbn
0151    7       format(' ',t8,'MSCP$L_RBN',t24,z8.8)
0152
0153            if (expand) then
0154
0155            call linchk (lun,1)
0156
0157            write(lun,8) mscp$l_rbn
0158    8       format(' ',t40,'REPLACEMENT BLOCK #',i<compress4 (mscp$l_rbn)>,'.')
0159            endif
0160            else
0161
0162            write(lun,10) mscp$l_byte_cnt
0163    10      format(' ',t8,'MSCP$L_BYTE_CNT',t24,z8.8)
0164
0165            if (expand) then
0166
0167            call linchk (lun,1)
0168
0169            write(lun,11) mscp$l_byte_cnt
0170    11      format(' ',t40,'BYTE COUNT, ',i<compress4 (mscp$l_byte_cnt)>,
0171           1 ', BYTE(S)')
0172            endif
```

```
0173            endif
0174
0175            if (
0176            1 mscp$b_opcode .eq. 32
0177            1 .or.
0178            1 mscp$b_opcode .eq. 33
0179            1 .or.
0180            1 mscp$b_opcode .eq. 34
0181            1 ) then
0182
0183            call mscp$$z_buffer (lun,mscp$z_buffer)
0184            else
0185
0186            call mscp_reservedq (lun,reserved3)
0187
0188            call linchk (lun,4)
0189
0190            write(lun,15) reserved3(3)
0191    15      format(' ',t24,z8.8)
0192            endif
0193
0194            call linchk (lun,1)
0195
0196            write(lun,20) mscp$l_lbn
0197    20      format(' ',t8,'MSCP$[_LBN',t24,z8.8)
0198
0199            if (expand) then
0200
0201            If (LIB$EXTZV(6,2,mscp$b_opcode) .EQ. 0) then
0202
0203            call linchk (lun,1)
0204
0205            write(lun,22) mscp$l_lbn
0206    22      format(' ',t40,'LOGICAL BLOCK #',i<compress4 (mscp$l_lbn)>,'.')
0207            endif
0208            Endif
0209
0210            return
0211            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	670	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	209	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	100	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	1491	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-0000008E		INVALID_MSCP$$K_OP_ACCES	0-00000099		INVALID_MSCP$$K_OP_CMPCD
0-000000A4		INVALID_MSCP$$K_OP_COMP	0-000000AF		INVALID_MSCP$$K_OP_ERASE
0-000000BA		INVALID_MSCP$$K_OP_FLUSH	0-000000C5		INVALID_MSCP$$K_OP_READ
0-000000D0		INVALID_MSCP$$K_OP_REPLC	0-000000DB		INVALID_MSCP$$K_OP_WRITE
0-00000000		MSCP$$K_OP_ACCES	0-0000000B		MSCP$$K_OP_CMPCD
0-00000016		MSCP$$K_OP_COMP	0-00000021		MSCP$$K_OP_ERASE
0-0000002C		MSCP$$K_OP_FLUSH	0-00000037		MSCP$$K_OP_READ
0-00000042		MSCP$$K_OP_REPLC	0-0000004D		MSCP$$K_OP_WRITE

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000000	I*4	EMB$L_HD_SID	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	AP-00000008	L*1	EXPAND
AP-0000004A	L*1	LUN	3-0000002E	L*1	MSCP$B_OPCODE
3-00000032	I*4	MSCP$L_BYTE_CNT	3-00000026	I*4	MSCP$L_CMD_REF
3-00000042	I*4	MSCP$L_LBN	3-00000032	I*4	MSCP$L_RBN
3-00000030	I*2	MSCP$W_MODIFIER	3-0000002A	I*2	MSCP$W_UNIT
3-0000002C	I*2	RESERVED1	3-0000002F	L*1	RESERVED2

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	L*1	EMB	512	(0:511)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)
3-00000036	I*4	MSCP$Z_BUFFER	12	(3)
3-00000036	I*4	RESERVED3	12	(3)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-00000010	5'	1-00000015	7'	1-0000002C	8'	1-00000050	10'	1-0000006C	11'	1-00000091	15'
1-0000009A	20'	1-000000B1	22'								

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK
	MSCP$$Z_BUFFER		MSCP_FIRST_TWELVE_BYTES1		MSCP_RESERVED0

```
0001
0002
0003
0004            Subroutine MSCP$$K_OP_AVAIL (lun,expand)
0005
0006
0007            include 'src$:msghdr.for /nolist'
0066
0067
0068            byte            lun
0069
0070            logical*1       expand
0071
0072            integer*4       mscp$l_cmd_ref
0073
0074            integer*2       mscp$w_unit
0075            integer*2       reserved1
0076
0077            byte            mscp$b_opcode
0078            byte            reserved2
0079
0080            integer*2       mscp$w_modifier
0081
0082            equivalence     (emb(38),mscp$l_cmd_ref)
0083            equivalence     (emb(42),mscp$w_unit)
0084            equivalence     (emb(44),reserved1)
0085            equivalence     (emb(46),mscp$b_opcode)
0086            equivalence     (emb(47),reserved2)
0087            equivalence     (emb(48),mscp$w_modifier)
0088
0089
0090            Entry MSCP$$K_OP_DTACP (lun,expand)
0091
0092            Entry MSCP$$K_OP_SEREX (lun,expand)
0093
0094
0095            call linchk (lun,1)
0096
0097            write(lun,5)
0098        5   format(' ',:)
0099
0100            call mscp_first_twelve_bytes1 (lun,expand)
0101
0102
0103            Entry INVALID_MSCP$$K_OP_AVAIL (lun,expand)
0104
0105            Entry INVALID_MSCP$$K_OP_DTACP (lun,expand)
0106
0107
0108            return
0109            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	108	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	9	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	24	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	653	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-00000057		INVALID_MSCP$$K_OP_AVAIL	0-00000062		INVALID_MSCP$$K_OP_DTACP
0-00000000		MSCP$$K_OP_AVAIL	0-0000000B		MSCP$$K_OP_DTACP
0-00000016		MSCP$$K_OP_SEREX			

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000000	I*4	EMB$L_HD_SID	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	AP-00000008	L*1	EXPAND
AP-0000004A	L*1	LUN	3-0000002E	L*1	MSCP$B_OPCODE
3-00000026	I*4	MSCP$L_CMD_REF	3-00000030	I*2	MSCP$W_MODIFIER
3-0000002A	I*2	MSCP$W_UNIT	3-0000002C	I*2	RESERVED1
3-0000002F	L*1	RESERVED2			

ARRAYS

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

LABELS

Address	Label
1-00000004	5'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	LINCHK		MSCP_FIRST_TWELVE_BYTES1

```
0001
0002
0003
0004            Subroutine MSCP$$K_OP_ONLIN (lun,expand)
0005
0006
0007            include 'src$:msghdr.for /nolist'
0066
0067
0068            byte            lun
0069
0070            logical*1       expand
0071
0072            integer*4       mscp$l_cmd_ref
0073
0074            integer*2       mscp$w_unit
0075            integer*2       reserved1
0076
0077            byte            mscp$b_opcode
0078            byte            reserved2
0079
0080            integer*2       mscp$w_modifier
0081            integer*2       reserved3
0082            integer*2       mscp$w_unt_flgs
0083
0084            integer*4       reserved4(3)
0085            integer*4       mscp$l_dev_parm
0086
0087            integer*2       mscp$w_shdw_unt
0088            integer*2       mscp$w_copy_spd
0089
0090            equivalence     (emb(38),mscp$l_cmd_ref)
0091            equivalence     (emb(42),mscp$w_unit)
0092            equivalence     (emb(44),reserved1)
0093            equivalence     (emb(46),mscp$b_opcode)
0094            equivalence     (emb(47),reserved2)
0095            equivalence     (emb(48),mscp$w_modifier)
0096            equivalence     (emb(50),reserved3)
0097            equivalence     (emb(52),mscp$w_unt_flgs)
0098            equivalence     (emb(54),reserved4)
0099            equivalence     (emb(66),mscp$l_dev_parm)
0100            equivalence     (emb(70),mscp$w_shdw_unt)
0101            equivalence     (emb(72),mscp$w_copy_spd)
0102
0103
0104            Entry MSCP$$K_OP_STUNT (lun,expand)
0105
0106
0107            call linchk (lun,1)
0108
0109            write(lun,5)
0110    5       format(' ',:)
0111
0112            call mscp_first_twelve_bytes1 (lun,expand)
0113
0114
0115            Entry INVALID_MSCP$$K_OP_ONLIN (lun,expand)
```

```
0116
0117            Entry INVALID_MSCP$$K_OP_STUNT (lun,expand)
0118
0119
0120            call mscp$$w_unt_flgs (lun,mscp$w_unt_flgs,expand)
0121
0122            call mscp_reservedq (lun,reserved4)
0123
0124            call linchk (lun,1)
0125
0126            write(lun,10) reserved4(3)
0127    10      format(' ',t24,z8.8)
0128
0129            call mscp$$l_dev_parm (lun,mscp$l_dev_parm)
0130
0131            call mscp$$w_shdw_unt (lun,mscp$w_shdw_unt,expand)
0132
0133            call mscp$$w_copy_spd (lun,mscp$w_copy_spd)
0134
0135            return
0136            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	221	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	18	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	92	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	843	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-0000004C		INVALID_MSCP$$K_OP_ONLIN	0-00000057		INVALID_MSCP$$K_OP_STUNT
0-00000000		MSCP$$K_OP_ONLIN	0-0000000B		MSCP$$K_OP_STUNT

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000000	I*4	EMB$L_HD_SID	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	AP-00000008	L*1	EXPAND
AP-0000004A	L*1	LUN	3-0000002E	L*1	MSCP$B_OPCODE
3-00000026	I*4	MSCP$L_CMD_REF	3-00000042	I*4	MSCP$L_DEV_PARM
3-00000048	I*2	MSCP$W_COPY_SPD	3-00000030	I*2	MSCP$W_MODIFIER
3-00000046	I*2	MSCP$W_SHDW_UNT	3-0000002A	I*2	MSCP$W_UNIT
3-00000034	I*2	MSCP$W_UNT_FLGS	3-0000002C	I*2	RESERVED1
3-0000002F	L*1	RESERVED2	3-00000032	I*2	RESERVED3

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	L*1	EMB	512	(0:511)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)
3-00000036	I*4	RESERVED4	12	(3)

LABELS

Address	Label	Address	Label
1-00000004	5'	1-00000009	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	LINCHK		MSCP$$L_DEV_PARM		MSCP$$W_COPY_SPD
	MSCP$$W_SHDW_UNT		MSCP$$W_UNT_FLGS		MSCP_FIRST_TWELVE_BYTES1
	MSCP_RESERVED0				

```
0001
0002
0003
0004          Subroutine MSCP$$K_OP_STCON (lun,expand)
0005
0006
0007          include 'src$:msghdr.for /nolist'
0066
0067
0068          byte              lun
0069
0070          logical*1         expand
0071
0072          integer*4         mscp$l_cmd_ref
0073
0074          integer*2         mscp$w_unit
0075          integer*2         reserved1
0076
0077          byte              mscp$b_opcode
0078          byte              reserved2
0079
0080          integer*2         mscp$w_modifier
0081          integer*2         mscp$w_version
0082          integer*2         mscp$w_cnt_flgs
0083          integer*2         mscp$w_hst_tmo
0084          integer*2         reserved3
0085
0086          integer*4         mscp$l_time1
0087          integer*4         mscp$l_time2
0088
0089          equivalence       (emb(38),mscp$l_cmd_ref)
0090          equivalence       (emb(42),mscp$w_unit)
0091          equivalence       (emb(44),reserved1)
0092          equivalence       (emb(46),mscp$b_opcode)
0093          equivalence       (emb(47),reserved2)
0094          equivalence       (emb(48),mscp$w_modifier)
0095          equivalence       (emb(50),mscp$w_version)
0096          equivalence       (emb(52),mscp$w_cnt_flgs)
0097          equivalence       (emb(54),mscp$w_hst_tmo)
0098          equivalence       (emb(56),reserved3)
0099          equivalence       (emb(58),mscp$l_time1)
0100          equivalence       (emb(60),mscp$l_time2)
0101
0102
0103          call linchk (lun,1)
0104
0105          write(lun,5)
0106    5     format(' ',:)
0107
0108          call mscp_first_twelve_bytes1 (lun,expand)
0109
0110
0111          Entry INVALID_MSCP$$K_OP_STCON (lun,expand)
0112
0113
0114          call mscp$$w_version (lun,mscp$w_version)
0115
```

```
0116            call mscp$$w_cnt_flgs (lun,mscp$w_cnt_flgs,expand)
0117
0118            call mscp$$w_hst_tmo (lun,mscp$w_hst_tmo,expand)
0119
0120            call mscp_reservedw (lun,reserved3)
0121
0122            call mscp$$q_time (lun,mscp$l_time1,expand)
0123
0124            return
0125            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	155	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	9	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	96	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	772	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-00000041		INVALID_MSCP$$K_OP_STCON	0-00000000		MSCP$$K_OP_STCON

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000000	I*4	EMB$L_HD_SID	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	AP-00000008@	L*1	EXPAND
AP-00000004@	L*1	LUN	3-0000002E	L*1	MSCP$B_OPCODE
3-00000026	I*4	MSCP$L_CMD_REF	3-0000003A	I*4	MSCP$L_TIME1
3-0000003C	I*4	MSCP$L_TIME2	3-00000034	I*2	MSCP$W_CNT_FLGS
3-00000036	I*2	MSCP$W_HST_TMO	3-00000030	I*2	MSCP$W_MODIFIER
3-0000002A	I*2	MSCP$W_UNIT	3-00000032	I*2	MSCP$W_VERSION
3-0000002C	I*2	RESERVED1	3-0000002F	L*1	RESERVED2
3-00000038	I*2	RESERVED3			

ARRAYS

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

LABELS

Address	Label
1-00000004	5'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	LINCHK		MSCP$$Q_TIME		MSCP$$W_CNT_FLGS
	MSCP$$W_HST_TMO		MSCP$$W_VERSION		MSCP_FIRST_TWELVE_BYTES1
	MSCP_RESERVEDW				

```
0001
0002
0003
0004            Subroutine MSLG$$L_CMD_REF (lun,command_reference_number)
0005
0006
0007            byte            lun
0008
0009            integer*4       command_reference_number
0010
0011
0012            call linchk (lun,1)
0013
0014            write(lun,10) command_reference_number
0015    10      format(' ',t8,'MSLG$L_CMD_REF',t24,z8.8)
0016
0017            return
0018
0019
0020
0021            Entry MSCP$$L_CMD_REF (lun,command_reference_number)
0022
0023
0024            call linchk (lun,1)
0025
0026            write(lun,15) command_reference_number
0027    15      format(' ',t8,'MSCP$L_CMD_REF',t24,z8.8)
0028
0029            return
0030            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	112	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	58	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	12	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	182	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-00000038		MSCP$$L_CMD_REF	0-00000000		MSLG$$L_CMD_REF

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008@	I*4	COMMAND_REFERENCE_NUMBER	AP-00000004@	L*1	LUN

LABELS

Address	Label	Address	Label
1-00000004	10'	1-0000001F	15'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
	LINCHK

```
      Subroutine MSLG$$W_UNIT (lun,unit_number,expand)

      byte            lun

      integer*2       unit_number

      logical*1       expand

      integer*4       unit
      integer*4       compress4

      call linchk (lun,2)

      write(lun,10) 'MSLG$W_UNIT',unit_number
10    format(' ',t8,a,t28,z4.4)

15    if (expand) then

      unit = lib$extzv(0,16,unit_number)

      write(lun,20) unit
20    format(' ',t40,'UNIT #',i<compress4 (unit)>,'.')
      endif

      return


      Entry MSCP$$W_UNIT (lun,unit_number,expand)

      call linchk (lun,2)

      write(lun,10) 'MSCP$W_UNIT',unit_number

      goto 15

      end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	203	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	69	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	56	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	328	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-00000079		MSCP$$W_UNIT	0-00000000		MSLG$$W_UNIT

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name	Address	Type	Name
AP-0000000C@	L*1	EXPAND	AP-00000004@	L*1	LUN	2-00000000	I*4	UNIT	AP-00000008@	I*2	UNIT_NUMBER

LABELS

Address	Label	Address	Label	Address	Label
1-00000022	10'	0-00000042	15	1-0000002E	20'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK

```
0001
0002
0003
0004            Subroutine MSLG$$W_SEQ_NUM (lun,sequence_number)
0005
0006
0007            byte            lun
0008
0009            integer*2       sequence_number
0010
0011            integer*4       sequence
0012            integer*4       compress4
0013
0014
0015            call linchk (lun,2)
0016
0017            write(lun,10) sequence_number
0018    10      format(' ',t8,'MSLG$W_SEQ_NUM',t28,z4.4)
0019
0020            sequence = lib$extzv(0,16,sequence_number)
0021
0022            write(lun,20) sequence
0023    20      format(' ',t40,'SEQUENCE #',i<compress4 (sequence)>,'.')
0024
0025            return
0026            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	120	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	66	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	40	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	226	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$$W_SEQ_NUM

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000004@	L*1	LUN	2-00000000	I*4	SEQUENCE
AP-00000008@	I*2	SEQUENCE_NUMBER			

LABELS

Address	Label	Address	Label
1-0000000C	10'	1-00000027	20'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK

```
0001
0002
0003
0004            Subroutine MSLG$$B_FORMAT (lun,format)
0005
0006            byte            lun
0007            byte            format
0008
0009            integer*4       message_format
0010            integer*4       compress4
0011
0012
0013            call linchk (lun,2)
0014            write(lun,10) format
0015    10      format(' ',t8,'MSLG$B_FORMAT',t30,z2.2)
0016
0017            message_format = lib$extzv(0,8,format)
0018
0019            if (message_format .eq. 0) then
0020            write(lun,20) 'CONTROLLER ERROR'
0021    20      format(' ',t40,a)
0022
0023            else if (message_format .eq. 1) then
0024            write(lun,20) 'HOST MEMORY ACCESS ERROR'
0025
0026            else if (message_format .eq. 2) then
0027            write(lun,20) 'DISK TRANSFER ERROR'
0028
0029            else if (message_format .eq. 3) then
0030            write(lun,20) '"SDI" ERROR'
0031
0032            else if (message_format .eq. 4) then
0033            write(lun,20) '"SMALL DISK" ERROR'
0034
0035            Else if (message_format .EQ. 5) then
0036            Write (lun,20) 'TAPE TRANSFER ERROR'
0037
0038            Else if (message_format .EQ. 6) then
0039            Write (lun,20) '"STI" ERROR'
0040
0041            Else if (message_format .EQ. 7) then
0042            Write (lun,20) '"STI" DRIVE ERROR'
0043
0044            Else if (message_format .EQ. 8) then
0045            Write (lun,20) '"STI" FORMATTER ERROR'
0046
0047            else
0048    C
0049    C Unknown message format, output it decimal.
0050    C
0051            write(lun,25) message_format
0052    25      format(' ',t40,'FORMAT #',i<compress4 (message_format)>,'.')
0053            endif
0054
0055            return
0056            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	492	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	226	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	112	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	830	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$$B_FORMAT

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008@	L*1	FORMAT	AP-00000004@	L*1	LUN
2-00000000	I*4	MESSAGE_FORMAT			

LABELS

Address	Label	Address	Label	Address	Label
1-000000A8	10'	1-000000C2	20'	1-000000C9	25'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK

```
0001
0002
0003
0004          Subroutine MSLG$$B_FLAGS (lun,error_log_message_flags)
0005
0006
0007          byte              lun
0008          byte              error_log_message_flags
0009
0010          character*22      v1error_log_message_flags(0:0)
0011          data              v1error_log_message_flags(0) /'SEQUENCE NUMBER RESET*'/
0012
0013          character*21      v2error_log_message_flags(6:7)
0014          data              v2error_log_message_flags(6) /'OPERATION CONTINUING*'/
0015          data              v2error_log_message_flags(7) /'OPERATION SUCCESSFUL*'/
0016
0017
0018          call linchk (lun,1)
0019
0020          write(lun,10) error_log_message_flags
0021    10    format(' ',t8,'MSLG$B_FLAGS',t30,z2.2)
0022
0023          call output (lun,lib$extzv(0,8,error_log_message_flags),
0024          1 v1error_log_message_flags,0,0,'0')
0025
0026          call output (lun,lib$extzv(0,8,error_log_message_flags),
0027          1 v2error_log_message_flags,6,6,'0')
0028
0029          return
0030          end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	127	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	47	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	228	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	402	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$$B_FLAGS

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008@	L*1	ERROR_LOG_MESSAGE_FLAGS	AP-00000004@	L*1	LUN

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	CHAR	V1ERROR_LOG_MESSAGE_FLAGS	22	(0:0)
2-00000016	CHAR	V2ERROR_LOG_MESSAGE_FLAGS	42	(6:7)

LABELS

Address	Label
1-00000016	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	LIB$EXTZV		LINCHK		OUTPUT

```
0001
0002
0003
0004            Subroutine MSCP$$B_FLAGS (lun,end_message_flags,expand)
0005
0006
0007            byte            lun
0008            byte            end_message_flags
0009
0010            logical*1       expand
0011
0012            character*21    vlend_message_flags(4:7)
0013            data            vlend_message_flags(4)  /'SERIOUS EXCEPTION*'/
0014            data            vlend_message_flags(5)  /'ERROR LOG GENERATED*'/
0015            data            vlend_message_flags(6)  /'BAD BLOCK UNREPORTED*'/
0016            data            vlend_message_flags(7)  /'BAD BLOCK REPORTED*'/
0017
0018
0019            call linchk (lun,1)
0020
0021            write(lun,10) end_message_flags
0022    10      format(' ',t8,'MSCP$B_FLAGS',t30,z2.2)
0023
0024            if (expand) then
0025
0026            call output (lun,end_message_flags,vlend_message_flags,4,4,7,'0')
0027            endif
0028
0029            return
0030            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	80	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	39	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	168	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	287	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP$$B_FLAGS

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008@	L*1	END_MESSAGE_FLAGS	AP-0000000C@	L*1	EXPAND
AP-00000004@	L*1	LUN			

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	CHAR	V1END_MESSAGE_FLAGS	84	(4:7)

LABELS

Address	Label
1-0000000E	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	LINCHK		OUTPUT

005
005
006
006
006
006
006
006
006
006
006
006
007
007
007
007
007
007
007
007
007
008
008
008
008
008
008
008
008
008
008
009
009
009
009
009
009
009
009
009
009
010
010
010
010
010
010
010
010
010
010
011
011
011
011
011
011

```
0001
0002
0003
0004            Subroutine MSLG$$W_EVENT (lun,event_code)
0005
0006
0007            byte            lun
0008
0009            integer*2       event_code
0010
0011            integer*4       code
0012            integer*4       sub_code
0013            integer*4       compress4
0014
0015            logical*1       valid_subcode
0016
0017
0018            call linchk (lun,1)
0019
0020            write(lun,10) event_code
0021    10      format(' ',t8,'MSLG$W_EVENT',t28,z4.4)
0022
0023
0024            Entry AZTEC_FULL_ERROR_CODE1 (lun,event_code)
0025
0026            Call STATUS_AND_EVENT_CODES (lun)
0027
0028            Return
0029            End
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	81	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	29	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	36	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	146	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-0000003A		AZTEC_FULL_ERROR_CODE1	0-00000000		MSLG$$W_EVENT

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name	Address	Type	Name
2-00000004	I*4	CODE	2-0000000C	I*4	COMPRESS4	AP-00000008@	I*2	EVENT_CODE	AP-00000004@	L*1	LUN
2-00000008	I*4	SUB_CODE	2-00000000	L*1	VALID_SUBCODE						

LABELS

Address	Label
1-00000004	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	LINCHK		STATUS_AND_EVENT_CODES

```
0001
0002
0003
0004          Subroutine MSLG$$Q_CNT_ID (lun,controller_id1,controller_id2)
0005
0006
0007          byte              lun
0008
0009          integer*4         controller_id1
0010          integer*4         controller_id2
0011          integer*4         unique_identifier_high
0012          integer*4         class
0013          integer*4         model
0014
0015          character*24      subsystem_class(3)
0016          data              subsystem_class(1)      /'MASS STORAGE CONTROLLER*'/
0017          data              subsystem_class(2)      /'DISK CLASS DEVICE*'/
0018          data              subsystem_class(3)      /'TAPE CLASS DEVICE*'/
0019
0020          character*27      class_model(11)
0021          data              class_model(1)          /'HSC50*'/
0022          data              class_model(2)          /'UDA50*'/
0023          data              class_model(3)          /'RC25*'/
0024          data              class_model(4)          /'VMS (SOFTWARE MSCP SERVER*'/
0025          data              class_model(5)          /'TU81P*'/
0026          data              class_model(6)          /'UDA50A*'/
0027          data              class_model(7)          /'RD51*'/
0028          data              class_model(8)          /'TOPS 10/20 MSCP SERVER*'/
0029          data              class_model(9)          /'TK50*'/
0030          data              class_model(10)         /'RUX50*'/
0031          data              class_model(11)         /'RC26*'/
0032
0033          integer*4         compressc
0034
0035
0036
0037          call linchk (lun,2)
0038
0039          write(lun,10) controller_id1,controller_id2
0040    10    format(' ',t8,'MSLG$Q_CNT_ID',t24,z8.8,7,t24,z8.8)
0041
0042
0043
0044          Entry CDDB$Q_CNTRLID (lun,controller_id1,controller_id2)
0045
0046
0047          unique_identifier_high = lib$extzv (0,16,controller_id2)
0048
0049          call linchk (lun,1)
0050
0051          write(lun,15) unique_identifier_high,controller_id1
0052    15    format(' ',t40,'UNIQUE IDENTIFIER, ',z4.4,z8.8)
0053
0054          class = lib$extzv(24,8,controller_id2)
0055          model = lib$extzv(16,8,controller_id2)
0056
0057          if (
```

```
0058            1 class .gt. 0
0059            1 .and.
0060            1 class .le. 3
0061            1 ) then
0062
0063            call linchk (lun,1)
0064
0065            write(lun,20) subsystem_class(class)
0066    20      format(' ',t40,a<compressc (subsystem_class(class))>)
0067            endif
0068
0069            if (
0070            1 model .gt. 0
0071            1 .and.
0072            1 model .le. 7
0073            1 ) then
0074
0075            call linchk (lun,1)
0076
0077            write(lun,25) class_model(model)
0078    25      format(' ',t40,a<compressc (class_model(model))>)
0079            endif
0080
0081            return
0082            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	400	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	113	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	496	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	1009	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-00000044		CDDB$Q_CNTRLID	0-00000000		MSLG$$Q_CNT_ID

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000178	I*4	CLASS	AP-00000008@	I*4	CONTROLLER_ID1
AP-0000000C@	I*4	CONTROLLER_ID2	AP-00000004@	L*1	LUN
2-0000017C	I*4	MODEL	2-00000174	I*4	UNIQUE_IDENTIFIER_HIGH

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000048	CHAR	CLASS_MODEL	297	(11)
2-00000000	CHAR	SUBSYSTEM_CLASS	72	(3)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label
1-00000018	10'	1-00000038	15'	1-00000059	20'	1-00000065	25'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESSC	I*4	LIB$EXTZV		LINCHK

```
0001
0002
0003            Subroutine MSLG$$B_CNT_SVR (lun,controller_software_version)
0004
0005
0006            byte            lun
0007            byte            controller_software_version
0008
0009            integer*4       software_version
0010            integer*4       compress4
0011
0012
0013            software_version = lib$extzv(0,8,controller_software_version)
0014
0015            call linchk (lun,2)
0016
0017            write(lun,10) controller_software_version,software_version
0018    10      format(' ',t8,'MSLG$$B_CNT_SVR',t30,z2.2,/,
0019           1 t40,'CONTROLLER SOFTWARE VERSION #',
0020           1 i<compress4 (software_version)>,'.')
0021
0022            return
0023            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	96	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	82	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	40	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	218	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$$B_CNT_SVR

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008@	L*1	CONTROLLER_SOFTWARE_VERSION	AP-00000004@	L*1	LUN
2-00000000	I*4	SOFTWARE_VERSION			

LABELS

Address	Label
1-0000000C	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK

```
0001
0002
0003
0004            Subroutine MSLG$$B_CNT_HVR (lun,controller_hardware_version)
0005
0006
0007            byte            lun
0008            byte            controller_hardware_version
0009
0010            integer*4       hardware_version
0011            integer*4       compress4
0012
0013
0014            hardware_version = lib$extzv(0,8,controller_hardware_version)
0015
0016            call linchk (lun,2)
0017
0018            write(lun,10) controller_hardware_version,hardware_version
0019    10      format(' ',t8,'MSLG$B_CNT_HVR',t30,z2.2,/,
0020           1 t40,'CONTROLLER HARDWARE REVISION #',
0021           1 i<compress4 (hardware_version)>,'.')
0022
0023            return
0024            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	96	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	83	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	40	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	219	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$$B_CNT_HVR

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008@	L*1	CONTROLLER_HARDWARE_VERSION	2-00000000	I*4	HARDWARE_VERSION
AP-00000004@	L*1	LUN			

LABELS

Address	Label
1-0000000C	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK

0001
0002
0003
0004
0005
0006
0007
0008
0009
0010
0011
0012
0013
0014
0015
0016
0017
0018

PROC

0
1
2

ENTR

0-

VARI

2-

LABE

1-

```
0001
0002
0003
0004            Subroutine MSLG$$W_MULT_UNT (lun,multi_unit_code)
0005
0006
0007            byte              lun
0008
0009            integer*2         multi_unit_code
0010
0011
0012            call linchk (lun,1)
0013
0014            write(lun,10) 'MSLG$W_MULT_UNT',multi_unit_code
0015    10      format(' ',t8,a,t28,z4.4)
0016
0017            return
0018
0019
0020
0021            Entry MSCP$$W_MULT_UNT (lun,multi_unit_code)
0022
0023
0024            call linchk (lun,1)
0025
0026            write(lun,10) 'MSCP$W_MULT_UNT',multi_unit_code
0027
0028            return
0029            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	132	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	46	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	28	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	206	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-00000042		MSCP$$W_MULT_UNT	0-00000000		MSLG$$W_MULT_UNT

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000004@	L*1	LUN	AP-00000008@	I*2	MULTI_UNIT_CODE

LABELS

Address	Label
1-00000022	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
	LINCHK

```
0001
0002
0003
0004            Subroutine MSLG$$Q_UNIT_ID (lun,unit_id1,unit_id2,expand)
0005
0006
0007            byte            lun
0008
0009            integer*4       unit_id1
0010            integer*4       unit_id2
0011
0012            logical*1       expand
0013
0014            integer*4       class
0015            integer*4       model
0016            integer*4       unique_identifier_high
0017            integer*4       compressc
0018
0019            character*24    subsystem_class(4)
0020            data            subsystem_class(1)      /'MASS STORAGE CONTROLLER*'/
0021            data            subsystem_class(2)      /'DISK CLASS DEVICE*'/
0022            data            subsystem_class(3)      /'TAPE CLASS DEVICE*'/
0023            data            subsystem_class(4)      /'DISK CLASS DEVICE*'/
0024
0025
0026            call linchk (lun,2)
0027
0028            write(lun,5) 'MSLG$Q_UNIT_ID',unit_id1,unit_id2
0029    5       format(' ',t8,a,t24,z8.8,7,t24,z8.8)
0030
0031    10      continue
0032
0033            if (expand) then
0034
0035            unique_identifier_high = lib$extzv (0,16,unit_id2)
0036
0037            call linchk (lun,1)
0038
0039            write(lun,15) unique_identifier_high,unit_id1
0040    15      format(' ',t40,'UNIQUE IDENTIFIER, ',z4.4,z8.8)
0041
0042            class = lib$extzv(24,8,unit_id2)
0043            model = lib$extzv(16,8,unit_id2)
0044
0045            if (
0046            1 class .gt. 0
0047            1 .and.
0048            1 class .le. 4
0049            1 ) then
0050
0051            call linchk (lun,1)
0052
0053            write(lun,20) subsystem_class(class)
0054    20      format(' ',t40,a<compressc (subsystem_class(class))>)
0055            endif
0056
0057            if (class .eq. 2) then
```

```
0058
0059            if (model .eq. 1) then
0060
0061            call linchk (lun,1)
0062            write(lun,25) 'RA80'
0063    25      format(' ',t40,a)
0064
0065            else if (model .eq. 2) then
0066
0067            call linchk (lun,1)
0068            write(lun,25) 'RC25'
0069
0070            else if (model .eq. 3) then
0071
0072            call linchk (lun,1)
0073            write(lun,25) 'RCF25'
0074
0075            else if (model .eq. 4) then
0076
0077            call linchk (lun,1)
0078            write(lun,25) 'RA60'
0079
0080            else if (model .eq. 5) then
0081
0082            call linchk (lun,1)
0083            write(lun,25) 'RA81'
0084
0085            else if (model .eq. 6) then
0086
0087            call linchk (lun,1)
0088            write(lun,25) 'RD51'
0089
0090            else if (model .eq. 7) then
0091
0092            call linchk (lun,1)
0093            write(lun,25) 'RX50'
0094
0095            else if (model .eq. 8) then
0096
0097            call linchk (lun,1)
0098            write(lun,25) 'RD52'
0099
0100            else if (model .eq. 9) then
0101
0102            call linchk (lun,1)
0103            write(lun,25) 'RD53'
0104
0105            else if (model .eq. 10) then
0106
0107            call linchk (lun,1)
0108            write(lun,25) 'RD26'
0109
0110            else if (model .eq. 11) then
0111
0112            call linchk (lun,1)
0113            write(lun,25) 'RA82'
0114
```

```
0115            else if (model .eq. 12) then
0116
0117            call linchk (lun,1)
0118            write(lun,25) 'RC26'
0119
0120            else if (model .eq. 13) then
0121
0122            call linchk (lun,1)
0123            write(lun,25) 'RCF26'
0124            endif
0125
0126            else if (class .eq. 3) then
0127
0128            if (model .eq. 1) then
0129
0130            call linchk (lun,1)
0131            write(lun,25) 'TU78'
0132
0133            else if (model .eq. 2) then
0134
0135            call linchk (lun,1)
0136            write(lun,25) 'TU81'
0137            endif
0138            endif
0139            endif
0140
0141            return
0142
0143
0144            Entry MSCP$$Q_UNIT_ID (lun,unit_id1,unit_id2,expand)
0145
0146
0147            call linchk (lun,2)
0148
0149            write(lun,5) 'MSCP$Q_UNIT_ID',unit_id1,unit_id2
0150
0151            goto 10
0152
0153            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	1223	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	184	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	340	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	1747	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-00000455		MSCP$$Q_UNIT_ID	0-00000000		MSLG$$Q_UNIT_ID

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000060	I*4	CLASS	AP-00000010@	L*1	EXPAND
AP-00000004@	L*1	LUN	2-00000064	I*4	MODEL
2-00000068	I*4	UNIQUE_IDENTIFIFR_HIGH	AP-00000008@	I*4	UNIT_ID1
AP-0000000C@	I*4	UNIT_ID2			

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	CHAR	SUBSYSTEM_CLASS	96	(4)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-00000072	5'	0-0000004C	10	1-00000084	15'	1-000000A5	20'	1-000000B1	25'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESSC	I*4	LIB$EXTZV		LINCHK

```
0001
0002
0003
0004            Subroutine MSLG$$B_UNIT_SVR (lun,unit_software_version)
0005
0006
0007            byte            lun
0008            byte            unit_software_version
0009
0010            integer*4       software_version
0011            integer*4       compress4
0012
0013
0014            software_version = lib$extzv(0,8,unit_software_version)
0015
0016            call linchk (lun,2)
0017
0018            write(lun,10) unit_software_version,software_version
0019    10      format(' ',t8,'MSLG$B_UNIT_SVR',t30,z2.2,/,
0020           1 t40,'UNIT SOFTWARE VERSION #',
0021           1 i<compress4 (software_version)>,'.')
0022
0023            return
0024            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	96	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	77	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	40	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	213	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$$B_UNIT_SVR

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000004@	L*1	LUN	2-00000000	I*4	SOFTWARE_VERSION
AP-00000008@	L*1	UNIT_SOFTWARE_VERSION			

LABELS

Address	Label
1-0000000C	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK

000
000
000
000
000
000
000
000
000
001
001
001
001
001
001
001
001
001

PRO

0
1
2

ENT

0

VAR

AP

LAB

1

```
0001
0002
0003            Subroutine MSLG$$B_UNIT_HVR (lun,unit_hardware_version)
0004
0005
0006            byte            lun
0007            byte            unit_hardware_version
0008
0009            integer*4       hardware_version
0010            integer*4       compress4
0011
0012
0013            hardware_version = lib$extzv(0,8,unit_hardware_version)
0014
0015            call linchk (lun,2)
0016
0017            write(lun,10) unit_hardware_version,hardware_version
0018    10      format(' ',t8,'MSLG$B_UNIT_HVR',t30,z2.2,/,
0019           1 t40,'UNIT HARDWARE REVISION #',i<compress4 (hardware_version)>,'.')
0020
0021            return
0022            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	96	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	78	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	40	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	214	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$$B_UNIT_HVR

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000000	I*4	HARDWARE_VERSION	AP-00000004@	L*1	LUN
AP-00000008@	L*1	UNIT_HARDWARE_VERSION			

LABELS

Address	Label
1-0000000C	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK

0000
0000
0000
0000
0000
0000
0000
0006
0013
0013
0013
0013
0013
0013
0014
0014
0014
0014
0014
0014
0014
0014
0014
0014
0014
0014
0015
0015
0015
0015
0015
0015
0015
0015
0015
0016
0016
0016
0016
0016
0016
0016
0016
0016
0016
0016
0017
0017
0017
0017
0017
0017
0017
0017
0017
0017
0017
0018
0018
0018
0018

```
0001
0002
0003
0004          Subroutine MSLG$$L_VOL_SER (lun,volume_serial_number)
0005
0006          byte              lun
0007
0008          External          Sys$fao
0009
0010          Logical*4         Sys$fao
0011          Logical*1         Status
0012
0013          Character*10      cnvrted_serial_number
0014
0015          integer*4         volume_serial_number
0016          integer*4         compress4
0017
0018
0019          call linchk (lun,2)
0020
0021          Status = SYS$FAO( '!ZL',,cnvrted_serial_number,
0022          1                       %VAL(volume_serial_number) )
0023
0024          If (.NOT. status) then
0025          Call LIB$SIGNAL (status)
0026          Endif
0027
0028          write(lun,10) volume_serial_number,cnvrted_serial_number
0029    10    format(' ',t8,'MSLG$L_VOL_SER',t24,z8.8,/,
0030          1 t40,'VOLUME SERIAL #',A10,'.')
0031
0032          return
0033          end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	95	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	60	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	72	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	227	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$$L_VOL_SER

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000001	CHAR	CNVRTED_SERIAL_NUMBER	2-0000000C	I*4	COMPRESS4
AP-00000004@	L*1	LUN	2-00000000	L*1	STATUS
AP-00000008@	I*4	VOLUME_SERIAL_NUMBER			

LABELS

Address	Label
1-00000008	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	LIB$SIGNAL		LINCHK	L*4	SYS$FAO

```
0001
0002
0003
0004            Subroutine MSLG$$L_GAP_CNT (lun,gap_cnt)
0005
0006            Byte            lun
0007
0008            Integer*4       gap_cnt, COMPRESS4, count
0009
0010            Count = LIB$EXTZV (0,16,gap_cnt)
0011
0012            Call LINCHK (lun,1)
0013            Write (lun,10) 'MSLG$L_GAP_CNT',gap_cnt,count
0014    10      Format (' ',T8,A,T24,Z8.8,7,
0015            1 T40,'GAP COUNT = ',I<COMPRESS4 (count)>,'.')
0016
0017            Return
0018            End
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	106	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	64	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	48	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	218	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$$L_GAP_CNT

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name
2-00000000	I*4	COUNT	AP-00000008@	I*4	GAP_CNT	AP-00000004@	L*1	LUN

LABELS

Address	Label
1-0000001A	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK

```
0001
0002
0003            Subroutine FORMATTER_VERSIONS (lun,s_vers,h_vers)
0004
0005            Byte            lun, h_vers, s_vers
0006
0007            Integer*4       COMPRESS4, version
0008
0009
0010            Version = LIB$EXTZV (0,8,s_vers)
0011
0012            Call LINCHK (lun,4)
0013
0014            Write (lun,10) 'MSLG$B_FMTR_SVR',s_vers,version
0015    10      Format (' ',T8,A,T30,Z2.2,/,T40,
0016           1 'FORMATTER SOFTWARE VERSION #',I<COMPRESS4 (version)>,'.')
0017
0018            Version = LIB$EXTZV (0,8,h_vers)
0019
0020            Write (lun,15) 'MSLG$B_FMTR_HVR',h_vers,version
0021    15      Format (' ',T8,A,T30,Z2.2,/,T40,
0022           1 'FORMATTER HARDWARE VERSION #',I<COMPRESS4 (version)>,'.')
0023
0024            Return
0025            End
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	189	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	150	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	56	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	395	

ENTRY POINTS

Address	Type	Name
0-00000000		FORMATTER_VERSIONS

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name	Address	Type	Name
AP-0000000C@	L*1	H_VERS	AP-00000004@	L*1	LUN	AP-00000008@	L*1	S_VERS	2-00000000	I*4	VERSION

LABELS

Address	Label	Address	Label
1-0000002A	10'	1-00000060	15'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK

```
0001
0002
0003
0004            Subroutine MSLG$$L_BUS_ADDR (lun,bus_address)
0005
0006
0007            byte            lun
0008
0009            integer*4       bus_address
0010
0011
0012            call linchk (lun,1)
0013
0014            write(lun,10) bus_address
0015    10      format(' ',t8,'MSLG$L_BUS_ADDR',t24,z8.8)
0016
0017            call calc_map (lun,16,bus_address,bus_address)
0018
0019            return
0020            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	79	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	36	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	32	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	147	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$$L_BUS_ADDR

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008@	I*4	BUS_ADDRESS	AP-00000004@	L*1	LUN

LABELS

Address	Label
1-00000008	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	CALC_MAP		LINCHK

```
0001
0002
0003
0004            Subroutine MSLG$$B_LEVEL (lun,level)
0005
0006
0007            byte            lun
0008            byte            level
0009
0010
0011            call linchk (lun,1)
0012
0013            write(lun,10) level
0014    10      format(' ',t8,'MSLG$B_LEVEL',t30,z2.2)
0015
0016            return
0017            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	56	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	29	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	12	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	97	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$$B_LEVEL

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008@	L*1	LEVEL	AP-00000004@	L*1	LUN

LABELS

Address	Label
1-00000004	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
	LINCHK

```
            Subroutine MSLG$$B_RETRY (lun,retry)

            byte            lun
            byte            retry

            call linchk (lun,1)

            write(lun,10) retry
      10    format(' ',t8,'MSLG$B_RETRY',t30,z2.2)

            return
            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	56	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	29	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	12	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	97	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$$B_RETRY

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000004@	L*1	LUN	AP-00000008@	L*1	RETRY

LABELS

Address	Label
1-00000004	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
	LINCHK

```
0001
0002
0003
0004          Subroutine MSLG$$Z_SDI (lun,sdi1,sdi2,sdi3)
0005
0006
0007          include 'src$:msghdr.for /nolist'
0066          include 'src$:emblmdef.for /nolist'
0135
0136
0137          byte              lun
0138
0139          integer*4         sdi1
0140          integer*4         sdi2
0141          integer*4         sdi3
0142          integer*4         c_flags
0143          integer*4         retry_count
0144
0145          logical*1         init_diagnostic_failure
0146
0147          integer*4         previous_command
0148          integer*4         cylinder_low
0149          integer*4         cylinder_high
0150          integer*4         cylinder
0151          integer*4         current_group
0152          integer*4         led_display_code
0153
0154
0155          call linchk (lun,3)
0156
0157          write(lun,10) sdi1,sdi2,sdi3
0158    10    format(' ',t8,'MSLG$Z_SDI',t24,z8.8,/,t24,z8.8,/,t24,z8.8)
0159
0160          call generic_sdi (lun,sdi1,sdi2,sdi3)
0161
0162          If (LIB$EXTZV(3,1,sdi1) .EQ. 1) then
0163
0164          if (
0165          1 emb$b_lm_type .eq. 20
0166          1 .or.
0167          1 emb$b_lm_type .eq. 21
0168          1 ) then
0169
0170          call ra8x_status (lun,sdi1,sdi2,sdi3)
0171          endif
0172
0173          if (emb$b_lm_type .eq. 20) then
0174
0175          call ra80_status (lun,sdi1,sdi2,sdi3)
0176
0177          else if (emb$b_lm_type .eq. 21) then
0178
0179          call ra81_status (lun,sdi1,sdi2,sdi3)
0180
0181          else if (emb$b_lm_type .eq. 22) then
0182
0183          call ra60_status (lun,sdi1,sdi2,sdi3)
```

```
0184            endif
0185            endif
0186
0187            return
0188            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	278	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	43	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	84	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	917	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$$Z_SDI

VARIABLES

Address	Type	Name	Address	Type	Name
2-0000001C	I*4	CURRENT_GROUP	2-00000018	I*4	CYLINDER
2-00000014	I*4	CYLINDER_HIGH	2-00000010	I*4	CYLINDER_LOW
2-00000004	I*4	C_FLAGS	3-00000010	L*1	EMB$B_LM_CLASS
3-00000014	L*1	EMB$B_LM_NAMLNG	3-00000011	L*1	EMB$B_LM_TYPE
3-00000000	I*4	EMB$L_HD_SID	3-00000015	CHAR	EMB$T_LM_NAME
3-00000004	I*2	EMB$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
2-00000000	L*1	INIT_DIAGNOSTIC_FAILURE	2-00000020	I*4	LED_DISPLAY_CODE
AP-0000004@	L*1	LUN	2-0000000C	I*4	PREVIOUS_COMMAND
2-00000008	I*4	RETRY_COUNT	AP-0000008@	I*4	SDI1
AP-0000000C@	I*4	SDI2	AP-0000010@	I*4	SDI3

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)

LABELS

Address	Label
1-00000008	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name	Type	Name	Type	Name
	GENERIC_SDI	I*4	LIB$EXTZV		LINCHK		RA60_STATUS		RA80_STATUS		RA81_STATUS
	RA8X_STATUS										

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

```
0001
0002
0003
0004            Subroutine MSLG$$W_SDE_CYL (lun,small_disk_cylinder)
0005
0006
0007            byte            lun
0008
0009            integer*2       small_disk_cylinder
0010
0011
0012            call linchk (lun,1)
0013
0014            write(lun,10) small_disk_cylinder
0015    10      format(' ',t8,'MSLG$W_SDE_CYL',t28,z4.4)
0016
0017            return
0018            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	56	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	31	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	12	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	99	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$$W_SDE_CYL

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000004@	L*1	LUN	AP-00000008@	I*2	SMALL_DISK_CYLINDER

LABELS

Address	Label
1-00000004	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
	LINCHK

00
00
00

```
0001
0002
0003
0004            Subroutine MSLG$$L_HEADER (lun,header)
0005
0006
0007            byte            lun
0008
0009            integer*4       header
0010            integer*4       lbn
0011            integer*4       code
0012            integer*4       compress4
0013
0014
0015            call linchk (lun,1)
0016
0017            write(lun,5) header
0018    5       format(' ',t8,'MSLG$L_HEADER',t24,z8.8)
0019
0020
0021            Entry DISK_HEADER (lun,header)
0022
0023
0024            lbn = lib$extzv(0,28,header)
0025            code = lib$extzv(28,4,header)
0026
0027            call linchk (lun,2)
0028
0029            if (code .eq. 0) then
0030
0031            write(lun,10) lbn,'GOOD LOGICAL SECTOR'
0032    10      format(' ',t40,'LBN #',i<compress4 (lbn)>,'.',/,t40,a)
0033
0034            else if (code .eq. 3) then
0035
0036            write(lun,10) lbn,'BAD REVECTORED LOG. SECTOR'
0037
0038            else if (code .eq. 5) then
0039
0040            write(lun,10) lbn,'BAD PRIMARY REVECT. LOG. SECTOR'
0041
0042            else if (code .eq. 6) then
0043
0044            write(lun,11) lbn,'BAD REPLACEMENT SECTOR'
0045    11      format(' ',t40,'RBN #',i<compress4 (lbn)>,'.',/,t40,a)
0046
0047            else if (code .eq. 11) then
0048
0049            write(lun,12) 'BAD SECTOR #',lbn
0050    12      format(' ',t40,a,i<compress4 (lbn)>,'.')
0051
0052            else if (code .eq. 12) then
0053
0054            write(lun,13) lbn,'GOOD EXTERNAL SECTOR'
0055    13      format(' ',t40,'XBN #',i<compress4 (lbn)>,'.',/,t40,a)
0056
0057            else if (code .eq. 14) then
```

```
0058
0059            write(lun,14) lbn,'GOOD DIAGNOSTIC SECTOR'
0060     14     format(' ',t40,'DBN #',i<compress4 (lbn)>,'.',/,t40,a)
0061            else
0062
0063            write(lun,15) 'HEADER #',lbn,'.','RESERVED HEADER CODE, #',code,'.'
0064     15     format(' ',t40,a,i<compress4 (lbn)>,a,/,t40,a,i<compress4 (code)>,a)
0065            endif
0066
0067            return
0068            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	661	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	375	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	160	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	1196	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-0000003A		DISK_HEADER	0-00000000		MSLG$$L_HEADER

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name	Address	Type	Name
2-00000004	I*4	CODE	AP-00000008@	I*4	HEADER	2-00000000	I*4	LBN	AP-00000004@	L*1	LUN

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-000000CC	5'	1-000000E6	10'	1-00000100	11'	1-0000011A	12'	1-0000012A	13'	1-00000144	14'
1-0000015E	15'										

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK

```
0001
0002
0003
0004            Subroutine MSCP_RESERVEDB (lun,reservedb)
0005
0006
0007            byte            lun
0008            byte            reservedb
0009
0010
0011            call linchk (lun,i)
0012
0013            write(lun,10) 'RESERVED',reservedb
0014     10     format(' ',t8,a,t30,z2.2)
0015
0016            return
0017            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	66	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	24	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	20	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	110	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP_RESERVEDB

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000004@	L*1	LUN	AP-00000008@	L*1	RESERVEDB

LABELS

Address	Label
1-0000000C	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
	LINCHK

MS

00
00
00

```
0001
0002
0003
0004            subroutine MSCP_RESERVEDW (lun,reservedw)
0005
0006
0007            byte            lun
0008
0009            integer*2       reservedw
0010
0011
0012            call linchk (lun,1)
0013
0014            write(lun,10) 'RESERVED',reservedw
0015    10      format(' ',t8,a,t28,z4.4)
0016
0017            return
0018            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	66	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	24	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	20	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	110	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP_RESERVEDW

VARIABLES

Address	Type	Name	Address	Type	Name
AP-0000004@	L*1	LUN	AP-00000008@	I*2	RESERVEDW

LABELS

Address	Label
1-0000000C	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
	LINCHK

```
                subroutine MSCP_RESERVEDL (lun,reservedl)

                byte            lun

                integer*4       reservedl

                call linchk (lun,1)

                write(lun,10) 'RESERVED',reservedl
        10      format(' ',t8,a,t24,z8.8)

                return
                end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	66	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	24	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	20	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	110	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP_RESERVEDL

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000004@	L*1	LUN	AP-00000008@	I*4	RESERVEDL

LABELS

Address	Label
1-0000000C	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
	LINCHK

```
0001
0002
0003
0004            Subroutine MSCP_RESERVEDQ (lun,reservedq)
0005
0006
0007            byte            lun
0008
0009            integer*4       reservedq(2)
0010
0011
0012            call linchk (lun,2)
0013
0014            write(lun,10) 'RESERVED',(reservedq(i),i = 1,2)
0015    10      format(' ',t8,a,t24,z8.8,/,t24,z8.8)
0016
0017            return
0018            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	92	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	30	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	44	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	166	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP_RESERVEDQ

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000000	I*4	I	AP-0000004@	L*1	LUN

ARRAYS

Address	Type	Name	Bytes	Dimensions
AP-00000008@	I*4	RESERVEDQ	8	(2)

LABELS

Address	Label
1-0000000C	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
	LINCHK

```
0001
0002
0003
0004          Subroutine MSCP$$W_UNT_FLGS (lun,unit_flags,expand)
0005
0006
0007          byte            lun
0008
0009          integer*2       unit_flags
0010
0011          logical*1       expand
0012
0013          character*17    v1unit_flags(0:2)
0014          data            v1unit_flags(0) /'COMPARE READS*'/
0015          data            v1unit_flags(1) /'COMPARE WRITES*'/
0016          data            v1unit_flags(2) /'576 BYTE SECTORS*'/
0017
0018          character*26    v2unit_flags(6:7)
0019          data            v2unit_flags(6) /'WRITE BACK (NON-VOLATILE)*'/
0020          data            v2unit_flags(7) /'REMOVABLE MEDIA*'/
0021
0022          character*35    v3unit_flags(10:15)
0023          data            v3unit_flags(10)/'SUPPRESS CACHING (LOW SPEED)*'/
0024          data            v3unit_flags(11)/'SUPPRESS CACHING (HIGH SPEED*'/
0025          data            v3unit_flags(12)/'WRITE PROTECT (SOFTWARE OR VOLUME)*'/
0026          data            v3unit_flags(13)/'WRITE PROTECT (HARDWARE)*'/
0027          data            v3unit_flags(14)/'INACTIVE SHADOW SET UNIT*'/
0028          data            v3unit_flags(15)/'CONTROLLER BAD BLOCK REPLACEMENT*'/
0029
0030
0031          call linchk (lun,1)
0032
0033          write(lun,10) unit_flags
0034    10    format(' ',t8,'MSCP$W_UNT_FLGS',t28,z4.4)
0035
0036          if (expand) then
0037
0038          call output (lun,unit_flags,v1unit_flags,0,0,2,'0')
0039
0040          call output (lun,unit_flags,v2unit_flags,6,6,7,'0')
0041
0042          call output (lun,unit_flags,v3unit_flags,10,10,15,'0')
0043          endif
0044
0045          return
0046          end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	122	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	58	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	528	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	708	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP$$W_UNT_FLGS

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name
AP-0000000C@	L*1	EXPAND	AP-00000004@	L*1	LUN	AP-00000008@	I*2	UNIT_FLAGS

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	CHAR	V1UNIT_FLAGS	51	(0:2)
2-00000033	CHAR	V2UNIT_FLAGS	52	(6:7)
2-00000067	CHAR	V3UNIT_FLAGS	210	(10:15)

LABELS

Address	Label
1-0000001E	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	LINCHK		OUTPUT

```
0001
0002
0003
0004          Subroutine MSCP$$L_MEDIA_ID (lun,media_id,expand)
0005
0006
0007          byte            lun
0008
0009          integer*4       media_id
0010
0011          logical*1       expand
0012
0013          character*2             letter(31)
0014          data            letter(1)       /'A*'/
0015          data            letter(2)       /'B*'/
0016          data            letter(3)       /'C*'/
0017          data            letter(4)       /'D*'/
0018          data            letter(5)       /'E*'/
0019          data            letter(6)       /'F*'/
0020          data            letter(7)       /'G*'/
0021          data            letter(8)       /'H*'/
0022          data            letter(9)       /'I*'/
0023          data            letter(10)      /'J*'/
0024          data            letter(11)      /'K*'/
0025          data            letter(12)      /'L*'/
0026          data            letter(13)      /'M*'/
0027          data            letter(14)      /'N*'/
0028          data            letter(15)      /'O*'/
0029          data            letter(16)      /'P*'/
0030          data            letter(17)      /'Q*'/
0031          data            letter(18)      /'R*'/
0032          data            letter(19)      /'S*'/
0033          data            letter(20)      /'T*'/
0034          data            letter(21)      /'U*'/
0035          data            letter(22)      /'V*'/
0036          data            letter(23)      /'W*'/
0037          data            letter(24)      /'X*'/
0038          data            letter(25)      /'Y*'/
0039          data            letter(26)      /'Z*'/
0040          data            letter(27)      /'.*'/
0041          data            letter(28)      /'.*'/
0042          data            letter(29)      /'.*'/
0043          data            letter(30)      /'.*'/
0044          data            letter(31)      /'.*'/
0045
0046          integer*4       d(0:1)
0047          integer*4       a(0:2)
0048          integer*4       n
0049          integer*4       compress4
0050
0051          character*2     d_string
0052          character*3     a_string
0053
0054
0055          call linchk (lun,1)
0056
0057          write(lun,5) 'MSCP$L_MEDIA_ID',media_id
```

```
0058     5      format(' ',t8,a,t24,z8.8)
0059
0060            if (expand) then
0061
0062            d(0) = lib$extzv(27,5,media_id)
0063
0064            d(1) = lib$extzv(22,5,media_id)
0065
0066            a(0) = lib$extzv(17,5,media_id)
0067
0068            a(1) = lib$extzv(12,5,media_id)
0069
0070            a(2) = lib$extzv(7,5,media_id)
0071
0072            n = lib$extzv(0,7,media_id)
0073
0074            do 10,i = 0,1
0075
0076            if (d(i) .eq. 0) then
0077
0078            d_string(i+1:i+1) = char(0)
0079            else
0080
0081            d_string(i+1:i+1) = letter(d(i))
0082            endif
0083
0084     10     continue
0085
0086            do 15,i = 0,2
0087
0088            if (a(i) .eq. 0) then
0089
0090            a_string(i+1:i+1) = char(0)
0091            else
0092
0093            a_string(i+1:i+1) = letter(a(i))
0094            endif
0095
0096     15     continue
0097
0098            call linchk (lun,2)
0099
0100            write(lun,20) 'DEVICE TYPE, /',d_string,'/'
0101     20     format(' ',t40,3(a))
0102
0103            write(lun,25) 'MEDIA TYPE, /',a_string,n,'/'
0104     25     format(' ',t40,a,a,i<compress4 (n)>,a)
0105            endif
0106
0107            return
0108            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	393	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	118	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	272	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	783	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP$$L_MEDIA_ID

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name	Address	Type	Name
2-00000054	CHAR	A_STRING	2-00000052	CHAR	D_STRING	AP-0000000C@	L*1	EXPAND	2-0000005C	I*4	I
AP-00000004@	L*1	LUN	AP-00000008@	I*4	MEDIA_ID	2-00000058	I*4	N			

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000008	I*4	A	12	(0:2)
2-00000000	I*4	D	8	(0:1)
2-00000014	CHAR	LETTER	62	(31)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-0000004F	5'	**	10	**	15	1-0000005B	20'	1-00000067	25'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK

```
0001
0002
0003
0004            Subroutine MSCP$$B_OPCODE (lun,opcode,expand)
0005
0006
0007            byte            lun
0008            byte            opcode
0009
0010            logical*1       expand
0011
0012            integer*4       control_message_type
0013            integer*4       compress4
0014
0015
0016            call linchk (lun,1)
0017
0018            If (LIB$EXTZV(6,2,opcode) .EQ. 0) then
0019
0020            write(lun,10) 'MSCP$B_OPCODE',opcode
0021    10      format(' ',t8,a,t30,z2.2)
0022
0023            Else if (LIB$EXTZV(6,2,opcode) .EQ. 1) then
0024
0025            Write (lun,10) 'MSCP$B_ATNCODE',opcode
0026
0027            Else if (LIB$EXTZV(6,2,opcode) .EQ. 2) then
0028
0029            Write (lun,10) 'MSCP$B_ENDCODE',opcode
0030            Else
0031
0032            Write (lun,12) opcode
0033    12      Format (' ',T30,Z2.2)
0034            Endif
0035
0036            if (expand) then
0037
0038            call linchk (lun,1)
0039
0040            control_message_type = lib$extzv(0,8,opcode)
0041
0042            if (control_message_type .eq. 1) then
0043
0044            write(lun,15) 'ABORT'
0045    15      format(' ',t40,a)
0046
0047            else if (control_message_type .eq. 2) then
0048
0049            write(lun,15) 'GET COMMAND STATUS'
0050
0051            else if (control_message_type .eq. 3) then
0052
0053            write(lun,15) 'GET UNIT STATUS'
0054
0055            else if (control_message_type .eq. 4) then
0056
0057            write(lun,15) 'SET CONTROLLER CHARACTERISTICS'
```

```
0058
0059            else if (control_message_type .eq. 8) then
0060
0061            write(lun,15) 'AVAILABLE'
0062
0063            else if (control_message_type .eq. 9) then
0064
0065            write(lun,15) 'ONLINE'
0066
0067            else if (control_message_type .eq. 10) then
0068
0069            write(lun,15) 'SET UNIT CHARACTERISTICS'
0070
0071            else if (control_message_type .eq. 11) then
0072
0073            write(lun,15) 'DETERMINE ACCESS PATH(S)'
0074
0075            else if (control_message_type .eq. 16) then
0076
0077            write(lun,15) 'ACCESS'
0078
0079            else if (control_message_type .eq. 17) then
0080
0081            write(lun,15) 'COMPARE CONTROLLER DATA'
0082
0083            else if (control_message_type .eq. 18) then
0084
0085            write(lun,15) 'ERASE'
0086
0087            else if (control_message_type .eq. 19) then
0088
0089            write(lun,15) 'FLUSH'
0090
0091            else if (control_message_type .eq. 20) then
0092
0093            write(lun,15) 'REPLACE'
0094
0095            else if (control_message_type .eq. 32) then
0096
0097            write(lun,15) 'COMPARE HOST DATA'
0098
0099            else if (control_message_type .eq. 33) then
0100
0101            write(lun,15) 'READ'
0102
0103            else if (control_message_type .eq. 34) then
0104
0105            write(lun,15) 'WRITE'
0106
0107            else if (control_message_type .eq. 64) then
0108
0109            write(lun,15) 'AVAILABLE'
0110
0111            else if (control_message_type .eq. 65) then
0112
0113            write(lun,15) 'DUPLICATE UNIT NUMBER'
0114
```

```
0115            else if (control_message_type .eq. 66) then
0116
0117            write(lun,15) 'ACCESS PATH'
0118            else
0119
0120            write(lun,20) control_message_type
0121    20      format(' ',t40,'CONTROL MESSAGE TYPE, #'
0122            1 i<compress4 (control_message_type)>,'.')
0123            endif
0124            endif
0125
0126            return
0127            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	1149	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	364	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	224	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	1737	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP$$B_OPCODE

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000000	I*4	CONTROL_MESSAGE_TYPE	AP-0000000C@	L*1	EXPAND
AP-00000004@	L*1	LUN	AP-00000008@	L*1	OPCODE

LABELS

Address	Label	Address	Label	Address	Label	Address	Label
1-00000128	10'	1-00000134	12'	1-0000013D	15'	1-00000144	20'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK

```
0001
0002
0003
0004          Subroutine MSCP$$W_MODIFIER (lun,opcode,command_modifier,expand)
0005
0006
0007          byte            lun
0008          byte            opcode
0009
0010          integer*2       command_modifier
0011
0012          logical*1       expand
0013
0014          integer*4       control_message_type
0015
0016          character*33    v1modifier(4:15)
0017          data            v1modifier(4)   /'WRITE SHADOW SET, SINGLE UNIT(S)*'/
0018          data            v1modifier(5)   /'WRITE BACK (VOLATILE)*'/
0019          data            v1modifier(6)   /'WRITE BACK (NON-VOLATILE)*'/
0020          data            v1modifier(7)   /'SUPPRESS SHADOWING*'/
0021          data            v1modifier(8)   /'SUPRESS ERROR RECOVERY*'/
0022          data            v1modifier(9)   /'SUPRESS ERROR CORRECTION*'/
0023          data            v1modifier(10)  /'SUPRESS CACHING (LOW SPEED)*'/
0024          data            v1modifier(11)  /'SUPPRESS CACHING (HIGH SPEED)*'/
0025          data            v1modifier(12)  /'FORCE ERROR*'/
0026          data            v1modifier(13)  /'CLEAR SERIOUS EXCEPTION*'/
0027          data            v1modifier(14)  /'COMPARE*'/
0028          data            v1modifier(15)  /'EXPRESS REQUEST*'/
0029
0030          character*10    v1gus_modifier(0:0)
0031          data            v1gus_modifier(0)       /'NEXT UNIT*'/
0032
0033          character*18    v1available_modifier(0:1)
0034          data            v1available_modifier(0) /'ALL CLASS DRIVERS*'/
0035          data            v1available_modifier(1) /'SPIN-DOWN*'/
0036
0037          character*27    v1online_modifier(0:4)
0038          data            v1online_modifier(0)    /'ALLOW SELF DESTRUCTION*'/
0039          data            v1online_modifier(1)    /'IGNORE MEDIA FORMAT ERROR*'/
0040          data            v1online_modifier(2)    /'ENABLE SET WRITE PROTECT*'/
0041          data            v1online_modifier(3)    /'CLEAR WRITE-BACK DATA LOST*'/
0042          data            v1online_modifier(4)    /'SHADOW UNIT SPECIFIED*'/
0043
0044          character*27    v1suc_modifier(2:4)
0045
0046          equivalence     (v1suc_modifier(2),v1online_modifier(2))
0047
0048          character*18    v1flush_modifier(0:1)
0049          data            v1flush_modifier(0)     /'FLUSH ENTIRE UNIT*'/
0050          data            v1flush_modifier(1)     /'VOLATILE ONLY*'/
0051
0052          character*26    v1replace_modifier(0:0)
0053          data            v1replace_modifier(0)   /'PRIMARY REPLACEMENT BLOCK*'/
0054
0055
0056          call linchk (lun,1)
0057
```

```
0058            write(lun,10) command_modifier
0059    10      format(' ',t8,'MSCP$W_MODIFIER',t28,z4.4)
0060
0061            if (expand) then
0062
0063            control_message_type = lib$extzv(0,8,opcode)
0064
0065            if (control_message_type .eq. 3) then
0066
0067            call output (lun,command_modifier,v1gus_modifier,0,0,0,'0')
0068            call output (lun,command_modifier,v1modifier,4,13,13,'0')
0069
0070            else if (control_message_type .eq. 8) then
0071
0072            call output (lun,command_modifier,v1available_modifier,0,0,1,'0')
0073            call output (lun,command_modifier,v1modifier,4,13,13,'0')
0074
0075            else if (control_message_type .eq. 9) then
0076
0077            call output (lun,command_modifier,v1online_modifier,0,0,4,'0')
0078            call output (lun,command_modifier,v1modifier,4,13,13,'0')
0079
0080            else if (control_message_type .eq. 10) then
0081
0082            call output (lun,command_modifier,v1suc_modifier,2,2,4,'0')
0083            call output (lun,command_modifier,v1modifier,4,13,13,'0')
0084
0085            else if (
0086            1 control_message_type .eq. 16
0087            1 .or.
0088            1 control_message_type .eq. 17
0089            1 .or.
0090            1 control_message_type .eq. 32
0091            1 ) then
0092
0093            call output (lun,command_modifier,v1modifier,4,7,11,'0')
0094            call output (lun,command_modifier,v1modifier,4,13,13,'0')
0095            call output (lun,command_modifier,v1modifier,4,15,15,'0')
0096
0097            else if (control_message_type .eq. 18) then
0098
0099            call output (lun,command_modifier,v1modifier,4,4,8,'0')
0100            call output (lun,command_modifier,v1modifier,4,12,13,'0')
0101            call output (lun,command_modifier,v1modifier,4,15,15,'0')
0102
0103            else if (control_message_type .eq. 19) then
0104
0105            call output (lun,command_modifier,v1flush_modifier,0,0,1,'0')
0106            call output (lun,command_modifier,v1modifier,4,7,9,'0')
0107            call output (lun,command_modifier,v1modifier,4,13,13,'0')
0108            call output (lun,command_modifier,v1modifier,4,15,15,'0')
0109
0110            else if (control_message_type .eq. 20) then
0111
0112            call output (lun,command_modifier,v1replace_modifier,0,0,0,'0')
0113            call output (lun,command_modifier,v1modifier,4,13,13,'0')
0114            call output (lun,command_modifier,v1modifier,4,15,15,'0')
```

```
0115
0116            else if (control_message_type .eq. 33) then
0117
0118            call output (lun,command_modifier,v1modifier,4,7,11,'0')
0119            call output (lun,command_modifier,v1modifier,4,13,15,'0')
0120
0121            else if (control_message_type .eq. 34) then
0122
0123            call output (lun,command_modifier,v1modifier,4,4,9,'0')
0124            call output (lun,command_modifier,v1modifier,4,12,15,'0')
0125            endif
0126            endif
0127
0128            return
0129            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	698	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	74	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	1704	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	2476	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP$$W_MODIFIER

VARIABLES

Address	Type	Name	Address	Type	Name
AP-0000000C@	I*2	COMMAND_MODIFIER	2-00000280	I*4	CONTROL_MESSAGE_TYPE
AP-00000010@	L*1	EXPAND	AP-00000004@	L*1	LUN
AP-00000008@	L*1	OPCODE			

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-0000021D	CHAR	V1AVAILABLE_MODIFIER	36	(0:1)
2-00000241	CHAR	V1FLUSH_MODIFIER	36	(0:1)
2-00000213	CHAR	V1GUS_MODIFIER	10	(0:0)
2-00000087	CHAR	V1MODIFIER	396	(4:15)
2-00000000	CHAR	V1ONLINE_MODIFIER	135	(0:4)
2-00000265	CHAR	V1REPLACE_MODIFIER	26	(0:0)
2-00000036	CHAR	V1SUC_MODIFIER	81	(2:4)

LABELS

Address	Label
1-0000002E	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	LIB$EXTZV		LINCHK		OUTPUT

```
0001
0002
0003
0004            Subroutine MSCP$$Z_BUFFER (lun,buffer)
0005
0006
0007            include 'src$:msghdr.for /nolist'
0066
0067
0068            byte            lun
0069
0070            integer*4       buffer(3)
0071
0072
0073            call linchk (lun,3)
0074
0075            write(lun,10) (buffer(i),i = 1,3)
0076    10      format(' ',t8,'MSCP$Z_BUFFER',t24,z8.8,/,t24,z8.8,/,t24,z8.8)
0077
0078            return
0079            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	82	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	42	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	36	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	672	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP$$Z_BUFFER

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000000	I*4	EMB$L_HD_SID	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	2-00000000	I*4	I
AP-00000004@	L*1	LUN			

ARRAYS

Address	Type	Name	Bytes	Dimensions
AP-00000008@	I*4	BUFFER	12	(3)
3-00000000	L*1	EMB	512	(0:511)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)

LABELS

Address	Label
1-00000004	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
	LINCHK

```
0001
0002
0003
0004            Subroutine MSCP$$L_DEV_PARM (lun,device_dependant_parameters)
0005
0006
0007            byte            lun
0008
0009            integer*4       device_dependant_parameters
0010
0011
0012            call linchk (lun,1)
0013
0014            write(lun,10) device_dependant_parameters
0015    10      format(' ',t8,'MSCP$[_DEV_PARM',t24,z8.8)
0016
0017            return
0018            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	56	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	32	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	12	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	100	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP$$L_DEV_PARM

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008@	I*4	DEVICE_DEPENDANT_PARAMETERS	AP-00000004@	L*1	LUN

LABELS

Address	Label
1-00000004	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
	LINCHK

```
0001
0002
0003
0004            Subroutine MSCP$$W_SHDW_UNT (lun,shadow_unit,expand)
0005
0006
0007            byte            lun
0008
0009            integer*2       shadow_unit
0010
0011            logical*1       expand
0012
0013            integer*4       compress4
0014
0015
0016
0017            call linchk (lun,1)
0018
0019            write(lun,10) shadow_unit
0020    10      format(' ',t8,'MSCP$W_SHDW_UNT',t28,z4.4)
0021
0022            if (expand) then
0023
0024            call linchk (lun,1)
0025
0026            write(lun,15) shadow_unit
0027    15      format(' ',t40,'SHADOW UNIT #',i<compress4 (shadow_unit)>,'.')
0028            endif
0029
0030            return
0031            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	138	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	62	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	24	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	224	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP$$W_SHDW_UNT

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name
AP-0000000C@	L*1	EXPAND	AP-00000004@	L*1	LUN	2-00000000@	I*2	SHADOW_UNIT

LABELS

Address	Label	Address	Label
1-00000004	10'	1-00000020	15'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
I*4	COMPRESS4		LINCHK

```
0001
0002
0003
0004            Subroutine MSCP$$W_COPY_SPD (lun,copy_speed)
0005
0006
0007            byte            lun
0008
0009            integer*2       copy_speed
0010
0011
0012            call linchk (lun,1)
0013
0014            write(lun,10) copy_speed
0015    10      format(' ',t8,'MSCP$W_COPY_SPD',t28,z4.4)
0016
0017            return
0018            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	56	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	32	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	12	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	100	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP$$W_COPY_SPD

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008@	I*2	COPY_SPEED	AP-00000004@	L*1	LUN

LABELS

Address	Label
1-00000004	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
	LINCHK

```
0001
0002
0003
0004              Subroutine MSCP$$W_VERSION (lun,mscp_version)
0005
0006
0007              byte              lun
0008
0009              integer*2         mscp_version
0010
0011
0012              call linchk (lun,1)
0013
0014              write(lun,10) mscp_version
0015        10    format(' ',t8,'MSCP$W_VERSION',t28,z4.4)
0016
0017              return
0018              end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	56	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	31	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	12	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	99	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP$$W_VERSION

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000004@	L*1	LUN	AP-00000008@	I*2	MSCP_VERSION

LABELS

Address	Label
1-00000004	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
	LINCHK

```
      Subroutine MSCP$$W_CNT_FLGS (lun,controller_flags,expand)

      byte            lun

      integer*2       controller_flags

      logical*1       expand

      character*17    v1controller_flags(0:1)
      data            v1controller_flags(0)   /'576 BYTE SECTORS*'/
      data            v1controller_flags(1)   /'SHADOWING*'/

      character*34    v2controller_flags(4:7)

      data            v2controller_flags(4)
      1 /'ENABLE ERROR LOG MSG., THIS HOST*'/

      data            v2controller_flags(5)
      1 /'ENABLE ERROR LOG MSG., OTHER HOST*'/

      data            v2controller_flags(6)
      1 /'ENABLE MISC. ERROR LOG MESSAGES*'/

      data            v2controller_flags(7)
      1 /'ENABLE "ATTENTION" MESSAGES*'/

      character*33    v3controller_flags(15:15)

      data            v3controller_flags(15)
      1 /'CONTROLLER BAD BLOCK REPLACEMENT*'/


      call linchk (lun,1)

      write(lun,10) controller_flags
10    format(' ',t8,'MSCP$W_CNT_FLGS',t28,z4.4)

      if (expand) then

      call output (lun,controller_flags,v1controller_flags,0,0,1,'0')
      call output (lun,controller_flags,v2controller_flags,4,4,7,'0')
      call output (lun,controller_flags,v3controller_flags,15,15,15,'0')
      endif

      return
      end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	122	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	50	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	416	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	588	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP$$W_CNT_FLGS

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008@	I*2	CONTROLLER_FLAGS	AP-0000000C@	L*1	EXPAND
AP-00000004@	L*1	LUN			

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	CHAR	V1CONTROLLER_FLAGS	34	(0:1)
2-00000022	CHAR	V2CONTROLLER_FLAGS	136	(4:7)
2-000000AA	CHAR	V3CONTROLLER_FLAGS	33	(15:15)

LABELS

Address	Label
1-00000016	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	LINCH%		OUTPUT

```
0001
0002
0003
0004            Subroutine MSCP$$W_HST_TMO (lun,host_timeout,expand)
0005
0006
0007            byte            lun
0008
0009            integer*2       host_timeout
0010
0011            logical*1       expand
0012
0013            integer*4       compress4
0014
0015
0016            call linchk (lun,1)
0017
0018            write(lun,10) host_timeout
0019    10      format(' ',t8,'MSCP$W_HST_TMO',t28,z4.4)
0020
0021            if (expand) then
0022
0023            call linchk (lun,1)
0024
0025            write(lun,15) host_timeout
0026    15      format(' ',t40,'HOST TIMEOUT, ',i<compress4 (host_timeout)>,
0027            1 '. SECONDS')
0028            endif
0029
0030            return
0031            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	138	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	70	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	24	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	232	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP$$W_HST_TMO

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name
AP-0000000C@	L*1	EXPAND	2-00000000@	I*2	HOST_TIMEOUT	AP-00000004@	L*1	LUN

LABELS

Address	Label	Address	Label
1-00000004	10'	1-0000001F	15'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
I*4	COMPRESS4		LINCHK

```
0001
0002
0003
0004            Subroutine MSCP$$Q_TIME (lun,vms_time,expand)
0005
0006
0007            byte            lun
0008
0009            integer*4       vms_time(2)
0010
0011            logical*1       expand
0012
0013            logical*4       sys$asctim
0014
0015            character*23    timbuf
0016
0017
0018            call linchk (lun,2)
0019
0020            write(lun,10) vms_time(1)
0021    10      format(' ',t8,'MSCP$Q_TIME',t24,z8.8)
0022
0023            if (expand) then
0024
0025            if (sys$asctim (,timbuf,vms_time,%val(0))) then
0026
0027            write(lun,15) vms_time(2),timbuf
0028    15      format(' ',t24,z8.8,:t40,:a)
0029
0030            goto 20
0031            endif
0032            endif
0033
0034            write(lun,15) vms_time(2)
0035
0036    20      return
0037            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	173	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	42	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	84	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	299	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP$$Q_TIME

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name
AP-0000000C@	L*1	EXPAND	AP-00000004@	L*1	LUN	2-00000000	CHAR	TIMBUF

ARRAYS

Address	Type	Name	Bytes	Dimensions
AP-00000008@	I*4	VMS_TIME	8	(2)

LABELS

Address	Label	Address	Label	Address	Label
1-00000004	10'	1-0000001C	15'	0-000000AC	20

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	LINCHK	L*4	SYS$ASCTIM

017
017
017
017
017
017
018
018
018
018
018
018
018
018
018
019
019
019
019
019
019
019
019
019
020
020
020
020
020
020
021
021
021
021
021
021
021
021
021
021
022

```
0001
0002
0003
0004            Subroutine MSCP$$W_STATUS (lun,end_message_status_code,expand)
0005
0006
0007
0008            byte            lun
0009
0010            integer*2       end_message_status_code
0011
0012            logical*1       expand
0013
0014            integer*4       status_code
0015            integer*4       compress4
0016
0017
0018
0019            call linchk (lun,1)
0020
0021            write(lun,10) end_message_status_code
0022    10      format(' ',t8,'MSCP$W_STATUS',t28,z4.4)
0023
0024            if (expand) then
0025
0026            Call STATUS_AND_EVENT_CODES (lun)
0027            Endif
0028
0029            return
0030            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	75	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	30	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	28	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	133	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP$$W_STATUS

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000004	I*4	COMPRESS4	AP-00000008@	I*2	END_MESSAGE_STATUS_CODE
AP-0000000C@	L*1	EXPAND	AP-00000004@	L*1	LUN
2-00000000	I*4	STATUS_CODE			

LABELS

Address	Label
1-00000004	10'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	LINCHK		STATUS_AND_EVENT_CODES

```
0001
0002
0003
0004          Subroutine MSCP_FIRST_TWELVE_BYTES1 (lun,expand)
0005
0006
0007          include 'src$:msghdr.for /nolist'
0066
0067
0068          byte            lun
0069
0070          logical*1       expand
0071
0072          integer*4       mscp$l_cmd_ref
0073
0074          integer*2       mscp$w_unit
0075          integer*2       reserved1
0076
0077          byte            mscp$b_opcode
0078          byte            reserved2
0079
0080          integer*2       mscp$w_modifier
0081
0082          integer*4       mscp$l_out_ref
0083
0084          equivalence     (emb(38),mscp$l_cmd_ref)
0085          equivalence     (emb(42),mscp$w_unit)
0086          equivalence     (emb(44),reserved1)
0087          equivalence     (emb(46),mscp$b_opcode)
0088          equivalence     (emb(47),reserved2)
0089          equivalence     (emb(48),mscp$w_modifier)
0090
0091          Logical*1       End_message
0092
0093          End_message = .false.
0094
0095          If (LIB$EXTZV(7,1,mscp$b_opcode) .EQ. 1) then
0096
0097          End_message = .true.
0098          Endif
0099
0100          call mscp$$l_cmd_ref (lun,mscp$l_cmd_ref)
0101
0102          call mscp$$w_unit (lun,mscp$w_unit,expand)
0103
0104          call mscp_reservedw (lun,reserved1)
0105
0106          call mscp$$b_opcode (lun,mscp$b_opcode,expand)
0107
0108          If (.NOT. end_message) then
0109
0110          call mscp_reservedb (lun,reserved2)
0111          Else
0112
0113          Call MSCP$$B_FLAGS (lun,reserved2,expand)
0114          Endif
0115
```

```
0116            If (.NOT. end_message) then
0117
0118            call mscp$$w_modifier (lun,mscp$b_opcode,mscp$w_modifier,expand)
0119            Else
0120
0121            Call MSCP$$W_STATUS (lun,mscp$w_modifier,expand)
0122            Endif
0123
0124            return
0125            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	172	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	8	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	144	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	836	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP_FIRST_TWELVE_BYTES1

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000000	I*4	EMB$L_HD_SID	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	2-00000000	L*1	END_MESSAGE
AP-00000008@	L*1	EXPAND	AP-00000004@	L*1	LUN
3-0000002E	L*1	MSCP$B_OPCODE	3-00000026	I*4	MSCP$L_CMD_REF
2-00000004	I*4	MSCP$L_OUT_REF	3-00000030	I*2	MSCP$W_MODIFIER
3-0000002A	I*2	MSCP$W_UNIT	3-0000002C	I*2	RESERVED1
3-0000002F	L*1	RESERVED2			

ARRAYS

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

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	LIB$EXTZV		MSCP$$B_FLAGS		MSCP$$B_OPCODE
	MSCP$$L_CMD_REF		MSCP$$W_MODIFIER		MSCP$$W_STATUS
	MSCP$$W_UNIT		MSCP_RESERVEDB		MSCP_RESERVEDW

RA6

011
011
011
011
011
012
012
012
012
012
012
012
012
012
012
013
013
013
013
013
013
013
013
013
013
013
013
013
013
014
014
014
014
014
014
014
014
014
015
015
015
015
015
015
015
015
015
015
016
016
016
016
016
016
016
016
016
016
017
017

```
0001
0002
0003
0004          Subroutine MSCP_FIRST_TWELVE_BYTES2 (lun,expand)
0005
0006
0007          include 'src$:msghdr.for /nolist'
0066
0067
0068          byte            lun
0069
0070          logical*1       expand
0071
0072          integer*4       reserved1
0073
0074          integer*2       mscp$w_unit
0075          integer*2       reserved2
0076
0077          byte            mscp$b_opcode
0078          byte            reserved3
0079
0080          integer*2       reserved4
0081
0082          equivalence     (emb(38),reserved1)
0083          equivalence     (emb(42),mscp$w_unit)
0084          equivalence     (emb(44),reserved2)
0085          equivalence     (emb(46),mscp$b_opcode)
0086          equivalence     (emb(47),reserved3)
0087          equivalence     (emb(48),reserved4)
0088
0089
0090          call mscp_reservedl (lun,reserved1)
0091
0092          call mscp$$w_unit (lun,mscp$w_unit,expand)
0093
0094          call mscp_reservedw (lun,reserved2)
0095
0096          call mscp$$b_opcode (lun,mscp$b_opcode,expand)
0097
0098          call mscp_reservedb (lun,reserved3)
0099
0100          call mscp_reservedw (lun,reserved4)
0101
0102          return
0103          end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	97	PIC CON REL LCL SHR EXE RD NOWRT LONG
2	$LOCAL	80	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	689	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP_FIRST_TWELVE_BYTES2

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000000	I*4	EMB$L_HD_SID	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	AP-00000008@	L*1	EXPAND
AP-0000004@	L*1	LUN	3-0000002E	L*1	MSCP$B_OPCODE
3-0000002A	I*2	MSCP$W_UNIT	3-00000026	I*4	RESERVED1
3-0000002C	I*2	RESERVED2	3-0000002F	L*1	RESERVED3
3-00000030	I*2	RESERVED4			

ARRAYS

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

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	MSCP$$B_OPCODE		MSCP$$W_UNIT		MSCP_RESERVEDB
	MSCP_RESERVEDL		MSCP_RESERVEDW		

```
      subroutine MSCP_INVALID_COMMAND_END_MESAGE (lun)

      include 'src$:msghdr.for /nolist'

      byte            lun

      integer*4       mscp$l_cmd_ref

      integer*2       mscp$w_unit
      integer*2       reserved1

      byte            mscp$b_endcode
      byte            mscp$b_flags

      integer*2       mscp$w_status

      byte            manufactured_mscp$b_opcode

      equivalence     (emb(38),mscp$l_cmd_ref)
      equivalence     (emb(42),mscp$w_unit)
      equivalence     (emb(44),reserved1)
      equivalence     (emb(46),mscp$b_endcode)
      equivalence     (emb(47),mscp$b_flags)
      equivalence     (emb(48),mscp$w_status)
      equivalence     (emb(94),manufactured_mscp$b_opcode)

      call linchk (lun,1)

      write(lun,10)
10    format(' ',:)

      call mscp$$l_cmd_ref (lun,mscp$l_cmd_ref)

      call mscp$$w_unit (lun,mscp$w_unit,.true.)

      call mscp_reservedw (lun,reserved1)

      Call MSCP$$B_OPCODE (lun,mscp$b_endcode, .true.)

      Call MSCP$$B_FLAGS (lun,mscp$b_flags, .true.)

      Call MSCP$$W_STATUS (lun,mscp$w_status, .true.)

      Call MSCP$B_OPCODE_DISPATCHER (lun,manufactured_mscp$b_opcode,
      1 .false., .true.)

      call linchk (lun,1)

      write(lun,25)
25    format(/' ','RE-MANUFACTURED "MSCP" CONTROL MESSAGE')
```

```
0116            call movc3 (%val(36),emb(86),emb(38))
0117
0118            call mscp$b_opcode_dispatcher (lun,manufactured_mscp$b_opcode,.true.,
0119            1 .false.)
0120
0121            return
0122            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	196	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	62	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	156	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	926	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP_INVALID_COMMAND_END_MESAGE

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000000	I*4	EMB$L_HD_SID	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	AP-00000004@	L*1	LUN
3-0000005E	L*1	MANUFACTURED_MSCP$B_OPCODE	3-0000002E	L*1	MSCP$B_ENDCODE
3-0000002F	L*1	MSCP$B_FLAGS	3-00000026	I*4	MSCP$L_CMD_REF
3-00000030	I*2	MSCP$W_STATUS	3-0000002A	I*2	MSCP$W_UNIT
3-0000002C	I*2	RESERVED1			

ARRAYS

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

LABELS

Address	Label	Address	Label
1-0000000C	10'	1-00000011	25'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	LINCHK		MOVC3		MSCP$$B_FLAGS
	MSCP$$B_OPCODE		MSCP$$L_CMD_REF		MSCP$$W_STATUS
	MSCP$$W_UNIT		MSCP$B_OPCODE_DISPATCHER		MSCP_RESERVEDW

```
0001
0002
0003
0004          Subroutine MSCP$B_OPCODE_DISPATCHER (lun,mscp$b_opcode,expand,invalid)
0005
0006
0007          include 'src$:msghdr.for /nolist'
0066
0067
0068          byte            lun
0069          byte            mscp$b_opcode
0070
0071          logical*1       expand
0072          logical*1       invalid
0073
0074          integer*4       mscp$l_opcode
0075
0076
0077          mscp$l_opcode = lib$extzv(0,7,mscp$b_opcode)
0078
0079          if (.not. invalid) then
0080
0081          if (mscp$l_opcode .eq. 1) then
0082
0083          call mscp$$k_op_abort (lun,expand)
0084
0085          else if (mscp$l_opcode .eq. 2) then
0086          call mscp$$k_op_gtcmd (lun,expand)
0087
0088          else if (mscp$l_opcode .eq. 3) then
0089          call mscp$$k_op_gtunt (lun,expand)
0090
0091          else if (mscp$l_opcode .eq. 4) then
0092          call mscp$$k_op_stcon (lun,expand)
0093
0094          Else if (mscp$l_opcode .EQ. 7) then
0095          Call MSCP$$K_OP_SEREX (lun,expand)
0096
0097          else if (mscp$l_opcode .eq. 8) then
0098          call mscp$$k_op_avail (lun,expand)
0099
0100          else if (mscp$l_opcode .eq. 9) then
0101          call mscp$$k_op_onlin (lun,expand)
0102
0103          else if (mscp$l_opcode .eq. 10) then
0104          call mscp$$k_op_stunt (lun,expand)
0105
0106          else if (mscp$l_opcode .eq. 11) then
0107          call mscp$$k_op_dtacp (lun,expand)
0108
0109          else if (mscp$l_opcode .eq. 16) then
0110          call mscp$$k_op_acces (lun,expand)
0111
0112          else if (mscp$l_opcode .eq. 17) then
0113          call mscp$$k_op_cmpcd (lun,expand)
0114
0115          else if (mscp$l_opcode .eq. 18) then
```

```
0116            call mscp$$k_op_erase (lun,expand)
0117
0118            else if (mscp$l_opcode .eq. 19) then
0119            call mscp$$k_op_flush (lun,expand)
0120
0121            else if (mscp$l_opcode .eq. 20) then
0122            call mscp$$k_op_replc (lun,expand)
0123
0124            else if (mscp$l_opcode .eq. 32) then
0125            call mscp$$k_op_comp (lun,expand)
0126
0127            else if (mscp$l_opcode .eq. 33) then
0128            call mscp$$k_op_read (lun,expand)
0129
0130            else if (mscp$l_opcode .eq. 34) then
0131            call mscp$$k_op_write (lun,expand)
0132
0133            else if (mscp$l_opcode .eq. 64) then
0134            call mscp$$k_op_avatn (lun,expand)
0135
0136            else if (mscp$l_opcode .eq. 65) then
0137            call mscp$$k_op_dupun (lun,expand)
0138
0139            else if (mscp$l_opcode .eq. 66) then
0140            call mscp$$k_op_acpth (lun,expand)
0141
0142            else
0143            call linchk (lun,1)
0144
0145            write(lun,5)
0146     5      format(' ',:)
0147
0148            Call DUMPREG (lun,9,emb(38))
0149            endif
0150
0151            else if (invalid) then
0152            if (mscp$l_opcode .eq. 1) then
0153
0154            call invalid_mscp$$k_op_abort (lun,expand)
0155
0156            else if (mscp$l_opcode .eq. 2) then
0157            call invalid_mscp$$k_op_gtcmd (lun,expand)
0158
0159            else if (mscp$l_opcode .eq. 3) then
0160            call invalid_mscp$$k_op_gtunt (lun,expand)
0161
0162            else if (mscp$l_opcode .eq. 4) then
0163            call invalid_mscp$$k_op_stcon (lun,expand)
0164
0165            else if (mscp$l_opcode .eq. 8) then
0166            call invalid_mscp$$k_op_avail (lun,expand)
0167
0168            else if (mscp$l_opcode .eq. 9) then
0169            call invalid_mscp$$k_op_onlin (lun,expand)
0170
0171            else if (mscp$l_opcode .eq. 10) then
0172            call invalid_mscp$$k_op_stunt (lun,expand)
```

```
0173
0174            else if (mscp$l_opcode .eq. 11) then
0175            call invalid_mscp$$k_op_dtacp (lun,expand)
0176
0177            else if (mscp$l_opcode .eq. 16) then
0178            call invalid_mscp$$k_op_acces (lun,expand)
0179
0180            else if (mscp$l_opcode .eq. 17) then
0181            call invalid_mscp$$k_op_cmpcd (lun,expand)
0182
0183            else if (mscp$l_opcode .eq. 18) then
0184            call invalid_mscp$$k_op_erase (lun,expand)
0185
0186            else if (mscp$l_opcode .eq. 19) then
0187            call invalid_mscp$$k_op_flush (lun,expand)
0188
0189            else if (mscp$l_opcode .eq. 20) then
0190            call invalid_mscp$$k_op_replc (lun,expand)
0191
0192            else if (mscp$l_opcode .eq. 32) then
0193            call invalid_mscp$$k_op_comp (lun,expand)
0194
0195            else if (mscp$l_opcode .eq. 33) then
0196            call invalid_mscp$$k_op_read (lun,expand)
0197
0198            else if (mscp$l_opcode .eq. 34) then
0199            call invalid_mscp$$k_op_write (lun,expand)
0200
0201            else if (mscp$l_opcode .eq. 64) then
0202            call invalid_mscp$$k_op_avatn (lun,expand)
0203
0204            else if (mscp$l_opcode .eq. 65) then
0205            call invalid_mscp$$k_op_dupun (lun,expand)
0206
0207            else if (mscp$l_opcode .eq. 66) then
0208            call invalid_mscp$$k_op_acpth (lun,expand)
0209
0210            else
0211            call linchk (lun,1)
0212
0213            write(lun,5)
0214
0215            Call DUMPREG (lun,9,emb(38))
0216            endif
0217            endif
0218
0219            return
0220            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	1180	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	21	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	60	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	1773	

ENTRY POINTS

Address	Type	Name
0-00000000		MSCP$B_OPCODE_DISPATCHER

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000000	I*4	EMB$L_HD_SID	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	AP-0000000C@	L*1	EXPAND
AP-00000010@	L*1	INVALID	AP-00000004@	L*1	LUN
AP-00000008@	L*1	MSCP$B_OPCODE	2-00000000	I*4	MSCP$L_OPCODE

ARRAYS

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

LABELS

Address	Label
1-00000010	5'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	DUMPREG		INVALID_MSCP$$K_OP_ABORT		INVALID_MSCP$$K_OP_ACCES
	INVALID_MSCP$$K_OP_ACPTH		INVALID_MSCP$$K_OP_AVAIL		INVALID_MSCP$$K_OP_AVATN
	INVALID_MSCP$$K_OP_CMPCD		INVALID_MSCP$$K_OP_COMP		INVALID_MSCP$$K_OP_DTACP
	INVALID_MSCP$$K_OP_DUPUN		INVALID_MSCP$$K_OP_ERASE		INVALID_MSCP$$K_OP_FLUSH
	INVALID_MSCP$$K_OP_GTCMD		INVALID_MSCP$$K_OP_GTUNT		INVALID_MSCP$$K_OP_ONLIN
	INVALID_MSCP$$K_OP_READ		INVALID_MSCP$$K_OP_REPLC		INVALID_MSCP$$K_OP_STCON
	INVALID_MSCP$$K_OP_STUNT		INVALID_MSCP$$K_OP_WRITE	I*4	LIB$EXTZV
	LINCHK		MSCP$$K_OP_ABORT		MSCP$$K_OP_ACCES
	MSCP$$K_OP_ACPTH		MSCP$$K_OP_AVAIL		MSCP$$K_OP_AVATN

```
        MSCP$$K_OP_CMPCD        MSCP$$K_OP_COMP         MSCP$$K_OP_DTACP
        MSCP$$K_OP_DUPUN        MSCP$$K_OP_ERASE        MSCP$$K_OP_FLUSH
        MSCP$$K_OP_GTCMD        MSCP$$K_OP_GTUNT        MSCP$$K_OP_ONLIN
        MSCP$$K_OP_READ         MSCP$$K_OP_REPLC        MSCP$$K_OP_SEREX
        MSCP$$K_OP_STCON        MSCP$$K_OP_STUNT        MSCP$$K_OP_WRITE
```

```
0001
0002
0003
0004          Subroutine RA60_STATUS (lun,sdi1,sdi2,sdi3)
0005
0006
0007          byte              lun
0008
0009          integer*4         sdi1
0010          integer*4         sdi2
0011          integer*4         sdi3
0012          integer*4         previous_cylinder
0013          integer*4         previous_head
0014          integer*4         current_cylinder
0015          integer*4         current_head
0016          integer*4         error_code
0017          integer*4         compress4
0018
0019
0020          previous_cylinder = lib$extzv(8,16,sdi2)
0021
0022          call linchk (lun,1)
0023
0024          write(lun,10) 'PREVIOUS CYLINDER, #',previous_cylinder
0025    10    format(' ',t40,a,i<compress4 (previous_cylinder)>,'.')
0026
0027          previous_head = lib$extzv(24,8,sdi2)
0028
0029          call linchk (lun,1)
0030
0031          write(lun,15) 'PREVIOUS HEAD, #',previous_head
0032    15    format(' ',t40,a,i<compress4 (previous_head)>,'.')
0033
0034          current_cylinder = lib$extzv(0,16,sdi3)
0035
0036          call linchk (lun,1)
0037
0038          write(lun,20) 'CURRENT CYLINDER, #',current_cylinder
0039    20    format(' ',t40,a,i<compress4 (current_cylinder)>,'.')
0040
0041          current_head = lib$extzv(16,8,sdi3)
0042
0043          call linchk (lun,1)
0044
0045          write(lun,25) 'CURRENT HEAD, #',current_head
0046    25    format(' ',t40,a,i<compress4 (current_head)>,'.')
0047
0048          error_code = lib$extzv(24,8,sdi3)
0049
0050          call linchk (lun,1)
0051
0052          write(lun,30) 'ERROR CODE ',error_code
0053    30    format(' ',t40,a,z2.2,' (HEX)')
0054
0055          call linchk (lun,1)
0056
0057          if (error_code .eq. '01'x) then
```

```
0058
0059            call linchk (lun,1)
0060
0061            write(lun,35) 'COMMAND CANNOT BE EXECUTED',
0062           1 ' - WITH DRIVE IN CURRENT STATE'
0063      35    format(' ',t40,a,/,t40,a)
0064
0065            else if (error_code .eq. '02'x) then
0066
0067            call linchk (lun,1)
0068
0069            write(lun,35) 'SEEK REQUESTED NON-EXISTENT',
0070           1 ' - HEAD OR CYLINDER'
0071
0072            else if (error_code .eq. '03'x) then
0073
0074            write(lun,40) 'NOT A VALID OPCODE'
0075      40    format(' ',t40,a)
0076
0077            else if (error_code .eq. '04'x) then
0078
0079            call linchk (lun,1)
0080
0081            write(lun,35) 'COVER NOT CLOSED',
0082           1 ' - WHEN "RUN" ISSUED'
0083
0084            else if (error_code .eq. '05'x) then
0085
0086            write(lun,40) 'LID NOT LOCKED WHEN "RUN" ISSUED'
0087
0088            else if (error_code .eq. '06'x) then
0089
0090            write(lun,40) 'PROGRAM ERROR DURING HEAD LOAD'
0091
0092            else if (error_code .eq. '08'x) then
0093
0094            write(lun,40) 'HEADS NOT HOME WHEN "RUN" ISSUED'
0095
0096            else if (error_code .eq. '09'x) then
0097
0098            call linchk (lun,1)
0099
0100            write(lun,35) 'HEADS HOME SWITCH OPEN',
0101           1 ' - DURING SPIN-UP'
0102
0103            else if (error_code .eq. '0a'x) then
0104
0105            call linchk (lun,1)
0106
0107            write(lun,35) 'HEADS HOME SWITCH WOULD NOT CLOSE',
0108           1 ' - DURING UNLOAD'
0109
0110            else if (error_code .eq. '0b'x) then
0111
0112            call linchk (lun,1)
0113
0114            write(lun,35) 'COVER OR LID LOCK OPENED',
```

```
0115            1 ' - DURING HEAD LOAD'
0116
0117            else if (error_code .eq. '10'x) then
0118
0119            write(lun,40) 'LID LOCK OPEN DURING SPIN-UP'
0120
0121            else if (error_code .eq. '11'x) then
0122
0123            write(lun,40) 'COVER OPEN DURING SPIN-UP'
0124
0125            else if (error_code .eq. '18'x) then
0126
0127            write(lun,40) 'SPIN-UP TIME TOO LONG'
0128
0129            else if (error_code .eq. '19'x) then
0130
0131            call linchk (lun,1)
0132
0133            write(lun,35) 'NO MOTOR SAMPLE CHANGE',
0134            1 ' - DURING SPIN-UP'
0135
0136            else if (error_code .eq. '20'x) then
0137
0138            write(lun,40) 'SPIN-UP TIME TOO SHORT'
0139
0140            else if (error_code .eq. '28'x) then
0141
0142            call linchk (lun,1)
0143
0144            write(lun,35) 'TOO MANY BAD SERVO SAMPLES',
0145            1 ' - DURING "RTZ"'
0146
0147            else if (error_code .eq. '29'x) then
0148
0149            call linchk (lun,1)
0150
0151            write(lun,35) 'TOO LONG TO AQUIRE GOOD SAMPLES',
0152            1 ' - ON HEAD LOAD'
0153
0154            else if (error_code .eq. '2a'x) then
0155
0156            call linchk (lun,1)
0157
0158            write(lun,35) 'HEAD LOAD SEEK DID NOT RETURN',
0159            1 ' - TO TRACK ZERO'
0160
0161            else if (error_code .eq. '2b'x) then
0162
0163            call linchk (lun,1)
0164
0165            write(lun,35) 'BAD SERVO SAMPLES WHEN BUMPING',
0166            1 ' - GUARD BAND ON HEAD LOAD'
0167
0168            else if (error_code .eq. '2c'x) then
0169
0170            call linchk (lun,1)
0171
```

```
0172            write(lun,35) 'BAD SERVO SAMPLES DURING "RTZ"',
0173            1 ' - ON HEAD LOAD'
0174
0175            else if (error_code .eq. '2d'x) then
0176
0177            write(lun,40) 'BAD SAMPLES DURING LINEAR MODE'
0178
0179            else if (error_code .eq. '2e'x) then
0180
0181            write(lun,40) 'OFF SPEED WHEN IN LINEAR STATE'
0182
0183            else if (error_code .eq. '30'x) then
0184
0185            call linchk (lun,1)
0186
0187            write(lun,35) 'VELOCITY TOO HIGH DURING "RTZ"',
0188            1 ' - ON HEAD LOAD'
0189
0190            else if (error_code .eq. '31'x) then
0191
0192            call linchk (lun,1)
0193
0194            write(lun,35) 'VELOCITY TOO LOW DURING "RTZ"',
0195            1 ' - ON HEAD LOAD'
0196
0197            else if (error_code .eq. '32'x) then
0198
0199            write(lun,40) 'BAD VELOCITY ON SEEK'
0200
0201            else if (error_code .eq. '38'x) then
0202
0203            call linchk (lun,1)
0204
0205            write(lun,35) 'TOO LONG TO GET ON TRACK',
0206            1 ' - IN LINEAR MODE'
0207
0208            else if (error_code .eq. '40'x) then
0209
0210            write(lun,40) 'RETRY ON SEEK, BAD SERVO SAMPLES'
0211
0212            else if (error_code .eq. '41'x) then
0213
0214            call linchk (lun,1)
0215
0216            write(lun,35) 'RETRY ON SEEK,',
0217            1 ' - BAD SERVO SAMPLES FAILED'
0218
0219            else if (error_code .eq. '42'x) then
0220
0221            call linchk (lun,1)
0222
0223            write(lun,35) 'RETRY ON SEEK,',
0224            1 ' - BAD GUARD BAND FLAGS'
0225
0226            else if (error_code .eq. '43'x) then
0227
0228            call linchk (lun,1)
```

```
0229
0230            write(lun,35) 'RETRY ON SEEK,',
0231            1 ' - BAD GUARD BAND FLAGS FAILED'
0232
0233            else if (error_code .eq. '44'x) then
0234
0235            write(lun,40) 'RETRY ON SEEK, SEEK TIMEOUT'
0236
0237            else if (error_code .eq. '45'x) then
0238
0239            call linchk (lun,1)
0240
0241            write(lun,35) 'RETRY ON SEEK,',
0242            1 ' - SEEK TIMEOUT FAILED'
0243
0244            else if (error_code .eq. '48'x) then
0245
0246            write(lun,40) 'LOST SERVO SAMPLES'
0247
0248            else if (error_code .eq. '50'x) then
0249
0250            call linchk (lun,1)
0251
0252            write(lun,35) 'TOO LONG FOR HEADS HOME SWITCH',
0253            1 ' - TO OPEN ON HEAD LOAD'
0254
0255            else if (error_code .eq. '81'x) then
0256
0257            write(lun,40) 'INITS AND TESTS, CONTROL PANEL'
0258
0259            else if (error_code .eq. '82'x) then
0260
0261            write(lun,40) 'CHECKS, PROCESSOR REGISTERS'
0262
0263            else if (error_code .eq. '83'x) then
0264
0265            write(lun,40) 'TESTS, RAM #0'
0266
0267            else if (error_code .eq. '84'x) then
0268
0269            write(lun,40) 'TESTS, ROM #0'
0270
0271            else if (error_code .eq. '85'x) then
0272
0273            write(lun,40) 'TESTS, ROM #1'
0274
0275            else if (error_code .eq. '86'x) then
0276
0277            write(lun,40) 'TESTS, ROM #2'
0278
0279            else if (error_code .eq. '87'x) then
0280
0281            write(lun,40) 'VERIFY ROM VERSION NUMBERS'
0282
0283            else if (error_code .eq. '88'x) then
0284
0285            write(lun,40) 'TESTS, "SDI" CLEAR INTERFACE BIT'
```

```
0286
0287            else if (error_code .eq. '89'x) then
0288
0289            call linchk (lun,1)
0290
0291            write(lun,35) 'TESTS,',
0292            1 ' - "SDI" CONTROL/STATUS REGISTER'
0293
0294            else if (error_code .eq. '8a'x) then
0295
0296            call linchk (lun,1)
0297
0298            write(lun,35) 'TESTS,',
0299            1 ' - FRONT PANEL AND SERIAL # ROM'
0300
0301            else if (error_code .eq. '8b'x) then
0302
0303            write(lun,40) 'INITS, "UART"'
0304
0305            else if (error_code .eq. '8c'x) then
0306
0307            write(lun,40) 'TESTS, ERROR REGISTERS'
0308
0309            else if (error_code .eq. '8d'x) then
0310
0311            write(lun,40) 'TESTS, "8155" TIMER'
0312
0313            else if (error_code .eq. '8e'x) then
0314
0315            write(lun,40) 'TESTS, SLAVE CONTROL PORT'
0316
0317            else if (error_code .eq. '8f'x) then
0318
0319            write(lun,40) 'SLAVE DIAGNOSTICS'
0320
0321            else if (error_code .eq. '90'x) then
0322
0323            call linchk (lun,2)
0324
0325            write(lun,45) '"CMD" AVAILABLE OR',
0326            1 ' - SLAVE ALREADY SET WHEN ISSUING',
0327            1 ' - A NEW SLAVE COMMAND'
0328      45    format(' ',t40,a,/,t40,a,/,t40,a)
0329
0330            else if (error_code .eq. '91'x) then
0331
0332            call linchk (lun,1)
0333
0334            write(lun,35) 'SLAVE COMMAND RECEIVED',
0335            1 ' - TIMEOUT ON OPCODE'
0336
0337            else if (error_code .eq. '92'x) then
0338
0339            write(lun,40) 'SLAVE ATTENTION TIMEOUT'
0340
0341            else if (error_code .eq. '93'x) then
0342
```

```
0343            write(lun,40) 'SLAVE DONE TIMEOUT, ''TOSTP'''
0344
0345            else if (error_code .eq. '94'x) then
0346
0347            call linchk (lun,1)
0348
0349            write(lun,35) 'SLAVE DONE TIMEOUT,',
0350            1 ' - STATUS (1MS)'
0351
0352            else if (error_code .eq. '95'x) then
0353
0354            write(lun,40) 'SLAVE DONE TIMEOUT'
0355
0356            else if (error_code .eq. '96'x) then
0357
0358            write(lun,40) 'SLAVE ATTENTION TIMEOUT'
0359
0360            else if (error_code .eq. '97'x) then
0361
0362            write(lun,40) 'SOLENOID RELEASE TIMEOUT'
0363
0364            else if (error_code .eq. '98'x) then
0365
0366            write(lun,40) 'WATCHDOG TIMER, MASTER INSANE'
0367
0368            else if (error_code .eq. '99'x) then
0369
0370            write(lun,40) 'WATCHDOG TIMER, SLAVE INSANE'
0371
0372            else if (error_code .eq. '9a'x) then
0373
0374            call linchk (lun,1)
0375
0376            write(lun,35) 'RUN SWITCH OR COVER INVALID',
0377            1 ' - AT SPIN-UP'
0378
0379            else if (error_code .eq. '9b'x) then
0380
0381            call linchk (lun,1)
0382
0383            write(lun,35) 'SPINDLE NOT READY DURING',
0384            1 ' - RECALIBRATE COMMAND'
0385
0386            else if (error_code .eq. '9c'x) then
0387
0388            call linchk (lun,1)
0389
0390            write(lun,35) '''SDI'' TRANSFER ERROR',
0391            1 ' - FROM ''DC703'''
0392
0393            else if (error_code .eq. '9d'x) then
0394
0395            call linchk (lun,1)
0396
0397            write(lun,35) 'WRITE OR FORMAT WHILE',
0398            1 ' - WRITE PROTECTED'
0399
```

```
0400            else if (error_code .eq. '9e'x) then
0401
0402            write(lun,40) 'FORMAT WHILE FORMAT DISABLED'
0403
0404            else if (error_code .eq. '9f'x) then
0405
0406            call linchk (lun,1)
0407
0408            write(lun,35) 'TRANSFER COMMAND',
0409           1 ' - WHILE DRIVE ERROR'
0410
0411            else if (error_code .eq. 'a0'x) then
0412
0413            write(lun,40) 'SECTOR READ/WRITE OVERRUN ERROR'
0414
0415            else if (error_code .eq. 'a1'x) then
0416
0417            write(lun,40) 'TRANSFER COMMAND WHILE R/W ERROR'
0418
0419            else if (errror_code .eq. 'a2'x) then
0420
0421            write(lun,40) '''SDI'' COMMAND CHECKSUM ERROR'
0422
0423            else if (error_code .eq. 'a3'x) then
0424
0425            write(lun,40) '''SDI'' FRAME ERROR'
0426
0427            else if (error_code .eq. 'a4'x) then
0428
0429            write(lun,40) '''SDI'' COMMAND PARITY ERROR'
0430
0431            else if (error_code .eq. 'a5'x) then
0432
0433            write(lun,40) '''SDI'' COMMAND OUT OF RANGE'
0434
0435            else if (error_code .eq. 'a6'x) then
0436
0437            write(lun,40) '''SDI'' COMMAND LENGTH ERROR'
0438
0439            else if (error_code .eq. 'a7'x) then
0440
0441            write(lun,40) '''SDI'' ERROR STATUS BYTE NON-ZERO'
0442
0443            else if (error_code .eq. 'a8'x) then
0444
0445            write(lun,40) 'SLAVE DONE TIMEOUT, SEEK COMMAND'
0446
0447            else if (error_code .eq. 'a9'x) then
0448
0449            call linchk (lun,1)
0450
0451            write(lun,35) 'SLAVE DONE TIMEOUT,',
0452           1 ' - ''RTZ'' COMMAND'
0453
0454            else if (error_code .eq. 'aa'x) then
0455
0456            call linchk (lun,1)
```

```
0457
0458            write(lun,35) 'SLAVE RESPONSE ERROR,',
0459            1 ' - "RTZ COMMAND'
0460
0461            else if (error_code .eq. 'ab'x) then
0462
0463            write(lun,40) 'SLAVE DONE TIMEOUT, FIRST PASS'
0464
0465            else if (error_code .eq. 'ac'x) then
0466
0467            call linchk (lun,2)
0468
0469            write(lun,45) 'COMMAND RECEIVED TIMEOUT DURING',
0470            1 ' - STOP OR ASCII PORT COMMAND,',
0471            1 ' - TIME = "TODGN2"'
0472
0473            else if (error_code .eq. 'ad'x) then
0474
0475            call linchk (lun,2)
0476
0477            write(lun,45) 'COMMAND RECEIVED TIMEOUT DURING',
0478            1 ' - STOP OR ASCII PORT COMMAND,',
0479            1 ' - TIME = 1MS'
0480
0481            else if (error_code .eq. 'ae'x) then
0482
0483            write(lun,40) 'DRIVE NOT ON-LINE'
0484
0485            else if (error_code .eq. 'af'x) then
0486
0487            write(lun,40) 'COMMAND BYTES 4 AND 5 NON-ZERO'
0488
0489            else if (error_code .eq. 'b0'x) then
0490
0491            write(lun,40) 'INVALID GROUP NUMBER'
0492
0493            else if (error_code .eq. 'b1'x) then
0494
0495            write(lun,40) 'INVALID HEAD SELECT NUMBER'
0496
0497            else if (error_code .eq. 'b2'x) then
0498
0499            write(lun,40) 'INVALID CYLINDER RANGE'
0500
0501            else if (error_code .eq. 'b3'x) then
0502
0503            write(lun,40) 'SPINDLE NOT READY'
0504
0505            else if (error_code .eq. 'b4'x) then
0506
0507            write(lun,40) 'CONTROLLER TIMEOUT = 0'
0508
0509            else if (error_code .eq. 'b5'x) then
0510
0511            write(lun,40) 'SPINDLE NOT READY'
0512
0513            else if (error_code .eq. 'b6'x) then
```

```
0514
0515            write(lun,40) 'ILLEGAL MEMORY REGION'
0516
0517            else if (error_code .eq. 'b7'x) then
0518
0519            write(lun,40) '"DD" BIT SET'
0520
0521            else if (error_code .eq. 'b8'x) then
0522
0523            write(lun,40) 'TRIED TO CLEAR HARD FAULT'
0524
0525            else if (error_code .eq. 'b9'x) then
0526
0527            write(lun,40) 'SLAVE DONE TIMEOUT, GET STATUS'
0528
0529            else if (error_code .eq. 'ba'x) then
0530
0531            write(lun,40) 'NON-ZERO LEVEL'
0532
0533            else if (error_code .eq. 'bb'x) then
0534
0535            write(lun,40) 'WRONG SUB-UNIT ERROR'
0536
0537            else if (error_code .eq. 'bc'x) then
0538
0539            call linchk (lun,1)
0540
0541            write(lun,35) 'DIAGNOSTIC COMMAND,',
0542           1 ' - MEMORY REGION "HI" NON-ZERO'
0543
0544            else if (error_code .eq. 'bd'x) then
0545
0546            call linchk (lun,1)
0547
0548            write(lun,35) 'DIAGNOSTIC COMMAND,',
0549           1 ' - INVALID PARAMETERS'
0550
0551            else if (error_code .eq. 'be'x) then
0552
0553            call linchk (lun,1)
0554
0555            write(lun,35) 'WRITE ENABLE',
0556           1 ' - TO WRITE PROTECTED DRIVE'
0557
0558            else if (error_code .eq. 'bf'x) then
0559
0560            write(lun,40) 'SETTING "S7" = 1'
0561
0562            else if (error_code .eq. 'c0'x) then
0563
0564            call linchk (lun,1)
0565
0566            write(lun,35) 'SLAVE DONE TIMEOUT,',
0567           1 ' - FROM GET STATUS COMMAND'
0568
0569            else if (error_code .eq. 'c1'x) then
0570
```

```
0571            call linchk (lun,1)
0572
0573            write(lun,35) 'ERROR RESPONSE,',
0574            1 ' - GET STATUS COMMAND'
0575
0576            else if (error_code .eq. 'c2'x) then
0577
0578            write(lun,40) 'INVALID GROUP NUMBER'
0579
0580            else if (error_code .eq. 'c3'x) then
0581
0582            write(lun,40) 'MICRO-CODE FAULT ERROR'
0583
0584            else if (error_code .eq. 'c4'x) then
0585
0586            write(lun,40) 'NO WATCHDOG INTERRUPT'
0587
0588            else if (error_code .eq. 'c5'x) then
0589
0590            call linchk (lun,1)
0591
0592            write(lun,35) 'SLAVE DONE TIMEOUT,',
0593            1 ' - ASCII PORT'
0594
0595            else if (error_code .eq. 'c6'x) then
0596
0597            call linchk (lun,1)
0598
0599            write(lun,35) 'SLAVE COMMAND ERROR,',
0600            1 ' - ASCII PORT'
0601            endif
0602
0603            return
0604            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	6358	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	3180	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	1224	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	10762	

ENTRY POINTS

Address	Type	Name
0-00000000		RA60_STATUS

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000008	I*4	CURRENT_CYLINDER	2-0000000C	I*4	CURRENT_HEAD
2-00000010	I*4	ERROR_CODE	2-00000014	R*4	ERRROR_CODE
AP-00000004@	L*1	LUN	2-00000000	I*4	PREVIOUS_CYLINDER
2-00000004	I*4	PREVIOUS_HEAD	AP-00000008@	I*4	SDI1
AP-0000000C@	I*4	SDI2	AP-00000010@	I*4	SDI3

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-00000BF9	10'	1-00000C09	15'	1-00000C19	20'	1-00000C29	25'	1-00000C39	30'	1-00000C4B	35'
1-00000C56	40'	1-00000C5D	45'								

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK

```
0001
0002
0003
0004            Subroutine RA80_STATUS (lun,sdi1,sdi2,sdi3)
0005
0006
0007            byte            lun
0008
0009            integer*4       sdi1
0010            integer*4       sdi2
0011            integer*4       sdi3
0012            integer*4       front_panel_fault_code
0013            integer*4       compress4
0014
0015
0016            front_panel_fault_code = lib$extzv (24,8,sdi3)
0017
0018            if (front_panel_fault_code .eq. '14'x) then
0019
0020            write(lun,55) 'R/W DIAGNOSTIC ERROR (SPIN-UP)'
0021     55     format(' ',t40,a)
0022
0023            else if (front_panel_fault_code .eq. '16'x) then
0024
0025            write(lun,55) 'SPINDLE MOTOR INTERLOCK'
0026
0027            else if (front_panel_fault_code .eq. '18'x) then
0028
0029            write(lun,55) 'SPIN-UP ERROR'
0030
0031            else if (front_panel_fault_code .eq. '1c'x) then
0032
0033            write(lun,55) 'R/W UNSAFE'
0034
0035            else if (front_panel_fault_code .eq. '1e'x) then
0036
0037            write(lun,55) 'SPINDLE MOTOR SPEED ERROR'
0038
0039            else if (front_panel_fault_code .eq. '34'x) then
0040
0041            write(lun,55) 'HDA OR SERVO BOARD OVER-TEMP'
0042
0043            else if (front_panel_fault_code .eq. '36'x) then
0044
0045            write(lun,55) 'INITIAL RECAL FAILURE'
0046
0047            else if (front_panel_fault_code .eq. '38'x) then
0048
0049            write(lun,55) 'DRIVE DISABLED BY "DD" BIT'
0050
0051            else if (front_panel_fault_code .eq. '3a'x) then
0052
0053            write(lun,55) 'SERVO DIAGNOSTIC ERROR (SPIN-UP)'
0054            else
0055
0056            write(lun,60) 'FRONT PANEL FAULT CODE, ',front_panel_fault_code,
0057           1 ' (HEX)'
```

```
0058     60       format(' ',t40,a,z2.2,a)
0059              endif
0060
0061              return
0062              end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	455	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	264	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	112	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	831	

ENTRY POINTS

Address	Type	Name
0-00000000		RA80_STATUS

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000004	I*4	COMPRESS4	2-00000000	I*4	FRONT_PANEL_FAULT_CODE
AP-00000004@	L*1	LUN	AP-00000008@	I*4	SDI1
AP-0000000C@	I*4	SDI2	AP-00000010@	I*4	SDI3

LABELS

Address	Label	Address	Label
1-000000F6	55'	1-000000FD	60'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
I*4	LIB$EXTZV

```
0001
0002
0003
0004          Subroutine RA8X_STATUS (lun,sdi1,sdi2,sdi3)
0005
0006
0007          byte              lun
0008
0009          integer*4         sdi1
0010          integer*4         sdi2
0011          integer*4         sdi3
0012          integer*4         compress4
0013
0014          logical*1         init_diagnostic_failure
0015
0016          integer*4         retry_count
0017          integer*4         previous_command
0018          integer*4         cylinder
0019          integer*4         current_group
0020          integer*4         led_display_code
0021
0022          character*33      v1ra8x_interface_error_status(3:5)
0023
0024          data              v1ra8x_interface_error_status(3)
0025          1 /'OVERRUN ERROR*'/
0026
0027          data              v1ra8x_interface_error_status(4)
0028          1 /'CONTROL R/T STATUS, PARITY ERROR*'7
0029
0030          data              v1ra8x_interface_error_status(5)
0031          1 /'CONTROL PULSE ERROR*'/
0032
0033          character*17      v2ra8x_interface_error_status(7:7)
0034
0035          data              v2ra8x_interface_error_status(7)
0036          1 /'DATA PULSE ERROR*'7
0037
0038
0039
0040
0041          if (lib$extzv (20,1,sdi1) .eq. 1) then
0042
0043          init_diagnostic_failure = .true.
0044          else
0045
0046          init_diagnostic_failure = .false.
0047          endif
0048
0049          retry_count = lib$extzv (0,8,sdi2)
0050
0051          if (.not. init_diagnostic_failure) then
0052
0053          if (retry_count .ne. 0) then
0054
0055          previous_command = lib$extzv (8,8,sdi2)
0056
0057          if (previous_command .eq. '8e'x) then
```



```
023
023
023
023
023
023
023
023
024
024
024
024
024
024
024
024
024
024
024
025
025
025
025
025
025
025
025
025
025
026
026
026
026
026
026
027
027
027
027
027
027
027
027
027
028
028
028
028
028
028
028
```

```
0058
0059            call linchk (lun,1)
0060
0061            write(lun,34) 'RECALIBRATE'
0062    34      format(' ',t40,a)
0063
0064            else if (previous_command .eq. '0a'x) then
0065
0066            call linchk (lun,1)
0067
0068            write(lun,34) 'SEEK'
0069            endif
0070            endif
0071            endif
0072
0073            call output (lun,sdi2,v1ra8x_interface_error_status,3,3,5,'0')
0074
0075            call output (lun,sdi2,v2ra8x_interface_error_status,7,7,7,'0')
0076
0077            if (
0078            1 .not. init_diagnostic_failure
0079            1 .and.
0080            1 retry_count .ne. 0
0081            1 .and.
0082            1 (previous_command .eq. '8e'x
0083            1 .or.
0084            1 previous_command .eq. '0a'x)
0085            1 ) then
0086
0087            cylinder = lib$extzv (24,16,sdi2)
0088
0089            call linchk (lun,1)
0090
0091            write(lun,40) cylinder
0092    40      format(' ',t40,'CYLINDER #',i<compress4 (cylinder)>,'.')
0093
0094            current_group = lib$extzv (8,8,sdi3)
0095
0096            call linchk (lun,1)
0097
0098            write(lun,45) current_group
0099    45      format(' ',t40,'CURRENT GROUP #',i<compress4 (current_group)>,'.')
0100            endif
0101
0102            led_display_code = lib$extzv (16,8,sdi3)
0103
0104            call linchk (lun,1)
0105
0106            write(lun,50) led_display_code
0107    50      format(' ',t40,''LED'' DISPLAY CODE, ',z2.2,' (HEX)')
0108
0109            return
0110            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	515	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	158	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	400	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	1073	

ENTRY POINTS

Address	Type	Name
0-00000000		RA8X_STATUS

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000084	I*4	CURRENT_GROUP	2-00000080	I*4	CYLINDER
2-00000074	L*1	INIT_DIAGNOSTIC_FAILURE	2-00000088	I*4	LED_DISPLAY_CODE
AP-00000004@	L*1	LUN	2-0000007C	I*4	PREVIOUS_COMMAND
2-00000078	I*4	RETRY_COUNT	AP-00000008@	I*4	SDI1
AP-0000000C@	I*4	SDI2	AP-00000010@	I*4	SDI3

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	CHAR	V1RA8X_INTERFACE_ERROR_STATUS	99	(3:5)
2-00000063	CHAR	V2RA8X_INTERFACE_ERROR_STATUS	17	(7:7)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label
1-00000035	34'	1-0000003C	40'	1-00000057	45'	1-00000077	50'

FUNCTIONS AND SUBROUTINES REFERENCED

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

MSL

```
0001
0002
0003
0004          Subroutine GENERIC_SDI (lun,sdi1,sdi2,sdi3)
0005
0006
0007          byte              lun
0008
0009          integer*4         sdi1
0010          integer*4         sdi2
0011          integer*4         sdi3
0012          integer*4         compress4
0013          integer*4         c_flags
0014          integer*4         retry_count
0015
0016          character*19      v1generic_sdi_status(0:1)
0017
0018          data              v1generic_sdi_status(0)
0019          1 /'RUN/STOP SWITCH IN*'/
0020
0021          data              v1generic_sdi_status(1)
0022          1 /'PORT SWITCH IN*'/
0023
0024          character*33      v2generic_sdi_status(3:15)
0025
0026          data              v2generic_sdi_status(3)
0027          1 /'LOG INFORMATION IN EXTENDED AREA*'/
0028
0029          data              v2generic_sdi_status(4)
0030          1 /'SPINDLE READY*'/
0031
0032          data              v2generic_sdi_status(5)
0033          1 /'DIAGNOSTIC LOAD REQUEST*'7
0034
0035          data              v2generic_sdi_status(6)
0036          1 /'RE-ADJUSTMENT REQUESTED*'7
0037
0038          data              v2generic_sdi_status(7)
0039          1 /'DRIVE UNAVAILABLE, BUT ONLINE*'/
0040
0041          data              v2generic_sdi_status(8)
0042          1 /'576 BYTE SECTOR*'/
0043
0044          data              v2generic_sdi_status(9)
0045          1 /'DIAG. CYLINDER ACCESS ENABLED*'/
0046
0047          data              v2generic_sdi_status(10)
0048          1 /'FORMATTING OPERATIONS ENABLED*'/
0049
0050          data              v2generic_sdi_status(11)
0051          1 /'FAULT*'/
0052
0053          data              v2generic_sdi_status(12)
0054          1 /'WRITE PROTECT, UNIT #0.*'7
0055
0056          data              v2generic_sdi_status(13)
0057          1 /'WRITE PROTECT, UNIT #1.*'7
```

```
0058
0059            data                v2generic_sdi_status(14)
0060            1 /'WRITE PROTECT, UNIT #2.*'7
0061
0062            data                v2generic_sdi_status(15)
0063            1 /'WRITE PROTECT, UNIT #3.*'7
0064
0065            character*34    v3generic_sdi_status(19:23)
0066
0067            data                v3generic_sdi_status(19)
0068            1 /'WRITE LOCK ERROR*'/
0069
0070            data                v3generic_sdi_status(20)
0071            1 /'INITIALIZATION DIAGNOSTIC FAILURE*'/
0072
0073            data                v3generic_sdi_status(21)
0074            1 /'LEVEL 2 PROTOCOL ERROR*'/
0075
0076            data                v3generic_sdi_status(22)
0077            1 /'TRANSMISSION ERROR*'/
0078
0079            data                v3generic_sdi_status(23)
0080            1 /'DRIVE ERROR*'/
0081
0082
0083
0084            call output (lun,sdi1,v1generic_sdi_status,0,0,1,'0')
0085
0086            call output (lun,sdi1,v2generic_sdi_status,3,3,15,'0')
0087
0088            call output (lun,sdi1,v3generic_sdi_status,19,19,23,'0')
0089
0090            c_flags = lib$extzv(24,4,sdi1)
0091
0092            if (c_flags .eq. 8) then
0093
0094            call linchk (lun,1)
0095
0096            write(lun,15) ''DIAGNOSTICS''
0097    15      format(' ',t40,'DRIVE-OFFLINE, ',a)
0098
0099            else if (c_flags .eq. 9) then
0100
0101            call linchk (lun,1)
0102
0103            write(lun,15) 'DUPLICATE UNIT IDENT'
0104            endif
0105
0106            do 25,i = 28,31
0107
0108            if (lib$extzv(i,1,sdi1) .eq. 1) then
0109
0110            call linchk (lun,1)
0111
0112            write(lun,20) 'SEND ''MSCP'' AVAILABLE SUB-UNIT #',28-i,'.'
0113    20      format(' ',t40,a,i<compress4 (28-i)>,a)
0114            endif
```

```
0115
0116     25       continue
0117
0118              retry_count = lib$extzv (0,8,sdi2)
0119
0120              if (lib$extzv(20,1,sdi1) .eq. 0) then
0121
0122              if (retry_count .ne. 0) then
0123
0124              call linchk (lun,1)
0125
0126              write(lun,30) 'RETRY COUNT ',retry_count
0127     30       format(' ',t40,a,i<compress4 (retry_count)>,'.')
0128              endif
0129              else
0130
0131              write(lun,35) 'FAILURE CODE ',retry_count,' (HEX)'
0132     35       format(' ',t40,a,z2.2,a)
0133              endif
0134
0135              return
0136              end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	505	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	204	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	1000	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	1709	

ENTRY POINTS

Address	Type	Name
0-00000000		GENERIC_SDI

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name	Address	Type	Name
2-00000280	I*4	C_FLAGS	2-00000288	I*4	I	AP-00000004@	L*1	LUN	2-00000284	I*4	RETRY_COUNT
AP-00000008@	I*4	SDI1	AP-0000000C@	I*4	SDI2	AP-00000010@	I*4	SDI3			

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	CHAR	V1GENERIC_SDI_STATUS	38	(0:1)
2-00000026	CHAR	V2GENERIC_SDI_STATUS	429	(3:15)
2-000001D3	CHAR	V3GENERIC_SDI_STATUS	170	(19:23)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-0000008B	15'	1-000000A3	20'	**	25	1-000000B1	30'	1-000000C1	35'

FUNCTIONS AND SUBROUTINES REFERENCED

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

MSL

057
057
057
057
057
057
057
057
058
058
058
058
058
058
058
058
058
058
059
059
059
059
059
059
059
059
059
059
059
060
060
060
060
060
060
060
060
060
061
061
061
061
061
061
061
061
061
062
062
062
062
062
062
062
062
062
062

```
0001
0002
0003
0004          Subroutine RA81_STATUS (lun,sdi1,sdi2,sdi3)
0005
0006
0007          byte              lun
0008
0009          integer*4         sdi1
0010          integer*4         sdi2
0011          integer*4         sdi3
0012          integer*4         front_panel_fault_code
0013          integer*4         compress4
0014
0015
0016          front_panel_fault_code = lib$extzv (24,8,sdi3)
0017
0018          if (front_panel_fault_code .eq. '18'x) then
0019
0020          write(lun,55) 'SPIN-UP ERROR'
0021     55   format(' ',t40,a)
0022
0023          else if (front_panel_fault_code .eq. '14'x) then
0024
0025          write(lun,55) 'UPROC FAULT'
0026
0027          else if (front_panel_fault_code .eq. '1c'x) then
0028
0029          write(lun,55) 'SDI ERROR'
0030
0031          else if (front_panel_fault_code .eq. '12'x) then
0032
0033          write(lun,55) 'MASTER/SLAVE ERROR'
0034
0035          else if (front_panel_fault_code .eq. '1a'x) then
0036
0037          write(lun,55) 'SERVO FINE POSITIONING ERROR'
0038
0039          else if (front_panel_fault_code .eq. '16'x) then
0040
0041          write(lun,55) 'SERVO COARSE POSITIONING ERROR'
0042
0043          else if (front_panel_fault_code .eq. '1e'x) then
0044
0045          write(lun,55) 'SPINDLE MOTOR INTERLOCK'
0046
0047          else if (front_panel_fault_code .eq. '30'x) then
0048
0049          write(lun,55) 'HDA OR SERVO BOARD OVER-TEMP'
0050
0051          else if (front_panel_fault_code .eq. '38'x) then
0052
0053          write(lun,55) 'R/W UNSAFE'
0054
0055          else if (front_panel_fault_code .eq. '34'x) then
0056
0057          write(lun,55) 'R/W COMMAND ERROR'
```

```
0058
0059            else if (front_panel_fault_code .eq. '3c'x) then
0060
0061            write(lun,55) 'SERVO/HDA INTERLOCK'
0062
0063            else if (front_panel_fault_code .eq. '32'x) then
0064
0065            write(lun,55) 'SERVO/MICRO INTERLOCK'
0066
0067            else if (front_panel_fault_code .eq. '3a'x) then
0068
0069            write(lun,55) 'RW/MICRO INTERLOCK'
0070
0071            else if (front_panel_fault_code .eq. '36'x) then
0072
0073            write(lun,55) 'CTL PAN/MICRO INTERLOCK'
0074
0075            else if (front_panel_fault_code .eq. '3e'x) then
0076
0077            write(lun,55) 'PER/MICRO INTERLOCK'
0078
0079            else if (front_panel_fault_code .eq. '11'x) then
0080
0081            write(lun,55) 'INDEX ERROR'
0082
0083            else if (front_panel_fault_code .eq. '19'x) then
0084
0085            write(lun,55) 'WRITE AND BAD IMBEDDED'
0086
0087            else if (front_panel_fault_code .eq. '15'x) then
0088
0089            write(lun,55) 'DRIVE DISABLED BY "DD" BIT'
0090
0091            else if (front_panel_fault_code .eq. '1d'x) then
0092
0093            write(lun,55) 'WRITE AND WRITE PROTECT'
0094
0095            else if (front_panel_fault_code .eq. '13'x) then
0096
0097            write(lun,55) 'SERVO DIAGNOSTICS FAILED'
0098
0099            else if (front_panel_fault_code .eq. '1b'x) then
0100
0101            write(lun,55) 'R/W DIAGNOSTICS FAILED'
0102
0103            else if (front_panel_fault_code .eq. '17'x) then
0104
0105            write(lun,55) 'IDLE LOOP DIAGNOSTICS FAILED'
0106
0107            else if (front_panel_fault_code .eq. '3f'x) then
0108
0109            write(lun,55) 'HARDCORE FAILURE'
0110            else
0111
0112            write(lun,60) 'FRONT PANEL FAULT CODE, ',front_panel_fault_code,
0113           1 ' (HEX)'
0114    60      format(' ',t40,a,z2.2,a)
```

```
0115            endif
0116
0117            return
0118            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	1055	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	515	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	224	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
	Total Space Allocated	1794	

ENTRY POINTS

Address	Type	Name
0-00000000		RA81_STATUS

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000004	I*4	COMPRESS4	2-00000000	I*4	FRONT_PANEL_FAULT_CODE
AP-00000004@	L*1	LUN	AP-00000008@	I*4	SDI1
AP-0000000C@	I*4	SDI2	AP-00000010@	I*4	SDI3

LABELS

Address	Label	Address	Label
1-000001F1	55'	1-000001F8	60'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name
I*4	LIB$EXTZV

```
0001
0002
0003
0004            Subroutine MSLG$K_SML_DSK_AZTEC (lun)
0005
0006
0007            include 'src$:msghdr.for /nolist'
0066
0067
0068            byte            lun
0069
0070            integer*4       mslg$l_cmd_ref
0071
0072            integer*2       mslg$w_event
0073            integer*2       full_error_code
0074            integer*2       front_panel
0075
0076            byte            l5b
0077            byte            pl5b
0078
0079            integer*2       track_in_error
0080            integer*2       servo_state
0081            integer*2       spindle_state
0082            integer*2       distance_travelled
0083
0084            integer*4       last_header_read
0085
0086            integer*2       desired_track
0087            integer*2       starting_track
0088            integer*2       dsts
0089            integer*2       track_error_value
0090
0091            byte            sel
0092            byte            sector
0093
0094            integer*4       compress4
0095            integer*4       lib$extzv
0096            integer*4       selected_head
0097            integer*4       base_unit
0098
0099            logical*1       drive_positioner_error
0100
0101            equivalence     (emb(38),mslg$l_cmd_ref)
0102            equivalence     (emb(48),mslg$w_event)
0103            equivalence     (emb(78),full_error_code)
0104            equivalence     (emb(80),front_panel,l5b)
0105            equivalence     (emb(81),pl5b)
0106            equivalence     (emb(82),track_in_error,servo_state)
0107            equivalence     (emb(84),spindle_state,distance_travelled)
0108            equivalence     (emb(86),last_header_read,desired_track)
0109            equivalence     (emb(88),starting_track)
0110            equivalence     (emb(90),dsts,track_error_value)
0111            equivalence     (emb(92),sel)
0112            equivalence     (emb(93),sector)
0113
0114            character*25    vlaztec_dsts(0:6)
0115
```

```
0116            data                v1aztec_dsts(0)
0117            1 /'DISABLE WRITE*'/
0118
0119            data                v1aztec_dsts(1)
0120            1 /'CARTRIDGE OR DOOR NOT OK*'/
0121
0122            data                v1aztec_dsts(2)
0123            1 /'OVER TEMPERATURE*'/
0124
0125            data                v1aztec_dsts(3)
0126            1 /'HEADS HOME*'/
0127
0128            data                v1aztec_dsts(4)
0129            1 /'SPINDLE STOPPED*'/
0130
0131            data                v1aztec_dsts(5)
0132            1 /'SPEED ERROR*'/
0133
0134            data                v1aztec_dsts(6)
0135            1 /'SERVO TIMEOUT*'/
0136
0137
0138            character*27        v2aztec_dsts(15:15)
0139
0140            data                v2aztec_dsts(15)
0141            1 /'FRONT PANEL SERIAL DATA IN*'/
0142
0143
0144            character*31        v1aztec_sel(0:4)
0145
0146            data                v1aztec_sel(0)
0147            1 /'PEAK DETECTED READ THRESHOLDS*'/
0148
0149            data                v1aztec_sel(1)
0150            1 /'EARLY-LATE INTEGRALS*'/
0151
0152            data                v1aztec_sel(2)
0153            1 /'DAC MODE*'/
0154
0155            data                v1aztec_sel(3)
0156            1 /'TACH. VELOCITY FEEDBACK TO A/D*'/
0157
0158            data                v1aztec_sel(4)
0159            1 /'ACTUATOR CURRENT TO A/D*'/
0160
0161
0162            character*21        v2aztec_sel(7:7)
0163
0164            data                v2aztec_sel(7)
0165            1 /'START A/D CONVERSION*'/
0166
0167
0168            character*6         v1front_panel(0:0)
0169
0170            data                v1front_panel(0)
0171            1 /'FAULT*'/
0172
```

```
0173
0174            character*26    v2front_panel(8:10)
0175
0176            data            v2front_panel(8)
0177            1 /'WRITE PROTECT, FIXED*'/
0178
0179            data            v2front_panel(9)
0180            1 /'WRITE PROTECT, REMOVEABLE*'/
0181
0182            data            v2front_panel(10)
0183            1 /'RUN*'/
0184
0185
0186
0187            drive_positioner_error = .false.
0188
0189            if (
0190            1 lib$extzv(0,5,mslg$w_event) .eq. 11
0191            1 .and.
0192            1 lib$extzv(5,11,mslg$w_event) .eq. 3
0193            1 ) then
0194
0195            drive_positioner_error = .true.
0196            endif
0197
0198            call linchk (lun,2)
0199
0200            write(lun,15) 'FULL ERROR CODE',full_error_code
0201    15      format(' ',t8,a,t28,z4.4)
0202
0203            if (lib$extzv(0,5,full_error_code) .eq. '13'o) then
0204
0205            minor_code = lib$extzv(5,11,full_error_code)
0206
0207            if (minor_code .eq. '10'o) then
0208
0209            write(lun,35) 'SEEK INTO GUARDBAND'
0210
0211            else if (minor_code .eq. '20'o) then
0212
0213            write(lun,35) 'NEGATIVE TRACK DIFFERENCE'
0214
0215            else if (minor_code .eq. '30'o) then
0216
0217            write(lun,35) 'DISTANCE TRAVELLED'
0218
0219            else if (minor_code .eq. '40'o) then
0220
0221            write(lun,35) 'NOT ONE TRACK AWAY'
0222
0223            else if (minor_code .eq. '50'o) then
0224
0225            write(lun,35) 'NOT ZERO TRACKS AWAY'
0226
0227            else if (minor_code .eq. '60'o) then
0228
0229            write(lun,35) 'DRIVE OFF TRACK'
```

```
0230
0231            else if (minor_code .eq. '70'o) then
0232
0233            write(lun,35) 'SERVO SAMPLE PARITY ERROR'
0234
0235            else if (minor_code .eq. '100'o) then
0236
0237            write(lun,35) 'HEAD SWITCH DEAD BAND'
0238
0239            else if (minor_code .eq. '120'o) then
0240
0241            write(lun,35) 'SERVO SAMPLE ERROR'
0242
0243            else if (minor_code .eq. '130'o) then
0244
0245            write(lun,35) 'A/D CONVERSION ERROR'
0246
0247            else if (minor_code .eq. '140'o) then
0248
0249            write(lun,35) 'HEAD SWITCH SSR TIMEOUT'
0250
0251            else if (minor_code .eq. '150'o) then
0252
0253            write(lun,35) 'HEAD SWITCH SSE ERROR'
0254
0255            else if (minor_code .eq. '160'o) then
0256
0257            write(lun,35) 'HEAD SWITCH L5B ERROR'
0258
0259            else if (minor_code .eq. '170'o) then
0260
0261            write(lun,35) 'ANL OR SCB.KT SERVO TIMEOUT'
0262
0263            else if (minor_code .eq. '11'o) then
0264
0265            write(lun,35) 'DSTS DISABLE WRITE ERROR'
0266
0267            else if (minor_code .eq. '21'o) then
0268
0269            write(lun,35) 'DSTS ANL SERVO TIMEOUT'
0270
0271            else if (minor_code .eq. '31'o) then
0272
0273            write(lun,35) 'D PROCESSOR SSR TIMEOUT'
0274
0275            else if (minor_code .eq. '41'o) then
0276
0277            write(lun,35) 'D PROCESSOR FOUND SSE'
0278
0279            else if (minor_code .eq. '51'o) then
0280
0281            write(lun,35) 'D PROCESSOR INDEX TIMEOUT'
0282
0283            else if (minor_code .eq. '61'o) then
0284
0285            write(lun,35) 'SERDES PARALLEL READY TIMEOUT'
0286
```

```
0287            else if (minor_code .eq. '71'o) then
0288
0289            write(lun,35) 'RESIDUE READY TIMEOUT'
0290
0291            else if (minor_code .eq. '101'o) then
0292
0293            write(lun,35) 'HEADER WORD RATE CLOCK TIMEOUT'
0294
0295            else if (minor_code .eq. '111'o) then
0296
0297            write(lun,35) 'DATA WORD RATE CLOCK TIMEOUT'
0298
0299            else if (minor_code .eq. '121'o) then
0300
0301            write(lun,35) 'REPLACE FOUND POSITIONER ERROR'
0302
0303            else if (minor_code .eq. '131'o) then
0304
0305            write(lun,35) 'ATTEMPT TO REVECTOR NON-LBN'
0306
0307            else if (minor_code .eq. '141'o) then
0308
0309            write(lun,35) 'RCT SEEK, READ OR EDC FAILED'
0310
0311            else if (minor_code .eq. '151'o) then
0312
0313            write(lun,35) 'RBN NOT FOUND IN RCT'
0314
0315            else if (minor_code .eq. '161'o) then
0316
0317            write(lun,35) 'SEEK TO RBN FAILED'
0318
0319            else if (minor_code .eq. '171'o) then
0320
0321            write(lun,35) 'SEEK BACK TO ORIGONAL FAILED'
0322
0323            else if (minor_code .eq. '231'o) then
0324
0325            write(lun,35) 'DRIVE ACLO ASSERTED'
0326
0327            else if (minor_code .eq. '241'o) then
0328
0329            write(lun,35) 'DRIVE OVER TEMPERATURE'
0330
0331            else if (minor_code .eq. '251'o) then
0332
0333            write(lun,35) 'DUPLICATE UNIT DETECTED'
0334
0335            else if (minor_code .eq. '261'o) then
0336
0337            write(lun,35) 'FRONT PANEL LOOP-BACK FAILED'
0338
0339            else if (minor_code .eq. '271'o) then
0340
0341            write(lun,35) 'WRONG STOPPED STATE (DS.STP)'
0342
0343            else if (minor_code .eq. '301'o) then
```



```
011
011
011
011
012
012
012
012
012
012
012
012
012
013
013
013
013
013
013
013
013
013
014
014
014
014
014
014
```

```
0344
0345            write(lun,35) 'SPIN DOWN TIMEOUT (DS.STP)'
0346
0347            else if (minor_code .eq. '311'o) then
0348
0349            write(lun,35) 'WRONG SPEED STATE (DS.STE)'
0350
0351            else if (minor_code .eq. '321'o) then
0352
0353            write(lun,35) 'SPIN UP TIMEOUT (DS.STE)'
0354
0355            else if (minor_code .eq. '331'o) then
0356
0357            write(lun,35) 'HEADS NOT OVER DISK (DS.HDH)'
0358
0359            else if (minor_code .eq. '341'o) then
0360
0361            write(lun,35) 'HEADS HOME TIMEOUT (DS.HDH)'
0362
0363            else if (minor_code .eq. '351'o) then
0364
0365            write(lun,35) 'HEADS HOME NEVER WENT AWAY'
0366
0367            else if (minor_code .eq. '361'o) then
0368
0369            write(lun,35) 'UNSAFE ATTEMPT TO UNLOCK DOOR'
0370
0371            else if (minor_code .eq. '371'o) then
0372
0373            write(lun,35) 'CARTRIDGE OR DOOR ERROR (DS.COD)'
0374
0375            else if (minor_code .eq. '401'o) then
0376
0377            write(lun,35) 'ARCS SEEK HARD ERROR'
0378
0379            else if (minor_code .eq. '411'o) then
0380
0381            write(lun,35) 'ARCS SECTOR HARD ERROR'
0382
0383            else if (minor_code .eq. '421'o) then
0384
0385            write(lun,35) 'ARCS DETENT HARD ERROR'
0386
0387            else if (minor_code .eq. '431'o) then
0388
0389            write(lun,35) 'SERVO ERROR RATE TOO HIGH'
0390
0391            else if (minor_code .eq. '441'o) then
0392
0393            write(lun,35) 'SPINDLE MACH. FAILED TO INIT SERVO'
0394
0395            else if (minor_code .eq. '501'o) then
0396
0397            write(lun,35) 'TPD DIAGNOSTICS FAILED'
0398
0399            else if (minor_code .eq. '511'o) then
0400
```

```
0401            write(lun,35) 'TPD DATA BUS CHECK FAILED'
0402
0403            else if (minor_code .eq. '521'o) then
0404
0405            write(lun,35) 'A/D OFFSET/TIMING FAILED'
0406
0407            else if (minor_code .eq. '531'o) then
0408
0409            write(lun,35) 'D/A OFFSET/TIMING FAILED'
0410
0411            else if (minor_code .eq. '541'o) then
0412
0413            write(lun,35) 'A/D - D/A LINEARITY FAILED'
0414
0415            else if (minor_code .eq. '551'o) then
0416
0417            write(lun,35) 'INTEGRATOR OFFSET FAILED'
0418
0419            else if (minor_code .eq. '561'o) then
0420
0421            write(lun,35) 'DRIVE CARD FAILED INIT TEST'
0422
0423            else if (minor_code .eq. '571'o) then
0424
0425            write(lun,35) 'DRIVE CABLE NOT IN PLACE'
0426            else
0427
0428            write(lun,17) 'ERROR CODE #',minor_code
0429     17     format(' ',t40,a,o5.5,' (OCTAL)')
0430            endif
0431            endif
0432
0433     c
0434     c      other than drive, positioner error
0435     c
0436
0437            if (.not. drive_positioner_error) then
0438
0439            call linchk (lun,1)
0440
0441            write(lun,20) 'FRONT PANEL',front_panel
0442     20     format(' ',t8,a,t28,z4.4)
0443
0444            call output (lun,front_panel,v1front_panel,0,0,0,'1')
0445
0446            base_unit = lib$extzv(1,7,front_panel) * 2
0447
0448            call linchk (lun,1)
0449
0450            write(lun,22) 'BASE UNIT #',base_unit
0451     22     format(' ',t40,a,i<compress4 (base_unit)>,'.')
0452
0453            call output (lun,front_panel,v2front_panel,8,8,10,'1')
0454
0455            call linchk (lun,2)
0456
0457            write(lun,25) 'TRACK IN ERROR',track_in_error,
```

```
0458            1 'TRACK #',lib$extzv(0,16,track_in_error),'. AT ERROR'
0459     25     format(' ',t8,a,t28,z4.4,/,
0460            1 t40,a,i<compress4 (lib$extzv(0,16,track_in_error))>,a)
0461
0462            call linchk (lun,2)
0463
0464            write(lun,30) 'SPINDLE STATE',spindle_state
0465     30     format(' ',t8,a,t28,z4.4)
0466
0467            if (spindle_state .eq. 0) then
0468
0469            write(lun,35) 'ONLINE'
0470     35     format(' ',t40,a)
0471
0472            else if (spindle_state .eq. 2) then
0473
0474            write(lun,35) 'I/O RUNDOWN'
0475
0476            else if (spindle_state .eq. 3) then
0477
0478            write(lun,35) 'BEGIN UNLOAD'
0479            else if (spindle_state .eq. 4) then
0480
0481            write(lun,35) 'HEADS HOME WAIT'
0482
0483            else if (spindle_state .eq. 5) then
0484
0485            write(lun,35) 'END UNLOAD'
0486
0487            else if (spindle_state .eq. 6) then
0488
0489            write(lun,35) 'SPIN DOWN WAIT'
0490
0491            else if (spindle_state .eq. 7) then
0492
0493            write(lun,35) 'STOPPED'
0494
0495            else if (spindle_state .eq. 8) then
0496
0497            write(lun,35) 'CHECK DOOR'
0498
0499            else if (spindle_state .eq. 9) then
0500
0501            write(lun,35) 'SPIN UP WAIT'
0502
0503            else if (spindle_state .eq. 10) then
0504
0505            write(lun,35) 'PURGE WAIT'
0506
0507            else if (spindle_state .eq. 11) then
0508
0509            write(lun,35) 'BEGIN LOAD'
0510
0511            else if (spindle_state .eq. 12) then
0512
0513            write(lun,35) 'LOAD WAIT'
0514
```

**F

```
0515            else if (spindle_state .eq. 13) then
0516
0517            write(lun,35) 'INIT SERVO'
0518
0519            else if (spindle_state .eq. 14) then
0520
0521            write(lun,35) 'OFFLINE'
0522
0523            else if (spindle_state .eq. 15) then
0524
0525            write(lun,35) 'INITIAL SPIN-UP'
0526            else
0527
0528            write(lun,40) lib$extzv(0,16,spindle_state)
0529     40     format(' ',t40,'ERROR CODE #',
0530            1 i<compress4 (lib$extzv(0,16,spindle_state))>,'.')
0531            endif
0532
0533            call linchk (lun,1)
0534
0535            write(lun,45) 'HEADER/LBN',last_header_read
0536     45     format(' ',t8,a,t24,z8.8)
0537
0538            last_header_read = lib$extzv(0,27,last_header_read)
0539
0540            if (mslg$l_cmd_ref .ne. 0) then
0541
0542            call linchk (lun,1)
0543
0544            write(lun,50) last_header_read
0545     50     format(' ',t40,'LOGICAL BLOCK #',i<compress4 (last_header_read)>,'.')
0546            else
0547
0548            call linchk (lun,1)
0549
0550            if (lib$extzv(27,1,last_header_read) .eq. 1) then
0551
0552            write(lun,52) 'FIXED MEDIA'
0553     52     format(' ',t40,a)
0554            else
0555
0556            write(lun,52) 'REMOVEABLE MEDIA'
0557            endif
0558
0559            call disk_header (lun,last_header_read)
0560            endif
0561
0562            call linchk (lun,1)
0563
0564            write(lun,55) 'DSTS',dsts
0565     55     format(' ',t8,a,t28,z4.4)
0566
0567            call output (lun,dsts,v1aztec_dsts,0,0,6,'0')
0568
0569            call output (lun,dsts,v2aztec_dsts,15,15,15,'0')
0570            else
0571
```

```
0572    c
0573    c       drive, positioner error
0574    c
0575
0576            call linchk (lun,2)
0577
0578            write(lun,60) 'L5B',l5b,'PL5B',pl5b
0579    60      format(' ',t8,a,t30,z2.2,/,
0580           1 t8,a,t30,z2.2)
0581
0582            call linchk (lun,2)
0583
0584            write(lun,65) 'SERVO STATE',servo_state
0585    65      format(' ',t8,a,t28,z4.4)
0586
0587            if (servo_state .eq. 0) then
0588
0589            write(lun,70) 'FAST SEEK'
0590    70      format(' ',t40,a,:i<compress4 (lib$extzv(0,16,servo_state))>,:a)
0591
0592            else if (servo_state .eq. 1) then
0593
0594            write(lun,70) 'MEDIUM SEEK'
0595
0596            else if (servo_state .eq. 2) then
0597
0598            write(lun,70) 'SLOW SEEK'
0599
0600            else if (servo_state .eq. 3) THEN
0601
0602            write(lun,70) 'ONE TRACK AWAY'
0603
0604            else if (servo_state .eq. 4) then
0605
0606            write(lun,70) 'ZERO TRACKS AWAY'
0607
0608            else if (servo_state .eq. 5) then
0609
0610            write(lun,70) 'DETENT'
0611
0612            else if (servo_state .eq. 6) then
0613
0614            write(lun,70) 'RTZ IN PROGRESS'
0615
0616            else if (servo_state .eq. 7) then
0617
0618            write(lun,70) 'ERROR STATE'
0619
0620            else if (servo_state .eq. 8) then
0621
0622            write(lun,70) 'FAKE SAMPLES (HEAD LOAD/UNLOAD'
0623            else
0624
0625            write(lun,70) 'STATE #',lib$extzv(0,16,servo_state),'.'
0626            endif
0627
0628            call linchk (lun,2)
```

```
0629
0630            write(lun,75) 'DISTANCE TRAVELLED',
0631            1 (lib$extzv(0,16,distance_travelled),i = 1,2),
0632            1 ' TRACK(S)'
0633     75     format(' ',t8,a,t28,z4.4,/,
0634            1 t40,i<compress4 ([lib$extzv(0,16,distance_travelled))>,a)
0635
0636            call linchk (lun,2)
0637
0638            write(lun,80) 'DESIRED TRACK',desired_track,
0639            1 'TRACK #',lib$extzv(0,16,desired_track),'.'
0640     80     format(' ',t8,a,t28,z4.4,/,
0641            1 t40,a,i<compress4 (lib$extzv(0,16,desired_track))>,a)
0642
0643            call linchk (lun,2)
0644
0645            write(lun,82) 'STARTING TRACK',starting_track,
0646            1 'TRACK #',lib$extzv(0,16,starting_track),'.'
0647     82     format(' ',t8,a,t28,z4.4,/,
0648            1 t40,a,i<compress4 (lib$extzv(0,16,starting_track))>,a)
0649
0650            call linchk (lun,1)
0651
0652            write(lun,84) 'TRACK ERROR VALUE',track_error_value
0653     84     format(' ',t8,a,t28,z4.4)
0654            endif
0655
0656            call linchk (lun,1)
0657
0658            write(lun,85) 'SEL',sel
0659     85     format(' ',t8,a,t30,z2.2)
0660
0661            call output (lun,sel,v1aztec_sel,0,0,4,'1')
0662
0663            selected_head = lib$extzv(5,2,sel)
0664
0665            call linchk (lun,1)
0666
0667            if (selected_head .eq. 0) then
0668
0669            write(lun,90) selected_head,'(REMOVEABLE, TOP)'
0670     90     format(' ',t40,'HEAD #',i<compress4 (selected_head)>,'. ',a)
0671
0672            else if (selected_head .eq. 1) then
0673
0674            write(lun,90) selected_head,'(REMOVEABLE, BOTTOM)'
0675
0676            else if (selected_head .eq. 2) then
0677
0678            write(lun,90) selected_head,'(FIXED, TOP)'
0679
0680            else if (selected_head .eq. 3) then
0681
0682            write(lun,90) selected_head,'(FIXED, BOTTOM)'
0683            endif
0684
0685            call output (lun,sel,v2aztec_sel,7,7,7,'1')
```

```
0686
0687              call linchk (lun,2)
0688
0689              write(lun,95) 'SECTOR IN ERROR',sector
0690     95       format(' ',t8,a,t30,z2.2)
0691
0692              if (sector .ne. '255'o) then
0693
0694              write(lun,100) lib$extzv(0,8,sector)
0695     100      format(' ',t40,'SECTOR #',i<compress4 (lib$extzv(0,8,sector))>,'.')
0696              endif
0697
0698              return
0699              end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	5763	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	2350	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	2036	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	10661	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$K_SML_DSK_AZTEC

VARIABLES

Address	Type	Name	Address	Type	Name
2-000001D4	I*4	BASE_UNIT	3-00000056	I*2	DESIRED_TRACK
3-00000054	I*2	DISTANCE_TRAVELLED	2-000001CE	L*1	DRIVE_POSITIONER_ERROR
3-0000005A	I*2	DSTS	3-00000000	I*4	EMB$L_HD_SID
3-00000004	I*2	EMB$W_HD_ENTRY	3-0000000E	I*2	EMB$W_HD_ERRSEQ
3-00000050	I*2	FRONT_PANEL	3-0000004E	I*2	FULL_ERROR_CODE
2-000001DC	I*4	I	3-00000050	L*1	L5B
3-00000056	I*4	LAST_HEADER_READ	AP-0000004A	L*1	LUN
2-000001D8	I*4	MINOR_CODE	3-00000026	I*4	MSLG$L_CMD_REF
3-00000030	I*2	MSLG$Q_EVENT	3-00000051	L*1	PL5B
3-0000005D	L*1	SECTOR	3-0000005C	L*1	SEL
2-000001D0	I*4	SELECTED_HEAD	3-00000052	I*2	SERVO_STATE
3-00000054	I*2	SPINDLE_STATE	3-00000058	I*2	STARTING_TRACK
3-0000005A	I*2	TRACK_ERROR_VALUE	3-00000052	I*2	TRACK_IN_ERROR

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	L*1	EMB	512	(0:511)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)
2-00000000	CHAR	V1AZTEC_DSTS	175	(0:6)
2-000000CA	CHAR	V1AZTEC_SEL	155	(0:4)
2-0000017A	CHAR	V1FRONT_PANEL	6	(0:0)
2-000000AF	CHAR	V2AZTEC_DSTS	27	(15:15)
2-00000165	CHAR	V2AZTEC_SEL	21	(7:7)
2-00000180	CHAR	V2FRONT_PANEL	78	(8:10)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-000007A1	15'	1-000007AD	17'	1-000007C1	20'	1-000007CD	22'	1-000007DD	25'	1-000007F4	30'
1-00000800	35'	1-00000807	40'	1-00000824	45'	1-00000830	50'	1-00000850	52'	1-00000857	55'
1-00000863	60'	1-00000878	65'	1-00000884	70'	1-00000894	75'	1-000008AA	80'	1-000008C1	82'
1-000008D8	84'	1-000008E4	85'	1-000008F0	90'	1-00000909	95'	1-00000915	100'		

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name	Type	Name
I*4	COMPRESS4		DISK_HEADER	I*4	LIB$EXTZV		LINCHK		OUTPUT

```
0001
0002
0003
0004          Subroutine MSLG$K_BUS_ADDR_AZTEC (lun)
0005
0006
0007          include 'src$:msghdr.for /nolist'
0066
0067
0068          byte            lun
0069
0070          integer*2       full_error_code
0071          integer*2       start_word_count
0072          integer*2       base_ram_address
0073          integer*2       acs
0074
0075          equivalence     (emb(66),full_error_code)
0076          equivalence     (emb(68),start_word_count)
0077          equivalence     (emb(70),base_ram_address)
0078          equivalence     (emb(72),lesi)
0079
0080          character*34    v1acs(3:7)
0081
0082          data            v1acs(3)
0083          1 /'HOST HAS READ POLL REGISTER*'/
0084
0085          data            v1acs(4)
0086          1 /'PURGE COMPLETE*'/
0087
0088          data            v1acs(5)
0089          1 /'HOST BUS DATA PARITY ERROR*'/
0090
0091          data            v1acs(6)
0092          1 /'CABLE PARITY ERROR ON WRITE*'/
0093
0094          data            v1acs(7)
0095          1 /'LAST TRANSFER NON-EXISTENT MEMORY*'/
0096
0097          integer*4       start_byte_count
0098          integer*4       host_bus_type
0099          integer*4       compress4
0100          integer*4       lib$extzv
0101
0102
0103          call linchk (lun,1)
0104
0105          write(lun,15) 'FULL ERROR CODE',full_error_code
0106    15    format(' ',t8,a,t28,z4.4)
0107
0108          call aztec_full_error_code1 (lun,full_error_code)
0109
0110          start_byte_count = start_word_count * 2
0111
0112          call linchk (lun,2)
0113
0114          write(lun,20) 'START BYTE COUNT',start_word_count,
0115          1 start_byte_count,'. BYTES'
```

```
0116     20       format(' ',t8,a,t28,z4.4,/,
0117              1 T40,i<compress4 (start_byte_count)>,a)
0118
0119              call linchk (lun,2)
0120
0121              write(lun,25) 'BASE RAM ADDRESS',(base_ram_address,i = 1,2),' (OCTAL)'
0122     25       format(' ',t8,a,T28,z4.4,/,
0123              1 t40,o6.6,a)
0124
0125              call linchk (lun,1)
0126
0127              write(lun,30) ACS,acs
0128     30       format(' ',t8,a,z4.4)
0129
0130              host_bus_type = lib$extzv(0,3,acs)
0131
0132              call linchk (lun,1)
0133
0134              if (host_bus_type .eq. 1) then
0135
0136              write(lun,35) 'UNIBUS'
0137     35       format(' ',t40,'HOST BUS IS A ',a,:i<compress4 (host_bus_type)>,:a)
0138
0139              else if (host_bus_type .eq. 2) then
0140
0141              write(lun,35) 'Q-BUS'
0142              else
0143
0144              write(lun,35) 'TYPE #',host_bus_type,'.'
0145              endif
0146
0147              call output (lun,acs,v1acs,3,3,7,'0')
0148
0149              return
0150              end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	522	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	197	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	396	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	1627	

ENTRY POINTS

Address	Type	Name
0-00000000		MSLG$K_BUS_ADDR_AZTEC

VARIABLES

Address	Type	Name	Address	Type	Name
2-000000AA	I*2	ACS	3-00000046	I*2	BASE_RAM_ADDRESS
3-00000000	I*4	EMB$L_HD_SID	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	3-00000042	I*2	FULL_ERROR_CODE
2-000000B0	I*4	HOST_BUS_TYPE	2-000000B4	I*4	I
3-00000048	I*4	LESI	AP-00000004@	L*1	LUN
2-000000AC	I*4	START_BYTE_COUNT	3-00000044	I*2	START_WORD_COUNT

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	L*1	EMB	512	(0:511)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)
2-00000000	CHAR	V1ACS	170	(3:7)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-00000066	15'	1-00000072	20'	1-00000088	25'	1-0000009B	30'	1-000000A5	35'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	AZTEC_FULL_ERROR_CODE1	I*4	COMPRESS4	I*4	LIB$EXTZV
	LINCHR		OUTPUT		

```
0001
0002
0003            Subroutine STATUS_AND_EVENT_CODES (lun)
0004
0005            Include  'SRC$:MSGHDR.FOR /nolist'
0064
0065
0066            Byte            lun
0067
0068            Integer*2       str_len
0069
0070            Integer*4       value_1, compress4, code, sub_code
0071
0072            Character*25    Msg_part1
0073            Character*33    Msg_part2
0074
0075
0076        C
0077        C Decode the status and event codes.
0078        C
0079            Call STATUS_AND_EVENT_DECODE (msg_part1,msg_part2,value_1)
0080
0081        C
0082        C Determine whether it was decoded and output the text.
0083        C
0084
0085            Call LINCHK (lun,2)
0086
0087            Str_len = LIB$EXTZV (0,16,msg_part1)
0088
0089            If (str_len .NE. 1) then
0090              Write (lun,15) msg_part1
0091        15    Format (' ',T40,A<str_len>)
0092
0093            Else
0094        C Not decoded due to unknown code, extract the status/event code
0095        C and output the it in decimal.
0096        C
0097              Code = LIB$EXTZV(0,5,emb(48))
0098
0099              Write (lun,20) 'UNKNOWN CODE #', code
0100        20    Format (' ',T40,A,Z2.2,'(X)')
0101
0102            Endif
0103
0104        C
0105        C Determine whether it was decoded and output the text.
0106        C
0107            Str_len = LIB$EXTZV (0,16,msg_part2)
0108
0109            Call LINCHK (lun,1)
0110
0111            If (
0112            1 (str_len .NE. 1)
0113            1 .AND.
0114            1 (value_1 .NE. -1)
0115            1 ) then
```

```
0116
0117              Write (lun,17) value_1, msg_part2
0118     17       Format (' ',T40,I<COMPRESS4 (value_1)>,A<str_len>)
0119
0120
0121             Else if (
0122             1 (str_len .NE. 1)
0123             1 .AND.
0124             1 (value_1 .EQ. -1)
0125             1 ) then
0126
0127               Write (lun,15) msg_part2
0128
0129
0130     C Not decoded due to unknown subcode, extract the status/event subcode and
0131     C output the it in decimal.
0132     C
0133             Else if (str_len .EQ. 1) then
0134
0135               Sub_code = LIB$EXTZV(5,11,emb(48))
0136
0137               Write (lun,25) 'UNKNOWN SUBCODE #', sub_code
0138     25        Format (' ',T40,A17,Z4.4,'(X)')
0139
0140             Endif
0141
0142
0143             return
0144
0145             end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	373	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	116	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	216	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	1217	

ENTRY POINTS

Address	Type	Name
0-00000000		STATUS_AND_EVENT_CODES

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000040	I*4	CODE	3-00000000	I*4	EMB$L_HD_SID
3-00000004	I*2	EMB$W_HD_ENTRY	3-0000000E	I*2	EMB$W_HD_ERRSEQ
AP-00000004@	L*1	LUN	2-00000000	CHAR	MSG_PART1
2-00000019	CHAR	MSG_PART2	2-0000003A	I*2	STR_LEN
2-00000044	I*4	SUB_CODE	2-0000003C	I*4	VALUE_1

ARRAYS

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

LABELS

Address	Label	Address	Label	Address	Label	Address	Label
1-00000037	15'	1-00000052	17'	1-00000043	20'	1-00000064	25'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK
	STATUS_AND_EVENT_DECODE				

COMMAND QUALIFIERS

```
  FORTRAN /LIS=LIS$:MSCP/OBJ=OBJ$:MSCP MSRC$:MSCP

  /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:	80.92 seconds
Elapsed Time:	170.17 seconds
Page Faults:	591
Dynamic Memory:	328 pages

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PADRIVER
LIS

OUTPUT
LIS

NEW_RTN
LIS

MT_DISMT
LIS

OPNOUTFIL
LIS

OUTPUT0
LIS

NEWFILE
LIS