```
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
```

M 8

FILEID**TSTAPE

```
TTTTTTTTTT    SSSSSSSS  TTTTTTTTTT    AAAAAA    PPPPPPPP    EEEEEEEEEE
TTTTTTTTTT    SSSSSSSS  TTTTTTTTTT    AAAAAA    PPPPPPPP    EEEEEEEEEE
    TT      SS              TT      AA      AA  PP      PP  EE
    TT      SS              TT      AA      AA  PP      PP  EE
    TT      SS              TT      AA      AA  PP      PP  EE
    TT      SS              TT      AA      AA  PP      PP  EE
    TT        SSSSSS        TT      AA      AA  PPPPPPPP    EEEEEEEE
    TT        SSSSSS        TT      AA      AA  PPPPPPPP    EEEEEEEE
    TT              SS      TT      AAAAAAAAAA  PP          EE
    TT              SS      TT      AAAAAAAAAA  PP          EE
    TT              SS      TT      AA      AA  PP          EE              ....
    TT              SS      TT      AA      AA  PP          EE              ....
    TT      SSSSSSSS        TT      AA      AA  PP          EEEEEEEEEE      ....
    TT      SSSSSSSS        TT      AA      AA  PP          EEEEEEEEEE      ....

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
```

TS

08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08
08

```
C
C Version:      'V04-000'
C
C*****************************************************************************
C*                                                                           *
C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
C*  ALL RIGHTS RESERVED.                                                     *
C*                                                                           *
C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
C*  TRANSFERRED.                                                             *
C*                                                                           *
C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
C*  CORPORATION.                                                             *
C*                                                                           *
C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
C*                                                                           *
C*                                                                           *
C*****************************************************************************
C

        SUBROUTINE TSTAPE (LUN)

C
C       AUTHOR  BRIAN PORTER            CREATION DATE   20-SEP-1979
C
C       Functional description:
C
C       TS11 tape handler device error/timeout display module
C
C       Modified by:
C
C       V03-005 SAR0235         Sharon A. Reynolds,     28-Mar-1984
C               Changed the call to UCB$L_OWNUIC to ORB$L_OWNER.
C
C       V03-004 SAR0099         Sharon A. Reynolds,     20-Jun-1983
C               Changed the carriage control in the 'format' statements
C               for use with ERF.
C
C       V03-003 SAR0045         Sharon A. Reynolds,     9-Jun-1983
C               Remove brief/cryptic support.
C
C       v03-002 BP0007          Brian Porter,           20-AUG-1982
C               Minor edit.
C
C       v03-001 BP0006          Brian Porter,           05-MAR-1982
C               Corrected tssr and tsba in 'b' and 'c' formats.
C**

        INCLUDE 'SRC$:MSGHDR.FOR /NOLIST'
        INCLUDE 'SRC$:DEVERR.FOR /NOLIST'
```

```
      BYTE            LUN

      INTEGER*4       MODE
      INTEGER*4       FIELD
      INTEGER*4       COMPRESS4
      INTEGER*4       COMPRESSC
      INTEGER*4       TSBA
      INTEGER*4       TSSR
      INTEGER*4       TS_FUNCTION
      INTEGER*4       UBA_REGS(0:4)
      INTEGER*4       CMD_BUF(0:7)
      INTEGER*4       MSG_BUF(0:6)
      INTEGER*4       MSG_HDR
      INTEGER*4       MSG_LWD
      INTEGER*4       RBPCR
      INTEGER*4       XSTAT0
      INTEGER*4       XSTAT1
      INTEGER*4       XSTAT2
      INTEGER*4       XSTAT3

      PARAMETER       REWIND = 4
      PARAMETER       READ = 1
      PARAMETER       WRITE_CHAR = 4
      PARAMETER       WRITE = 5
      PARAMETER       WRITE_SUB = 6
      PARAMETER       POSITION = 8
      PARAMETER       FORMAT = 9
      PARAMETER       CONTROL = 10
      PARAMETER       INITIALIZE = 11
      PARAMETER       GET_STATUS = 15
      PARAMETER       TIMEOUT = 96
      PARAMETER       XFER_ERR = 1

      CHARACTER*16    V1CMD_HDR(0:1)

      EQUIVALENCE     (EMB$L_DV_REGSAV(0),TSBA)
      EQUIVALENCE     (EMB$L_DV_REGSAV(1),TSSR)
      EQUIVALENCE     (EMB$L_DV_REGSAV(2),UBA_REGS(0))
      EQUIVALENCE     (EMB$L_DV_REGSAV(7),CMD_BUF(0))
      EQUIVALENCE     (EMB$L_DV_REGSAV(15),MSG_BUF(0))
      EQUIVALENCE     (MSG_BUF(0),MSG_HDR)
      EQUIVALENCE     (MSG_BUF(1),MSG_LWD)
      EQUIVALENCE     (MSG_BUF(2),RBPCR)
      EQUIVALENCE     (MSG_BUF(3),XSTAT0)
      EQUIVALENCE     (MSG_BUF(4),XSTAT1)
      EQUIVALENCE     (MSG_BUF(5),XSTAT2)
      EQUIVALENCE     (MSG_BUF(6),XSTAT3)
      EQUIVALENCE     (V2MSG_HDR(0),V1CMD_HDR(0))

      CHARACTER*28    V1XSTAT0(0:15)
      DATA    V1XSTAT0(0)     /'END OF TAPE*'/
      DATA    V1XSTAT0(1)     /'BEGINNING OF TAPE*'/
      DATA    V1XSTAT0(2)     /'WRITE PROTECTED*'/
      DATA    V1XSTAT0(3)     /'PHASE ENCODED DRIVE*'/
      DATA    V1XSTAT0(4)     /'VOLUME CHECK*'/
```

```
      DATA    V1XSTAT0(5)     /'INTERRUPT ENABLE*'/
      DATA    V1XSTAT0(6)     /'DEVICE ON-LINE*'/
      DATA    V1XSTAT0(7)     /'TAPE MOTION ON LAST COMMAND*'/
      DATA    V1XSTAT0(8)     /'ILLEGAL ADDRESS*'/
      DATA    V1XSTAT0(9)     /'ILLEGAL COMMAND*'/
      DATA    V1XSTAT0(10)    /'NON-EXECUTABLE FUNCTION*'/
      DATA    V1XSTAT0(11)    /'WRITE LOCK ERROR*'/
      DATA    V1XSTAT0(12)    /'RECORD LENGTH LONG*'/
      DATA    V1XSTAT0(13)    /'LOGICAL END OF TAPE*'/
      DATA    V1XSTAT0(14)    /'RECORD LENGTH SHORT*'/
      DATA    V1XSTAT0(15)    /'TAPE MARK DETECTED*'/

      CHARACTER*20    V1XSTAT1(0:7)
      DATA    V1XSTAT1(0)     /'MULTI-TRACK ERROR*'/
      DATA    V1XSTAT1(1)     /'UNCORRECTABLE ERROR*'/
      DATA    V1XSTAT1(2)     /'POSTAMBLE LONG*'/
      DATA    V1XSTAT1(3)     /'POSTAMBLE SHORT*'/
      DATA    V1XSTAT1(4)     /'INVALID END DATA*'/
      DATA    V1XSTAT1(5)     /'INVALID POSTAMBLE*'/
      DATA    V1XSTAT1(6)     /'SYNCH FAILURE*'/
      DATA    V1XSTAT1(7)     /'INVALID PREAMBLE*'/

      CHARACTER*19    V2XSTAT1(9:13)
      DATA    V2XSTAT1(9)     /'SPEED CHECK*'/
      DATA    V2XSTAT1(10)    /'DESKEW BUFFER FAIL*'/
      DATA    V2XSTAT1(11)    /'TRASH IN GAP*'/
      DATA    V2XSTAT1(12)    /'CREASE DETECTED*'/
      DATA    V2XSTAT1(13)    /'CORRECTABLE ERROR*'/

      CHARACTER*10    V3XSTAT1(15:15)
      DATA    V3XSTAT1(15)    /'DATA LATE*'/

      CHARACTER*19    V1XSTAT2(10:10)
      DATA    V1XSTAT2(10)    /'WRITE CARD FAILURE*'/

      CHARACTER*29    V2XSTAT2(12:15)
      DATA    V2XSTAT2(12)    /'CAPSTAN ACCELERATION FAILURE*'/
      DATA    V2XSTAT2(13)    /'SERIAL BUS PE AT DRIVE*'/
      DATA    V2XSTAT2(14)    /'SILO PARITY ERROR*'/
      DATA    V2XSTAT2(15)    /'OPERATION IN PROGRESS*'/

      CHARACTER*25    V1XSTAT3(0:7)
      DATA    V1XSTAT3(0)     /'REVERSE INTO BOT*'/
      DATA    V1XSTAT3(1)     /'LIMIT EXCEEDED (STATIC)*'/
      DATA    V1XSTAT3(2)     /'NOISE RECORD*'/
      DATA    V1XSTAT3(3)     /'DENSITY CHECK*'/
      DATA    V1XSTAT3(4)     /'CAPSTAN RESPONSE FAIL*'/
      DATA    V1XSTAT3(5)     /'REVERSE*'/
      DATA    V1XSTAT3(6)     /'OPERATION INCOMPLETE*'/
      DATA    V1XSTAT3(7)     /'LIMIT EXCEEDED (DYNAMIC)*'/

      CHARACTER*34    FATAL_ERROR(0:3)
      DATA    FATAL_ERROR(0)  /'INTERNAL MICRO DIAGNOSTIC FAILURE*'/
      DATA    FATAL_ERROR(1)  /'I/O SEQUENCER CROM PARITY ERROR*'/
      DATA    FATAL_ERROR(2)  /'MICRO PROCESSOR CROM PARITY ERROR*'/
      DATA    FATAL_ERROR(3)  /'AC LO*'/
```

```
      CHARACTER*34    TERMINATION(0:7)
      DATA    TERMINATION(0)  /'NORMAL TERMINATION*'/
      DATA    TERMINATION(1)  /'ATTENTION*'/
      DATA    TERMINATION(2)  /'TAPE STATUS ALERT*'/
      DATA    TERMINATION(3)  /'FUNCTION REJECT*'/
      DATA    TERMINATION(4)  /'RECOVERABLE ERROR (MOTION)*'/
      DATA    TERMINATION(5)  /'RECOVERABLE ERROR (NO MOTION)*'/
      DATA    TERMINATION(6)  /'NON-RECOVERABLE ERROR (POS. LOST)*'/
      DATA    TERMINATION(7)  /'FATAL CONTROLLER ERROR*'/

      CHARACTER*10    V1MSG_HDR(15:20)
      DATA    V1MSG_HDR(15)   /'ILLEGAL*'/
      DATA    V1MSG_HDR(16)   /'END*'/
      DATA    V1MSG_HDR(17)   /'FAIL*'/
      DATA    V1MSG_HDR(18)   /'ERROR*'/
      DATA    V1MSG_HDR(19)   /'ATTENTION*'/
      DATA    V1MSG_HDR(20)   /'ILLEGAL*'/

      CHARACTER*16    V2MSG_HDR(0:1)
      DATA    V2MSG_HDR(0)    /'ONE WORD HEADER*'/
      DATA    V2MSG_HDR(1)    /'ILLEGAL*'/

      CHARACTER*31    V3MSG_HDR(0:2)
      DATA    V3MSG_HDR(0)    /'ON-LINE OR OFF-LINE TRANSITION*'/
      DATA    V3MSG_HDR(1)    /'MICRO DIAGNOSTIC FAILURE*'/
      DATA    V3MSG_HDR(2)    /'ILLEGAL CLASS CODE*'/

      CHARACTER*33    V4MSG_HDR(0:4)
      DATA    V4MSG_HDR(0)    /'PACKET BAD*'/
      DATA    V4MSG_HDR(1)    /'ILC,ILA OR NBA AFTER TAPE MOTION*'/
      DATA    V4MSG_HDR(2)    /'NON-EXECUTABLE FUNCTION*'/
      DATA    V4MSG_HDR(3)    /'MICRO DIAGNOSTIC FAILURE*'/
      DATA    V4MSG_HDR(4)    /'ILLEGAL CLASS CODE*'/

      CHARACTER*33    V5MSG_HDR(15:15)
      DATA    V5MSG_HDR(15)   /'ACKNOWLEDGE, CPU OWNS CMD BUFFER*'/

      CHARACTER*13    V1MSG_LWD(9:11)
      DATA    V1MSG_LWD(9)    /'ILLEGAL*'/
      DATA    V1MSG_LWD(10)   /'= 10.(BYTES)*'/
      DATA    V1MSG_LWD(11)   /'ILLEGAL*'/

      CHARACTER*22    TS_COMMAND(0:11)
      DATA    TS_COMMAND(0)   /'INITIALIZE*'/
      DATA    TS_COMMAND(1)   /'READ*'/
      DATA    TS_COMMAND(2)   /'ILLEGAL*'/
      DATA    TS_COMMAND(3)   /'ILLEGAL*'/
      DATA    TS_COMMAND(4)   /'WRITE CHARACTERISTICS*'/
      DATA    TS_COMMAND(5)   /'WRITE*'/
      DATA    TS_COMMAND(6)   /'DIAGNOSTIC COMMAND*'/
      DATA    TS_COMMAND(7)   /'ILLEGAL*'/
      DATA    TS_COMMAND(8)   /'POSITION*'/
      DATA    TS_COMMAND(9)   /'FORMAT*'/
      DATA    TS_COMMAND(10)  /'CONTROL*'/
      DATA    TS_COMMAND(11)  /'ILLEGAL*'/

      CHARACTER*17    READ_MODE(0:4)
```

```
      DATA    READ_MODE(0)    /'READ NEXT*'/
      DATA    READ_MODE(1)    /'READ PREVIOUS*'/
      DATA    READ_MODE(2)    /'RE-READ PREVIOUS*'/
      DATA    READ_MODE(3)    /'RE-READ NEXT*'/
      DATA    READ_MODE(4)    /'ILLEGAL MODE*'/

      CHARACTER*17    WRITE_MODE(0:2)
      DATA    WRITE_MODE(0)   /'WRITE DATA*'/
      DATA    WRITE_MODE(1)   /'WRITE DATA RETRY*'/
      DATA    WRITE_MODE(2)   /'ILLEGAL MODE*'/

      CHARACTER*24    POSITION_MODE(0:5)
      DATA    POSITION_MODE(0)/'SPACE RECORDS FORWARD*'/
      DATA    POSITION_MODE(1)/'SPACE RECORDS REVERSE*'/
      DATA    POSITION_MODE(2)/'SKIP TAPE MARKS FORWARD*'/
      DATA    POSITION_MODE(3)/'SKIP TAPE MARKS REVERSE*'/
      DATA    POSITION_MODE(4)/'REWIND*'/
      DATA    POSITION_MODE(5)/'ILLEGAL MODE*'/

      CHARACTER*23    CONTROL_MODE(0:3)
      DATA    CONTROL_MODE(0) /'MESSAGE BUFFER RELEASE*'/
      DATA    CONTROL_MODE(1) /'REWIND AND RELOAD*'/
      DATA    CONTROL_MODE(2) /'CLEAN*'/
      DATA    CONTROL_MODE(3) /'ILLEGAL MODE*'/

      CHARACTER*13    NORMAL_MODE(0:1)
      DATA    NORMAL_MODE(0)  /'NORMAL MODE*'/
      DATA    NORMAL_MODE(1)  /'ILLEGAL MODE*'/

      CHARACTER*22    FORMAT_MODE(0:3)
      DATA    FORMAT_MODE(0)  /'WRITE TAPE MARK*'/
      DATA    FORMAT_MODE(1)  /'ERASE*'/
      DATA    FORMAT_MODE(2)  /'WRITE TAPE MARK ENTRY*'/
      DATA    FORMAT_MODE(3)  /'ILLEGAL MODE*'/

      CHARACTER*17    V2CMD_HDR(7:7)
      DATA    V2CMD_HDR(7)    /'INTERRUPT ENABLE*'/

      CHARACTER*34    V3CMD_HDR(12:15)
      DATA    V3CMD_HDR(12)   /'SWAP BYTES*'/
      DATA    V3CMD_HDR(13)   /'OPPOSITE*'/
      DATA    V3CMD_HDR(14)   /'CLEAR VOLUME CHECK*'/
      DATA    V3CMD_HDR(15)   /'ACKNOWLEDGE, TS11 OWNS MSG BUFFER*'/

      CHARACTER*17    V1TSSR(6:7)
      DATA    V1TSSR(6)       /'OFF LINE*'/
      DATA    V1TSSR(7)       /'SUB-SYSTEM READY*'/

      CHARACTER*30    V2TSSR(10:15)
      DATA    V2TSSR(10)      /'NEED BUFFER ADDRESS*'/
      DATA    V2TSSR(11)      /'NON-EXISTENT MEMORY*'/
      DATA    V2TSSR(12)      /'REGISTER MODIFICATION REFUSED*'/
      DATA    V2TSSR(13)      /'SERIAL BUS PARITY ERROR*'/
      DATA    V2TSSR(14)      /'UNIBUS PARITY ERROR*'/
      DATA    V2TSSR(15)      /'SPECIAL CONDITION*'/

      CHARACTER*2     EX_ADDR(8:9)
```

```
      DATA    EX_ADDR(8)      /'16'/
      DATA    EX_ADDR(9)      /'17'/

      CHARACTER*7     V1MODE
      DATA    V1MODE          /'MODE = '/


      CALL FRCTOF (LUN)

      call dhead1 (lun,'UBA TS11')

      CALL LINCHK (LUN,2)

      WRITE(LUN,10) TSBA
10    FORMAT(/' ',T8,'TSBA',T24,Z8.4)

      CALL LINCHK (LUN,1)

      WRITE(LUN,15) TSSR
15    FORMAT(' ',T8,'TSSR',T24,Z8.4)

      IF (JIAND(TSSR,'800E'X) .EQ. '800E'X) THEN

      FIELD = LIB$EXTZV(4,2,TSSR)

      CALL LINCHK (LUN,2)

      WRITE(LUN,20) FATAL_ERROR(FIELD)
20    FORMAT(' ',T40,A<COMPRESSC (FATAL_ERROR(FIELD))>,/,
     1 T40,'FATAL CONTROLLER ERROR')
      ELSE

      FIELD = LIB$EXTZV(1,3,TSSR)

      CALL LINCHK (LUN,1)

      WRITE(LUN,25) TERMINATION(FIELD)
25    FORMAT(' ',T40,A<COMPRESSC (TERMINATION(FIELD))>)
      ENDIF

      CALL OUTPUT (LUN,TSSR,V1TSSR,6,7,7,'0')

      DO 28,I = 8,9

      IF (JIAND(TSSR,2**I) .NE. 0) THEN

      CALL LINCHK (LUN,1)

      WRITE(LUN,27) EX_ADDR(I)
27    FORMAT(' ',T40,'EXTENDED BUS ADDRESS BIT ',A2,'.')
      ENDIF

28    CONTINUE

      CALL OUTPUT (LUN,TSSR,V2TSSR,10,10,15,'0')

      CALL LINCHK (LUN,3)
```

```
              WRITE(LUN,30)
     30       FORMAT(/' ','COMMAND BUFFER',/)

              CALL LINCHK (LUN,1)

              WRITE(LUN,35) CMD_BUF(0)
     35       FORMAT(' ',T8,'CMD HDR',T24,Z8.4)

              TS_FUNCTION = LIB$EXTZV(0,5,CMD_BUF(0))

              CALL LINCHK (LUN,1)

              WRITE(LUN,40) TS_COMMAND(MIN(11,TS_FUNCTION))
     40       FORMAT(' ',T40,'FUNCTION = ',
              1 A<COMPRESSC (TS_COMMAND(MIN(11,TS_FUNCTION)))>)

              FIELD = LIB$EXTZV(5,2,CMD_BUF(0))

              CALL LINCHK (LUN,1)

              WRITE(LUN,45) V1CMD_HDR(MIN(1,FIELD))
     45       FORMAT(' ',T40,A<COMPRESSC (V1CMD_HDR(MIN(1,FIELD)))>)

              CALL OUTPUT (LUN,CMD_BUF(0),V2CMD_HDR,7,7,7,'0')

              IF (TS_FUNCTION .EQ. INITIALIZE
              1 .OR.
              2 TS_FUNCTION .EQ. CONTROL
              3 .OR.
              4 TS_FUNCTION .EQ. FORMAT
              5 .OR.
              6 TS_FUNCTION .EQ. POSITION
              7 .OR.
              8 TS_FUNCTION .EQ. WRITE_SUB
              9 .OR.
              1 TS_FUNCTION .EQ. WRITE
              2 .OR.
              3 TS_FUNCTION .EQ. WRITE_CHAR
              4 .OR.
              5 TS_FUNCTION .EQ. READ
              6 .OR.
              7 TS_FUNCTION .EQ. GET_STATUS) THEN

              CALL LINCHK (LUN,1)

              MODE = LIB$EXTZV(8,4,CMD_BUF(0))

              IF (TS_FUNCTION .EQ. INITIALIZE
              1 .OR.
              2 TS_FUNCTION .EQ. WRITE_SUB
              3 .OR.
              4 TS_FUNCTION .EQ. WRITE_CHAR
              5 .OR.
              6 TS_FUNCTION .EQ. GET_STATUS) THEN

              WRITE(LUN,50) V1MODE,NORMAL_MODE(MIN(1,MODE))
```

```
50      FORMAT(' ',T40,A7,A<COMPRESSC (NORMAL_MODE(MIN(1,MODE)))>)

        ELSE IF (TS_FUNCTION .EQ. CONTROL) THEN

        WRITE(LUN,55) V1MODE,CONTROL_MODE(MIN(4,MODE))
55      FORMAT(' ',T40,A7,A<COMPRESSC (CONTROL_MODE(MIN(4,MODE)))>)

        ELSE IF (TS_FUNCTION .EQ. FORMAT) THEN

        WRITE(LUN,60) V1MODE,FORMAT_MODE(MIN(3,MODE))
60      FORMAT(' ',T40,A7,A<COMPRESSC (FORMAT_MODE(MIN(3,MODE)))>)

        ELSE IF (TS_FUNCTION .EQ. POSITION) THEN

        WRITE(LUN,65) V1MODE,POSITION_MODE(MIN(5,MODE))
65      FORMAT(' ',T40,A7,A<COMPRESSC (POSITION_MODE(MIN(5,MODE)))>)

        ELSE IF (TS_FUNCTION .EQ. WRITE) THEN

        WRITE(LUN,70) V1MODE,WRITE_MODE(MIN(3,MODE))
70      FORMAT(' ',T40,A7,A<COMPRESSC (WRITE_MODE(MIN(3,MODE)))>)

        ELSE IF (TS_FUNCTION .EQ. READ) THEN

        WRITE(LUN,75) V1MODE,READ_MODE(MIN(4,MODE))
75      FORMAT(' ',T40,A7,A<COMPRESSC (READ_MODE(MIN(4,MODE)))>)
        ENDIF
        ENDIF

        CALL OUTPUT (LUN,CMD_BUF(0),V3CMD_HDR,12,12,15,'0')

        IF (TS_FUNCTION .EQ. WRITE
        1 .OR.
        2 TS_FUNCTION .EQ. WRITE_CHAR
        3 .OR.
        4 TS_FUNCTION .EQ. READ) THEN

        CALL LINCHK (LUN,1)

        WRITE(LUN,80) CMD_BUF(1)
80      FORMAT(' ',T8,'LO ADDR BITS',T24,Z8.4)

        CALL LINCHK (LUN,1)

        WRITE(LUN,85) CMD_BUF(2)
85      FORMAT(' ',T8,'HI ADDR BITS',T24,Z8.4)

        CALL CALC_MAP (LUN,0,CMD_BUF(2),CMD_BUF(1))

        CALL LINCHK (LUN,1)

        WRITE(LUN,90) CMD_BUF(3)
90      FORMAT(' ',T8,'SIZE',T24,Z8.4)

        CALL LINCHK (LUN,1)

        WRITE(LUN,95) CMD_BUF(3)
```

```
95      FORMAT(' ',T40,'BUFFER SIZE = ',I<COMPRESS4 (CMD_BUF(3))>,'. BYTES')

        ELSE IF (TS_FUNCTION .EQ. POSITION) THEN

        CALL LINCHK (LUN,1)

        WRITE(LUN,98) CMD_BUF(1)
98      FORMAT(' ',T8,'SKIP/SPACE COUNT',T24,Z8.4)

        CALL LINCHK (LUN,1)

        WRITE(LUN,100) CMD_BUF(1)
100     FORMAT(' ',T40,'SKIP/SPACE ',I<COMPRESS4 (CMD_BUF(1))>
        1 ,'. TAPE MARKS/REC')
        ENDIF

        IF (EMB$W_HD_ENTRY .NE. TIMEOUT) THEN

        CALL LINCHK (LUN,3)

        IF (JIAND(TSSR,'400'X) .NE. 0) THEN

        WRITE(LUN,105)
105     FORMAT(' ','MESSAGE BUFFER NOT VALID',/)
        ELSE

        WRITE(LUN,110)
110     FORMAT(/' ','MESSAGE BUFFER',/)

        CALL LINCHK (LUN,1)

        WRITE(LUN,115) MSG_HDR
115     FORMAT(' ',T8,'MSG HDR',T24,Z8.4)

        CALL LINCHK (LUN,1)

        FIELD = LIB$EXTZV(0,5,MSG_HDR)

        WRITE(LUN,120) V1MSG_HDR(MAX(15,MIN(20,FIELD)))
120     FORMAT(' ',T40,'MESSAGE TYPE = ',
        1 A<COMPRESSC (V1MSG_HDR(MAX(15,MIN(20,FIELD))))>)

        CALL LINCHK (LUN,1)

        FIELD = LIB$EXTZV(5,3,MSG_HDR)

        WRITE(LUN,125) V2MSG_HDR(MIN(1,FIELD))
125     FORMAT(' ',T40,A<COMPRESSC (V2MSG_HDR(MIN(1,FIELD)))>,
        1 ' PACKET FORMAT')

        IF (JIAND(MSG_HDR,'11'X) .EQ. '11'X) THEN

        FIELD = LIB$EXTZV(8,4,MSG_HDR)

        CALL LINCHK (LUN,1)

        IF (JIAND(MSG_HDR,'2'X) .EQ. '2'X) THEN
```

```
            WRITE(LUN,130) V3MSG_HDR(MIN(2,FIELD))
    130     FORMAT(' ',T40,A<COMPRESSC (V3MSG_HDR(MIN(2,FIELD)))>)
            ELSE

            WRITE(LUN,135) V4MSG_HDR(MIN(4,FIELD))
    135     FORMAT(' ',T40,A<COMPRESSC (V4MSG_HDR(MIN(4,FIELD)))>)
            ENDIF
            ENDIF

            CALL OUTPUT (LUN,MSG_HDR,V5MSG_HDR,15,15,15,'0')

            CALL LINCHK (LUN,1)

            WRITE(LUN,140) MSG_LWD
    140     FORMAT(' ',T8,'MSG LEN',T24,Z8.4)

            CALL LINCHK (LUN,1)

            FIELD = LIB$EXTZV(0,16,MSG_LWD)

            WRITE(LUN,145) V1MSG_LWD(MAX(9,MIN(11,FIELD)))
    145     FORMAT(' ',T40,'MESSAGE LENGTH ',
            1 A<COMPRESSC (V1MSG_LWD(MAX(9,MIN(11,FIELD)))>)

            CALL LINCHK (LUN,1)

            WRITE(LUN,150) RBPCR
    150     FORMAT(' ',T8,'RBPCR',T24,Z8.4)

            FIELD = LIB$EXTZV(0,16,RBPCR)

            IF (TS_FUNCTION .EQ. READ) THEN

            CALL LINCHK (LUN,1)

            WRITE(LUN,152) FIELD
    152     FORMAT(' ',T40,'RES. BYTES OF XFER = ',I<COMPRESS4 (FIELD)>,'.')

            ELSE IF (TS_FUNCTION .EQ. POSITION
            1 .AND.
            2 MODE .NE. REWIND) THEN

            CALL LINCHK (LUN,1)

            WRITE(LUN,154) FIELD
    154     FORMAT(' ',T40,'RES. REC/MARKS OF SKIP = ',I<COMPRESS4 (FIELD)>,'.')
            ENDIF

            CALL LINCHK (LUN,1)

            WRITE(LUN,155) XSTAT0
    155     FORMAT(' ',T8,'XSTAT0',T24,Z8.4)

            CALL OUTPUT (LUN,XSTAT0,V1XSTAT0,0,0,15,'0')

            CALL LINCHK (LUN,1)
```

```
            WRITE(LUN,160) XSTAT1
160         FORMAT(' ',T8,'XSTAT1',T24,Z8.4)

            CALL OUTPUT (LUN,XSTAT1,V1XSTAT1,0,0,7,'0')

            CALL OUTPUT (LUN,XSTAT1,V2XSTAT1,9,9,13,'0')

            CALL OUTPUT (LUN,XSTAT1  ?XSTAT1,15,15,15,'0')

            CALL LINCHK (LUN,1)

            WRITE(LUN,165) XSTAT2
165         FORMAT(' ',T8,'XSTAT2',T24,Z8.4)

            DO 178,I = 0,8

            IF (JIAND(XSTAT2,2**I) .NE. 0) THEN

            CALL LINCHK (LUN,1)

            IF (I .NE. 8) THEN

            WRITE(LUN,170) I
170         FORMAT(' ',T40,'DEAD TRACK CHANNEL ',I1,'.')
            ELSE

            WRITE(LUN,175)
175         FORMAT(' ',T40,'DEAD TRACK PARITY CHANNEL')
            ENDIF
            ENDIF

178         CONTINUE

            CALL OUTPUT (LUN,XSTAT2,V1XSTAT2,10,10,10,'0')

            CALL OUTPUT (LUN,XSTAT2,V2XSTAT2,12,12,15,'0')

            CALL LINCHK (LUN,1)

            WRITE(LUN,180) XSTAT3
180         FORMAT(' ',T8,'XSTAT3',T24,Z8.4)

            CALL OUTPUT (LUN,XSTAT3,V1XSTAT3,0,0,7,'0')

            FIELD = LIB$EXTZV(8,8,XSTAT3)

            IF (FIELD .NE. 0) THEN

            CALL LINCHK (LUN,1)

            WRITE(LUN,185) F'ELD
185         FORMAT(' ',T40,'MICRO DIAGNOSTIC ERROR CODE ',O3.3)
            ENDIF
            ENDIF
```

```
0786            if (
0787            1 ((ts_function .eq. read
0788            1 .or.
0789            1 ts_function .eq. write
0790            1 .or.
0791            1 ts_function .eq. write_char
0792            1 .or.
0793            1 ts_function .eq. write_sub)
0794            1 .and.
0795            1 emb$w_hd_entry .ne. 96)
0796            1 .or.
0797            1 emb$w_hd_entry .ne. 98
0798            1 ) then
0799
0800    C
0801    C       IF THE TS11 FUNCTION IS A READ REVERSE THEN
0802    C       THE PREVIOUS MAP IS THE FINAL MAP PLUS ONE.
0803    C
0804
0805            if (uba_regs(0) .ne. 0) then
0806
0807            call uba_datapath (lun,uba_regs(0),uba_regs(1))
0808            endif
0809
0810            call calc_map2 (0,cmd_buf(2),cmd_buf(1),field)
0811
0812            call uba_mapping (lun,field,uba_regs(2))
0813
0814            if (lib$extzv (16,16,emb$l_dv_iosb1) .gt. 512) then
0815
0816            IF (TS_FUNCTION .EQ. READ
0817            1 .AND.
0818            2 (MODE .EQ. 1
0819            3 .OR.
0820            4 MODE .EQ. 2)) THEN
0821
0822            call uba_mapping (lun,(field+2),uba_regs(4))
0823            ELSE
0824
0825            call uba_mapping (lun,(field+1),uba_regs(3))
0826            ENDIF
0827            endif
0828            ENDIF
0829            endif
0830
0831            call linchk (lun,1)
0832
0833            write(lun,190)
0834    190     format(' ',:)
0835
0836            if (emb$w_hd_entry .ne. 98) then
0837
0838            call ucb$b_ertcnt (lun,emb$b_dv_ertcnt)
0839
0840            call ucb$b_ertmax (lun,emb$b_dv_ertmax)
0841            endif
0842
```

```
      call orb$l_owner (lun,emb$l_dv_ownuic)

      call ucb$l_char (lun,emb$l_dv_char)

      call ucb$w_sts (lun,emb$w_dv_sts)

      call ucb$i_opcnt (lun,emb$l_dv_opcnt)

      call ucb$w_errcnt (lun,emb$w_dv_errcnt)

      if (emb$w_hd_entry .ne. 98) then

      call linchk (lun,1)

      write(lun,190)

      call tstape_qio (lun,emb$w_dv_func)

      call irp$w_bcnt (lun,emb$w_dv_bcnt)

      call irp$w_boff (lun,emb$w_dv_boff)

      call irp$l_pid (lun,emb$l_dv_rqpid)

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

      RETURN
      END
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	4242	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	1020	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	4504	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	10278	

ENTRY POINTS

Address	Type	Name
0-00000000		TSTAPE

VARIABLES

Address	Type	Name	Address	Type	Name
3-0000001C	L*1	EMB$B_DV_CLASS	3-00000010	L*1	EMB$B_DV_ERTCNT
3-00000011	L*1	EMB$B_DV_ERTMAX	3-0000003E	L*1	EMB$B_DV_NAMLNG
3-0000003A	L*1	EMB$B_DV_SLAVE	3-0000001D	L*1	EMB$B_DV_TYPE
3-00000036	I*4	EMB$L_DV_CHAR	3-00000012	I*4	EMB$L_DV_IOSB1
3-00000016	I*4	EMB$L_DV_IOSB2	3-00000026	I*4	EMB$L_DV_MEDIA
3-0000004E	I*4	EMB$L_DV_NUMREG	3-0000002E	I*4	EMB$L_DV_OPCNT
3-00000032	I*4	EMB$L_DV_OWNUIC	3-0000001E	I*4	EMB$L_DV_RQPID
3-00000000	I*4	EMB$L_HD_SID	3-0000003F	CHAR	EMB$T_DV_NAME
3-00000024	I*2	EMB$W_DV_BCNT	3-00000022	I*2	EMB$W_DV_BOFF
3-0000002C	I*2	EMB$W_DV_ERRCNT	3-0000003C	I*2	EMB$W_DV_FUNC
3-0000001A	I*2	EMB$W_DV_STS	3-0000002A	I*2	EMB$W_DV_UNIT
3-00000004	I*2	EMB$W_HD_ENTRY	3-0000000E	I*2	EMB$W_HD_ERRSEQ
2-00000BC4	I*4	FIELD	2-00000BCC	I*4	I
AP-00000004@	L*1	LUN	2-00000BC0	I*4	MODE
3-0000008E	I*4	MSG_HDR	3-00000092	I*4	MSG_LWD
3-00000096	I*4	RBPCR	3-00000052	I*4	TSBA
3-00000056	I*4	TSSR	2-00000BC8	I*4	TS_FUNCTION
2-00000BB7	CHAR	V1MODE	3-0000009A	I*4	XSTAT0
3-0000009E	I*4	XSTAT1	3-000000A2	I*4	XSTAT2
3-000000A6	I*4	XSTAT3			

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-0000006E	I*4	CMD_BUF	32	(0:7)
2-00000976	CHAR	CONTROL_MODE	92	(0:3)
3-00000000	L*1	EMB	512	(0:511)
3-00000052	I*4	EMB$L_DV_REGSAV	420	(0:104)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)
2-00000BB3	CHAR	EX_ADDR	4	(8:9)
2-00000438	CHAR	FATAL_ERROR	136	(0:3)
2-000009EC	CHAR	FORMAT_MODE	88	(0:3)

```
  3-0000008E  I*4  MSG_BUF                          28  (0:6)
  2-000009D2  CHAR NORMAL_MODE                      26  (0:1)
  2-000008E6  CHAR POSITION_MODE                   144  (0:5)
  2-0000085E  CHAR READ_MODE                        85  (0:4)
  2-000004C0  CHAR TERMINATION                     272  (0:7)
  2-00000756  CHAR TS_COMMAND                      264  (0:11)
  3-0000005A  I*4  UBA_REGS                         20  (0:4)
  2-00000000  CHAR V1CMD_HDR                        32  (0:1)
  2-000005D0  CHAR V1MSG_HDR                        60  (15:20)
  2-0000072F  CHAR V1MSG_LWD                        39  (9:11)
  2-00000ADD  CHAR V1TSSR                           34  (6:7)
  2-00000020  CHAR V1XSTAT0                        448  (0:15)
  2-000001E0  CHAR V1XSTAT1                        160  (0:7)
  2-000002E9  CHAR V1XSTAT2                         19  (10:10)
  2-00000370  CHAR V1XSTAT3                        200  (0:7)
  2-00000A44  CHAR V2CMD_HDR                        17  (7:7)
  2-00000000  CHAR V2MSG_HDR                        32  (0:1)
  2-00000AFF  CHAR V2TSSR                          180  (10:15)
  2-00000280  CHAR V2XSTAT1                         95  (9:13)
  2-000002FC  CHAR V2XSTAT2                        116  (12:15)
  2-00000A55  CHAR V3CMD_HDR                       136  (12:15)
  2-0000060C  CHAR V3MSG_HDR                        93  (0:2)
  2-000002DF  CHAR V3XSTAT1                         10  (15:15)
  2-00000669  CHAR V4MSG_HDR                       165  (0:4)
  2-0000070E  CHAR V5MSG_HDR                        33  (15:15)
  2-000008B3  CHAR WRITE_MODE                       51  (0:2)
```

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-00000047	10'	1-00000059	15'	1-0000006A	20'	1-00000091	25'	1-0000009D	27'	**	28
1-000000C3	30'	1-000000D9	35'	1-000000ED	40'	1-00000106	45'	1-00000112	50'	1-00000120	55'
1-0000012E	60'	1-0000013C	65'	1-0000014A	70'	1-00000158	75'	1-00000166	80'	1-0000017F	85'
1-00000198	90'	1-000001A9	95'	1-000001CE	98'	1-000001EB	100'	1-00000216	105'	1-00000235	110'
1-0000024B	115'	1-0000025F	120'	1-0000027C	125'	1-00000298	130'	1-000002A4	135'	1-000002B0	140'
1-000002C4	145'	1-000002E1	150'	1-000002F3	152'	1-00000319	154'	1-00000343	155'	1-00000356	160'
1-00000369	165'	1-0000037C	170'	1-0000039C	175'	**	178	1-000003BD	180'	1-000003D0	185'
1-000003F7	190'										

FUNCTIONS AND SUBROUTINES REFERENCED

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

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

      include 'src$:qiocommon.for /nolist'

      byte            lun

      integer*2       emb$w_dv_func

      integer*4       qiocode(0:1,0:63)

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

      qiocode(1,00) = %loc(io$_nop)

      qiocode(1,01) = %loc(io$_unload)

      qiocode(1,02) = %loc(io$_spacefile)

      qiocode(1,03) = %loc(io$_recal)

      qiocode(1,06) = %loc(io$_erasetape)

      qiocode(1,08) = %loc(io$_packack)

      qiocode(1,09) = %loc(io$_spacerecord)

      qiocode(1,10) = %loc(io$_writecheck)

      qiocode(1,11) = %loc(io$_writepblk)

      qiocode(1,12) = %loc(io$_readpblk)

      qiocode(1,22) = %loc(io$_rereadn)

      qiocode(1,23) = %loc(io$_rereadp)

      qiocode(1,24) = %loc(io$_writeret)

      qiocode(1,25) = %loc(io$_readpreset)

      qiocode(1,26) = %loc(io$_setchar)

      qiocode(1,27) = %loc(io$_sensechar)

      qiocode(1,28) = %loc(io$_writemark)

      qiocode(1,29) = %loc(io$_wrttmkr)

      qiocode(1,30) = %loc(io$_clean)

      qiocode(1,32) = %loc(io$_writelblk)
```

```
0321            qiocode(1,33) = %loc(io$_readlblk)                                  00
0322                                                                                00
0323            qiocode(1,34) = %loc(io$_rewindoff)                                 00
0324                                                                                00
0325            qiocode(1,35) = %loc(io$_setmode)                                   00
0326                                                                                00
0327            qiocode(1,36) = %loc(io$_rewind)                                    00
0328                                                                                00
0329            qiocode(1,37) = %loc(io$_skipfile)                                  00
0330                                                                                01
0331            qiocode(1,38) = %loc(io$_skiprecord)                                02
0332                                                                                02
0333            qiocode(1,39) = %loc(io$_sensemode)                                 02
0334                                                                                02
0335            qiocode(1,40) = %loc(io$_writeof)                                   02
0336                                                                                02
0337            qiocode(1,48) = %loc(io$_writevblk)                                 02
0338                                                                                02
0339            qiocode(1,49) = %loc(io$_readvblk)                                  02
0340                                                                                02
0341            qiocode(1,50) = %loc(io$_access)                                    02
0342                                                                                02
0343            qiocode(1,51) = %loc(io$_create)                                    02
0344                                                                                02
0345            qiocode(1,52) = %loc(io$_deaccess)                                  02
0346                                                                                02
0347            qiocode(1,53) = %loc(io$_delete)                                    02
0348                                                                                02
0349            qiocode(1,54) = %loc(io$_modify)                                    02
0350                                                                                02
0351            qiocode(1,56) = %loc(io$_acpcontrol)                                02
0352                                                                                02
0353            qiocode(1,57) = %loc(io$_mount)                                     02
0354                                                                                02
0355            do 10,i = 0,63                                                      02
0356                                                                                02
0357            qiocode(0,i) = 33                                                   02
0358                                                                                02
0359            if (qiocode(1,i) .eq. 0) then                                       02
0360                                                                                02
0361            qiocode(1,i) = %loc(qio_string)                                     02
0362            endif                                                               02
0363                                                                                02
0364    10      continue                                                            02
0365            endif                                                               02
0366                                                                                02
0367            call irp$w_func (lun,emb$w_dv_func,                                 02
0368            1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))                          02
0369                                                                                02
0370            return                                                              02
0371                                                                                02
0372            end                                                                 02
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	346	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	8	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	548	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	QIOCOMMON	1247	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	2149	

ENTRY POINTS

Address	Type	Name
0-00000000		TSTAPE_QIO

VARIABLES

Address	Type	Name	Address	Type	Name
AP-0000008@	I*2	EMBSW_DV_FUNC	2-00000200	I*4	I
3-00000442	CHAR	IO$_ABORT	3-0000034D	CHAR	IO$_ACCESS
3-000003C2	CHAR	IO$_ACPCONTROL	3-000004B3	CHAR	IO$_AVAILABLE
3-00000297	CHAR	IO$_CLEAN	3-00000369	CHAR	IO$_CREATE
3-00000385	CHAR	IO$_DEACCESS	3-00000393	CHAR	IO$_DELETE
3-0000026D	CHAR	IO$_DIAGNOSE	3-00000065	CHAR	IO$_DRVCLR
3-000004CB	CHAR	IO$_DSE	3-000000A9	CHAR	IO$_ERASETAPE
3-00000276	CHAR	IO$_FORMAT	3-00000071	CHAR	IO$_INITIALIZE
3-00000014	CHAR	IO$_LOADMCODE	3-000003A1	CHAR	IO$_MODIFY
3-000003E2	CHAR	IO$_MOUNT	3-00000000	CHAR	IO$_NOP
3-0000009D	CHAR	IO$_OFFSET	3-000000EB	CHAR	IO$_PACKACK
3-000000E0	CHAR	IO$_QSTOP	3-000003EF	CHAR	IO$_RDSTATS
3-00000421	CHAR	IO$_READCSR	3-00000169	CHAR	IO$_READHEAD
3-000002B6	CHAR	IO$_READLBLK	3-0000013F	CHAR	IO$_READPBLK
3-00000200	CHAR	IO$_READPRESET	3-00000195	CHAR	IO$_READTRACKD
3-0000033A	CHAR	IO$_READVBLK	3-0000045A	CHAR	IO$_READWTHBUF
3-00000484	CHAR	IO$_READWTHXBUF	3-0000004D	CHAR	IO$_RECAL
3-0000007C	CHAR	IO$_RELEASE	3-000001AB	CHAR	IO$_REREADN
3-000001B8	CHAR	IO$_REREADP	3-000000CA	CHAR	IO$_RETCENTER
3-000002E6	CHAR	IO$_REWIND	3-000002C9	CHAR	IO$_REWINDOFF
3-000000FC	CHAR	IO$_SEARCH	3-00000024	CHAR	IO$_SEEK
3-00000231	CHAR	IO$_SENSECHAR	3-00000309	CHAR	IO$_SENSEMODE
3-0000021D	CHAR	IO$_SETCHAR	3-000003B8	CHAR	IO$_SETCLOCK
3-00000088	CHAR	IO$_SETCLOCKP	3-000002DD	CHAR	IO$_SETMODE
3-000002ED	CHAR	IO$_SKIPFILE	3-000002FA	CHAR	IO$_SKIPRECORD
3-00000029	CHAR	IO$_SPACEFILE	3-0000010E	CHAR	IO$_SPACERECORD
3-000003D7	CHAR	IO$_STARTDATA	3-000000B4	CHAR	IO$_STARTDATAP
3-00000037	CHAR	IO$_STARTMPROC	3-0000020F	CHAR	IO$_STARTSPNDL
3-00000059	CHAR	IO$_STOP	3-0000000D	CHAR	IO$_UNLOAD
3-0000046B	CHAR	IO$_WRITEBUFNCRC	3-0000011E	CHAR	IO$_WRITECHECK
3-000001E4	CHAR	IO$_WRITECHECKH	3-000003FF	CHAR	IO$_WRITECSR
3-00000153	CHAR	IO$_WRITEHEAD	3-000002A2	CHAR	IO$_WRITELBLK
3-00000247	CHAR	IO$_WRITEMARK	3-00000314	CHAR	IO$_WRITEOF
3-0000012A	CHAR	IO$_WRITEPBLK	3-000001C9	CHAR	IO$_WRITERET

```
  3-0000017E  CHAR IO$_WRITETRACKD                3-00000326  CHAR IO$_WRITEVBLK
  3-00000448  CHAR IO$_WRITEWTHBUF                3-00000257  CHAR IO$_WRTTMKR
 AP-00000004@ L*1  LUN                            3-000004A1  CHAR QIO_STRING

ARRAYS

    Address  Type  Name                  Bytes  Dimensions

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

LABELS

    Address   Label

      **      10

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name              Type  Name

        IRP$W_FUNC         I*4  LIB$EXTZV

COMMAND QUALIFIERS

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

  /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:            14.63 seconds
  Elapsed Time:        39.25 seconds
  Page Faults:         289
  Dynamic Memory:      266 pages
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

TIMCMP LIS
TUTAPE LIS
SBI LIS
STSEVENT LIS
TRNS_BITS LIS
UBAERR LIS
TIMRB LIS
UBA LIS
TU81SENSE LIS
UNDEFINED LIS
SYSPWRFL LIS
SUMMARY LIS
UNKN_DISP LIS
SHRVECTOR LIS
SYSTARTUP LIS
UBAINT LIS
TOF LIS
UNKNOWN LIS
TSTAPE LIS