
MMM MMM TTTTTTTTTTTTTTT AAAAAAAAA AAAAAAAAA CCCCCCCCCCCC PPPPPPPPPPPP
MMM MMM TTTTTTTTTTTTTTT AAAAAAAAA AAAAAAAAA CCCCCCCCCCCC PPPPPPPPPPPP
MMM MMM TTTTTTTTTTTTTTT AAAAAAAAA AAAAAAAAA CCCCCCCCCCCC PPPPPPPPPPPP
MMMMMM MMMMMM TTT AAA AAA AAA AAA CCC PPP PPP
MMMMMM MMMMMM TTT AAA AAA AAA AAA CCC PPP PPP
MMMMMM MMMMMM TTT AAA AAA AAA AAA CCC PPP PPP
MMM MMM MMM TTT AAA AAA AAA AAA CCC PPP PPP
MMM MMM MMM TTT AAA AAA AAA AAA CCC PPP PPP
MMM MMM MMM TTT AAA AAA AAA AAA CCC PPP PPP
MMM MMM TTT AAA AAA AAA AAA CCC PPPPPPPPPPPP
MMM MMM TTT AAA AAA AAA AAA CCC PPPPPPPPPPPP
MMM MMM TTT AAA AAA AAA AAA CCC PPPPPPPPPPPP
MMM MMM TTT AAAAAAAAAAAAAAA AAAAAAAAAAAAAAA CCC PPP
MMM MMM TTT AAAAAAAAAAAAAAA AAAAAAAAAAAAAAA CCC PPP
MMM MMM TTT AAAAAAAAAAAAAAA AAAAAAAAAAAAAAA CCC PPP
MMM MMM TTT AAA AAA AAA AAA CCC PPP
MMM MMM TTT AAA AAA AAA AAA CCC PPP
MMM MMM TTT AAA AAA AAA AAA CCC PPP
MMM MMM TTT AAA AAA AAA AAA CCCCCCCCCCCC PPP
MMM MMM TTT AAA AAA AAA AAA CCCCCCCCCCCC PPP
MMM MMM TTT AAA AAA AAA AAA CCCCCCCCCCCC PPP

FILEID**ENDVOL

```
EEEEEEEEEE  NN      NN  DDDDDDDD    VV      VV    000000    LL
EEEEEEEEEE  NN      NN  DDDDDDDD    VV      VV    000000    LL
EE          NN      NN  DD      DD  VV      VV  00      00  LL
EE          NN      NN  DD      DD  VV      VV  00      00  LL
EE          NNNN    NN  DD      DD  VV      VV  00      00  LL
EE          NNNN    NN  DD      DD  VV      VV  00      00  LL
EEEEEEEE    NN  NN  NN  DD      DD  VV      VV  00      00  LL
EEEEEEEE    NN  NN  NN  DD      DD  VV      VV  00      00  LL
EE          NN    NNNN  DD      DD  VV      VV  00      00  LL
EE          NN    NNNN  DD      DD  VV      VV  00      00  LL
EE          NN      NN  DD      DD    VV  VV    00      00  LL          ....
EE          NN      NN  DD      DD    VV  VV    00      00  LL          ....
EEEEEEEEEE  NN      NN  DDDDDDDD        VV      000000    LLLLLLLLLL  ....
EEEEEEEEEE  NN      NN  DDDDDDDD        VV      000000    LLLLLLLLLL  ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
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
```

END
V04

:

```
;     1          0001  0
;     2          0002  0 MODULE ENDVOL (LANGUAGE (BLISS32) ,
;     3          0003  0                IDENT = 'V04-000'
;     4          0004  0                ) =
;     5          0005  1 BEGIN
;     6          0006  1 !
;     7          0007  1 !**************************************************************************
;     8          0008  1 !*                                                                        *
;     9          0009  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;    10          0010  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;    11          0011  1 !*  ALL RIGHTS RESERVED.                                                  *
;    12          0012  1 !*                                                                        *
;    13          0013  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    14          0014  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    15          0015  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
;    16          0016  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
;    17          0017  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
;    18          0018  1 !*  TRANSFERRED.                                                           *
;    19          0019  1 !*                                                                        *
;    20          0020  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
;    21          0021  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
;    22          0022  1 !*  CORPORATION.                                                           *
;    23          0023  1 !*                                                                        *
;    24          0024  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    25          0025  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    26          0026  1 !*                                                                        *
;    27          0027  1 !*                                                                        *
;    28          0028  1 !**************************************************************************
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 !
;    32          0032  1 ! FACILITY:  MTAACP
;    33          0033  1 !
;    34          0034  1 ! ABSTRACT:
;    35          0035  1 !       this module handles virtaul io errors including mapping errors.
;    36          0036  1 !
;    37          0037  1 !
;    38          0038  1 ! ENVIRONMENT:
;    39          0039  1 !
;    40          0040  1 !       starlet operating system, including privileged system services
;    41          0041  1 !       and internal exec routines.
;    42          0042  1 !
;    43          0043  1 !--
;    44          0044  1
;    45          0045  1 !
;    46          0046  1 !
;    47          0047  1 ! AUTHOR:  D. H. GILLESPIE,      CREATION DATE:
;    48          0048  1 !
;    49          0049  1 ! MODIFIED BY:
;    50          0050  1 !
;    51          0051  1 !       V03-005 ROW0258         Ralph O. Weber          21-NOV-1983
;    52          0052  1 !                       The Paul Painter Memorial Enhancement
;    53          0053  1 !               Named for one of the unfortunate customers who suffered much
;    54          0054  1 !               to determine the great UCB$L_MT_RECORD secret while trying to
;    55          0055  1 !               create a user-written magtape driver, this change eliminates
;    56          0056  1 !               use of the device dependent field, UCB$L_MT_RECORD in favor of
;    57          0057  1 !               the device independent field, UCB$L_RECORD.
```

```
;    58          0058  1 !
;    59          0059  1 !       V03-004 MMD0148         Meg Dumont,     26-Apr-1983  8:50
;    60          0060  1 !               Change references to 80 to the symbol ANSI_LBLSZ
;    61          0061  1 !
;    62          0062  1 !       V03-003 MMD003          Meg Dumont,     21-Jan-1983  12:29
;    63          0063  1 !               Add a check in User handle of EOT code. If reading a tape
;    64          0064  1 !               and EOT is encountered ignore the error and requeue the I/O's.
;    65          0065  1 !
;    66          0066  1 !       V03-002 MMD0002         Meg Dumont,     3-Jan-1983  15:31
;    67          0067  1 !               Fix to the User EOT handling code. Add code to stop access to
;    68          0068  1 !               a file if the trailer labels have been read. Fix to read
;    69          0069  1 !               reverse, to allow for 0 length files on tape.
;    70          0070  1 !
;    71          0071  1 !       V03-001 MMD0001         Meg Dumont,     5-Nov-1982  16:28
;    72          0072  1 !               Support for the SS$_SERIOUSEXCP code from the streaming
;    73          0073  1 !               tape drives. Support to check for USER EOT condition
;    74          0074  1 !               and pass the IO to the user with success code of EOV, EOT
;    75          0075  1 !               or EOF. Added a check so that if user is reading a
;    76          0076  1 !               file in reverse the ACP will not lose its postion on the
;    77          0077  1 !               tape.
;    78          0078  1 !
;    79          0079  1 !       V02-006 REFORMAT        Maria del C. Nasr       30-Jun-1980
;    80          0080  1 !
;    81          0081  1 !       V02-005 MCN0017         Maria del C. Nasr       18-Jun-1980
;    82          0082  1 !               Request START_VIO in next volume write after IO has been
;    83          0083  1 !               successfully completed.  This is to fix a problem with
;    84          0084  1 !               multivolume processing, in which the next volume was not
;    85          0085  1 !               requested if EOT was sensed when writing the header labels.
;    86          0086  1 !
;    87          0087  1 !       V02-004 SPR27361        Maria del C. Nasr       10-Jun-1980
;    88          0088  1 !               Hold posting of IO after all IO has been completed successfully,
;    89          0089  1 !               to avoid hanging the user process by the need to return errors
;    90          0090  1 !               after IO has been posted.  Also, eliminate the check for user
;    91          0091  1 !               labels AST's.
;    92          0092  1 !
;    93          0093  1 !
;    94          0094  1 !**
;    95          0095  1
;    96          0096  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    97          0097  1
;    98          0098  1 REQUIRE 'SRC$:MTADEF.B32';
;    99          0482  1
;   100          0483  1 FORWARD ROUTINE
;   101          0484  1     END_OF_VOL          : NOPRES NOVALUE,               ! end of volume processing
;   102          0485  1
;   103          0486  1     ! insert request in error at head of blocked io queue
;   104          0487  1     !
;   105          0488  1     INSERT_HEAD         : COMMON_CALL NOVALUE,
;   106          0489  1
;   107          0490  1     ! insert map failures at tail of blocked io queue
;   108          0491  1     !
;   109          0492  1     INSERT_TAIL         : COMMON_CALL NOVALUE,
;   110          0493  1     NEXT_VOL_READ       : NOVALUE L$NEXT_VOL_READ,      ! get next volume on read
;   111          0494  1     NEXT_VOL_WRITE      : NOVALUE L$NEXT_VOL_WRIT;      ! get next volume on write
;   112          0495  1
;   113          0496  1 EXTERNAL ROUTINE
;   114          0497  1     ADJTM               : COMMON_CALL,                  ! adjust tape mark counter
```

```
;  115     0498  1      CHCK_IO_CLR_EXCP    : COMMON_CALL NOVALUE,          ! wait for io from driver
;  116     0499  1      COMPLETE_VIO        : COMMON_CALL NOVALUE,          ! complete outstanding io's
;  117     0500  1      GTNEXT_VOL_READ     : L$GTNEXT_VOL_RE NOVALUE,      ! get next volume for read
;  118     0501  1      GTNEXT_VOL_WRIT     : L$GTNEXT_VOL_WR NOVALUE,      ! get next volume for write
;  119     0502  1      IO_DONE,                                            ! complete io
;  120     0503  1      LIB$CVT_DTB         : ADDRESSING_MODE (ABSOLUTE),   ! convert decimal to binary
;  121     0504  1      READ_BLOCK          : COMMON_CALL,                  ! read one tape block
;  122     0505  1      READ_BLOCK_REVERSE  : COMMON_CALL,                  ! read reverse one tape block
;  123     0506  1      RETURN_ALL_ERR      : COMMON_CALL,                  ! return blocked io in error
;  124     0507  1      SPACE_TM            : COMMON_CALL,                  ! space given number of tape marks
;  125     0508  1      START_VIO           : COMMON_CALL,                  ! start up virtual io
;  126     0509  1      STOP_VIO            : COMMON_CALL,                  ! disable virtual io requests
;  127     0510  1      RESTORE_POS         : COMMON_CALL,                  ! retores tape position lost when reading backwards
;  128     0511  1      REPOSITION          : L$REPOSITION,                 ! reposition tape
;  129     0512  1      WRITE_HEADERS       : L$WRITE_HEADER NOVALUE,       ! write hdr1 and hdr2
;  130     0513  1      WRITE_TM            : L$WRITE_TM,                   ! write tape mark
;  131     0514  1      WRITE_TRAILERS      : L$WRITE_TRAILER NOVALUE;      ! write trailer label set
;  132     0515  1
;  133     0516  1  EXTERNAL
;  134     0517  1      CURRENT_UCB         : REF BBLOCK,                   ! address of current unit control block
;  135     0518  1      CURRENT_WCB         : REF BBLOCK,                   ! address of current window control block
;  136     0519  1      HDR1                : REF BBLOCK,                   ! address of hdr1 (eof1) label
;  137     0520  1      IO_PACKET           : REF BBLOCK,                   ! address of io request packet
;  138     0521  1      USER_STATUS         : VECTOR [2];                   ! status returned to user
;  139     0522  1
```

```
;  141     0523  1 GLOBAL ROUTINE END_OF_VOL : NOPRES NOVALUE =
;  142     0524  1
;  143     0525  1 !++
;  144     0526  1 !
;  145     0527  1 ! FUNCTIONAL DESCRIPTION:
;  146     0528  1 !       this routine handles io errors on virtual io including mapping errors
;  147     0529  1 !
;  148     0530  1 ! CALLING SEQUENCE:
;  149     0531  1 !       end_of_vol()
;  150     0532  1 !
;  151     0533  1 ! INPUT PARAMETERS:
;  152     0534  1 !       none
;  153     0535  1 !
;  154     0536  1 ! IMPLICIT INPUTS:
;  155     0537  1 !       current_vcb     - address of current volume control block
;  156     0538  1 !       io_packet       - address of io request packet
;  157     0539  1 !
;  158     0540  1 ! OUTPUT PARAMETERS:
;  159     0541  1 !       none
;  160     0542  1 !
;  161     0543  1 ! IMPLICIT OUTPUTS:
;  162     0544  1 !       io_packet zeroed if it should not be completed
;  163     0545  1 !
;  164     0546  1 ! ROUTINE VALUE:
;  165     0547  1 !       none
;  166     0548  1 !
;  167     0549  1 ! SIDE EFFECTS:
;  168     0550  1 !       end of volume processing
;  169     0551  1 !
;  170     0552  1 !--
;  171     0553  1
;  172     0554  2     BEGIN
;  173     0555  2
;  174     0556  2     EXTERNAL REGISTER
;  175     0557  2         COMMON_REG;
;  176     0558  2
;  177     0559  2     LOCAL
;  178     0560  2         BLOCKS1,                                ! number of blocks read
;  179     0561  2         BLOCKS2,                                ! number of blocks recorded in trailer label
;  180     0562  2         TM,                                     ! number of TM read
;  181     0563  2         IO_ERROR,                               ! driver error
;  182     0564  2         PACKET  : REF BBLOCK;                   ! address of io request packet
;  183     0565  2
;  184     0566  2     PACKET = .IO_PACKET;
;  185     0567  2
;  186     0568  2     ! if virtual bit set then mapping error.  this occurs when io is being held
;  187     0569  2     ! up for either user labels, io error, end_of_volume processing or because
;  188     0570  2     ! end_of_file has been read
;  189     0571  2     !
;  190     0572  2
;  191     0573  2     IF .PACKET[IRP$V_VIRTUAL]
;  192     0574  2     THEN
;  193     0575  3         BEGIN
;  194     0576  3
;  195     0577  3         ! if mapping error caused by writing user labels and therefore forcing a
;  196     0578  3         ! close of the current file then return the io in error
;  197     0579  3         !
```

```
;  198          0580  3
;  199          0581  3          IF .CURRENT_VCB[VCB$V_MUSTCLOSE]
;  200          0582  3          THEN
;  201          0583  3              ERR_EXIT(SS$_MUSTCLOSEFL);
;  202          0584  3
;  203          0585  3          ! if the number of tape marks into the file is two and label indicates
;  204          0586  3          ! eof then at end_of_file
;  205          0587  3          !
;  206          0588  3
;  207          0589  3          IF .HDR1[HD1$L_HD1LID] EQL 'EOF1'
;  208          0590  3          THEN
;  209          0591  3              ERR_EXIT(SS$_ENDOFFILE);
;  210          0592  3
;  211          0593  3          ! blocked io because of mapping failure
;  212          0594  3          ! Check to see if user handling of EOT is enabled. If it
;  213          0595  3          ! is then complete request with abort if not queue for furture
;  214          0596  3          ! use
;  215          0597  3
;  216          0598  3          IF .CURRENT_VCB[VCB$V_ENUSEREOT]
;  217          0599  3          THEN
;  218          0600  4             BEGIN
;  219          0601  4                  PACKET[IRP$L_IOST1] = SS$_ABORT;
;  220          0602  4                  RETURN;
;  221          0603  4             END
;  222          0604  3          ELSE
;  223          0605  4              BEGIN
;  224          0606  4                  KERNEL_CALL(INSERT_TAIL, .PACKET);
;  225          0607  4                  IO_PACKET = 0;
;  226          0608  4                  RETURN;
;  227          0609  4               END;
;  228          0610  3
;  229          0611  3          END
;  230          0612  2      ELSE
;  231          0613  2
;  232          0614  2          ! if  virtual bit is not set then an error occurred on a virtual io
;  233          0615  2          ! request. the error if ss$_endoftape or ss$_endoffile indicates that
;  234          0616  2          ! end of volume processing is necessary
;  235          0617  2          !
;  236          0618  2          ! The call to STOP_VIO is required in order to eliminate any confusion
;  237          0619  2          ! between the user and the ACP about who should be reading
;  238          0620  2          ! or writing to the tape at any given time. This will cause any
;  239          0621  2          ! IO's tried to fail with either an SS$_ILLBLKNUM or SS$_ENDOFFILE
;  240          0622  2          ! depending on the type of processing the ACP is doing.
;  241          0623  2          !
;  242          0624  3          BEGIN
;  243          0625  3          KERNEL_CALL(STOP_VIO);                  ! Don't allow virtual IO's
;  244          0626  3          IO_ERROR = .(PACKET[IRP$L_IOST1])<0, 16>;
;  245          0627  3
;  246          0628  3          IF .IO_ERROR EQL SS$_MEDOFL
;  247          0629  3          THEN
;  248          0630  4              BEGIN
;  249          0631  4              REPOSITION( .CURRENT_UCB[UCB$L_RECORD] );
;  250          0632  4              KERNEL_CALL(INSERT_HEAD, .PACKET);
;  251          0633  4              IO_PACKET = 0;                      ! don't complete io
;  252          0634  4              KERNEL_CALL(START_VIO);
;  253          0635  4              RETURN;
;  254          0636  4
```

```
;  255     0637  3          END;
;  256     0638  3
;  257     0639  3      IF .IO_ERROR EQL SS$_SERIOUSEXCP
;  258     0640  3      THEN
;  259     0641  4          BEGIN
;  260     0642  4
;  261     0643  4              ! if the IO was complete because of a serious exception then
;  262     0644  4              ! check to see if that was caused by an EOF. If caused
;  263     0645  4              ! by EOF then the IO must be completed to the user.
;  264     0646  4              !
;  265     0647  4              ! NOTE: When doing read aheads RMS will wait for all IO's to
;  266     0648  4              ! be completed before it does any EOF processing.
;  267     0649  4
;  268     0650  4              IF .HDR1[HD1$L_HD1LID] EQL 'EOF1'
;  269     0651  4              THEN
;  270     0652  4                  ERR_EXIT(SS$_ENDOFFILE);
;  271     0653  4
;  272     0654  4              ! Check to see if user handling of EOT is enabled. If it
;  273     0655  4              ! is then complete request with abort if not queue for furture
;  274     0656  4              ! use.
;  275     0657  4
;  276     0658  4              IF .CURRENT_VCB[VCB$V_ENUSEREOT]
;  277     0659  4               THEN
;  278     0660  5                 BEGIN
;  279     0661  5                      PACKET[IRP$L_IOST1] = SS$_ABORT;
;  280     0662  5                      RETURN;
;  281     0663  5                 END
;  282     0664  4               ELSE
;  283     0665  5                  BEGIN
;  284     0666  5                      KERNEL_CALL(INSERT_TAIL, .PACKET);
;  285     0667  5                      IO_PACKET = 0;
;  286     0668  5                      RETURN;
;  287     0669  4                  END;
;  288     0670  3          END;
;  289     0671  3
;  290     0672  3          IF .IO_ERROR NEQ SS$_ENDOFFILE
;  291     0673  3              AND
;  292     0674  3              .IO_ERROR NEQ SS$_ENDOFTAPE
;  293     0675  3          THEN
;  294     0676  4              BEGIN                                   ! return all other errors to user
;  295     0677  4              USER_STATUS[0] = .PACKET[IRP$L_IOST1];
;  296     0678  4              USER_STATUS[1] = .PACKET[IRP$L_IOST2];
;  297     0679  4              KERNEL_CALL(START_VIO);                 ! allow other io to continue
;  298     0680  4              RETURN;                                 ! complete i/o in error
;  299     0681  4
;  300     0682  4              END
;  301     0683  3          ELSE
;  302     0684  4              BEGIN
;  303     0685  4
;  304     0686  4              ! end of volume processing
;  305     0687  4              !
;  306     0688  4
;  307     0689  4      ! Call to this code to ensure that all outstanding IO's on the driver have been
;  308     0690  4      ! completed
;  309     0691  4
;  310     0692  4              CHCK_IO_CLR_EXCP();
;  311     0693  4
```

```
;  312     0694  4 ! Check to see if user eot handling is enabled. If it is then the io must be
;  313     0695  4 ! posted to the user along with all outstanding io's. The user must then
;  314     0696  4 ! issue a command to the ACP to get processing to continue.  If we are reading
;  315     0697  4 ! the tape and the status returned is EOT then complete the request but
;  316     0698  4 ! do not stop processing in the tape. The EOT status should not be returned
;  317     0699  4 ! by the driver while reading a tape.
;  318     0700  4
;  319     0701  4             IF .CURRENT_VCB[VCB$V_ENUSEREOT]
;  320     0702  5                 AND NOT (.CURRENT_WCB[WCB$V_READ]
;  321     0703  5                           AND (.IO_ERROR EQ[ SS$_ENDOFTAPE))
;  322     0704  4             THEN
;  323     0705  5                 BEGIN
;  324     0706  5 ! When user eot handling is set ENDOFFILE and ENDOFVOLUME are not
;  325     0707  5 ! alternate success codes but are the code
;  326     0708  5
;  327     0709  5                 USER_STATUS[0] = .PACKET[IRP$L_IOST1];
;  328     0710  5                 USER_STATUS[1] = .PACKET[IRP$L_IOST2];
;  329     0711  5                 KERNEL_CALL(IO_DONE, .PACKET);
;  330     0712  5                 IO_PACKET = 0 ;                 ! Clear packet address
;  331     0713  5
;  332     0714  5 ! The call to complete_vio here will be useful to the pre-mscp drivers. All
;  333     0715  5 ! oustanding IO should be available to us here therefore we will complete
;  334     0716  5 ! it. Post-mscp drivers IO's should be completed via the SERIOUS EXCP path.
;  335     0717  5 !
;  336     0718  5                 KERNEL_CALL(COMPLETE_VIO);
;  337     0719  5                 RETURN;
;  338     0720  4                 END;
;  339     0721  4
;  340     0722  4             IF NOT .CURRENT_WCB[WCB$V_READ]
;  341     0723  4             THEN
;  342     0724  5                 BEGIN                           ! end of volume on write
;  343     0725  5                 USER_STATUS<16, 16> = .(PACKET[IRP$L_IOST1])<16, 16>;
;  344     0726  5                 USER_STATUS[1] = .PACKET[IRP$L_IOST2];
;  345     0727  5                 NEXT_VOL_WRITE();
;  346     0728  5
;  347     0729  5                 ! this io was written successfully
;  348     0730  5                 !
;  349     0731  5                 KERNEL_CALL(IO_DONE, .PACKET);
;  350     0732  5                 IO_PACKET = 0;
;  351     0733  5                 KERNEL_CALL(START_VIO);
;  352     0734  5                 RETURN;                         ! don't complete io
;  353     0735  5
;  354     0736  5                 END
;  355     0737  4             ELSE
;  356     0738  5                 BEGIN                           ! end of volume on read
;  357     0739  5
;  358     0740  5                 ! if end of tape on read, then make that io successful and
;  359     0741  5                 ! requeue any blocked io.  will eventually hit tape mark
;  360     0742  5                 !
;  361     0743  5
;  362     0744  5                 IF .IO_ERROR EQL SS$_ENDOFTAPE
;  363     0745  5                 THEN
;  364     0746  6                     BEGIN
;  365     0747  6                     KERNEL_CALL(START_VIO);     ! requeue any blocked io
;  366     0748  6                     USER_STATUS<16, 16> = .(PACKET[IRP$L_IOST1])<16, 16>;
;  367     0749  6                     USER_STATUS[1] = .PACKET[IRP$L_IOST2];
;  368     0750  6                     RETURN;                     ! complete current io
```

```
;  369     0751  6          END
;  370     0752  6      ELSE
;  371     0753  5          BEGIN
;  372     0754  6
;  373     0755  6          ! calculate the number of blocks since last tape mark.  this
;  374     0756  6          ! will be compared with the number in the trailer record.
;  375     0757  6          ! since a tape mark triggered end_of_vol processing, record
;  376     0758  6          ! it.  then discover if this is the end_of_vol or end
;  377     0759  6          ! _of_file.
;  378     0760  6          !
;  379     0761  6          BLOCKS1 = .CURRENT_UCB[UCB$L_RECORD] - 1 -
;  380     0762  6          .CURRENT_VCB[VCB$L_ST_RECORD];
;  381     0763  6
;  382     0764  6          IF .BLOCKS1 GEQ 0
;  383     0765  6          THEN
;  384     0766  6          BEGIN
;  385     0767  7
;  386     0768  7          ! tape mark triggered end_of_file
;  387     0769  7          !
;  388     0770  7          KERNEL_CALL(ADJTM, 1);
;  389     0771  7
;  390     0772  7          IF NOT READ_BLOCK(.HDR1, ANSI_LBLSZ)
;  391     0773  7              OR
;  392     0774  7              (.HDR1[EO1$L_EO1LID] NEQ 'EOF1'
;  393     0775  8              AND
;  394     0776  8              .HDR1[EO1$L_EO1LID] NEQ 'EOV1')
;  395     0777  8          THEN
;  396     0778  7              ERR_EXIT(SS$_TAPEPOSLOST);
;  397     0779  7
;  398     0780  7          ! convert the number of blocks recored in the trailer
;  399     0781  7          ! record   compare it with the number read.  if the numbers
;  400     0782  7          ! are not equal then return all virtual io to the user in
;  401     0783  7          ! error.
;  402     0784  7          !
;  403     0785  7          LIB$CVT_DTB(EO1$S_BLOCKCNT, HDR1[EO1$T_BLOCKCNT], BLOCKS2);
;  404     0786  7
;  405     0787  7          IF .BLOCKS1 NEQ .BLOCKS2
;  406     0788  7          THEN
;  407     0789  7              BBLOCK[.CURRENT_VCB[VCB$L_VPFL], VVP$L_BLOCKDIF] =
;  408     0790  7              .BBLOCK[.CURRENT_VCB[VCB$L_VPFL], VVP$L_BLOCKDIF] +
;  409     0791  7              .BLOCKS1 - .BLOCKS2;
;  410     0792  7
;  411     0793  7          ! if end_of_file read, return all physical io in error,
;  412     0794  7          ! signal user to read user labels now and return
;  413     0795  7          !
;  414     0796  7
;  415     0797  7          IF .HDR1[EO1$L_EO1LID] EQL 'EOF1'
;  416     0798  7          THEN
;  417     0799  7              ERR_EXIT(SS$_ENDOFFILE);
;  418     0800  7
;  419     0801  7          ! end of volume processing
;  420     0802  7          !
;  421     0803  7          NEXT_VOL_READ();            ! get next volume on read
;  422     0804  7
;  423     0805  7          ! put in blocked io queue
;  424     0806  7          !
;  425     0807  7
```

```
;  426     0808  7                    KERNEL_CALL(INSERT_HEAD, .PACKET);
;  427     0809  7                    IO_PACKET = 0;                  ! don't complete io
;  428     0810  7                    KERNEL_CALL(START_VIO);         ! requeue blocked io
;  429     0811  7                    RETURN;
;  430     0812  7
;  431     0813  7                    END
;  432     0814  7
;  433     0815  7 ! Scan the tape backwards to find the HDR1 label. Use the HDR1 label to
;  434     0816  7 ! determine if this is beginning of tape or beginning of volume.
;  435     0817  7 ! If while scanning backwards the VOL1 label is received then there
;  436     0818  7 ! is an error or the tape postion was lost.
;  437     0819  7 !
;  438     0820  6                    ELSE
;  439     0821  7                        BEGIN
;  440     0822  7                        WHILE 1 DO
;  441     0823  8                           BEGIN
;  442     0824  8                           IF NOT READ_BLOCK_REVERSE(.HDR1, ANSI_LBLSZ)
;  443     0825  8                              OR
;  444     0826  9                                 (.HDR1[HD1$L_HD1LID] EQL 'VOL1')
;  445     0827  8                           THEN
;  446     0828  8                                ERR_EXIT(SS$_TAPEPOSLOST);
;  447     0829  8
;  448     0830  8                           IF .HDR1[HD1$L_HD1LID] EQL 'HDR1'
;  449     0831  8                           THEN EXITLOOP;          ! if eql then found first label
;  450     0832  7                           END;
;  451     0833  7
;  452     0834  7 ! If eql one then first file section beginnng of file. Not one other
;  453     0835  7 ! file sections are on other volumes mount the volume and position to
;  454     0836  7 ! the end of the volume want last record of last file.
;  455     0837  7 !
;  456     0838  7                           TM = 0;
;  457     0839  7                           KERNEL_CALL(RESTORE_POS, .TM, .CURRENT_UCB[UCB$L_RECORD]);
;  458     0840  7                           IF .HDR1[HD1$T_FILESECNO] EQL 1
;  459     0841  7                           THEN
;  460     0842  8                              ERR_EXIT (SS$_BEGOFFILE)
;  461     0843  7                           ELSE
;  462     0844  7                              !& temp until I figure what to do
;  463     0845  8                              ERR_EXIT (SS$_BEGOFFILE)
;  464     0846  6                           END;                            ! end of BOF check else
;  465     0847  6
;  466     0848  5                       END;                                ! end of read EOT check else
;  467     0849  5
;  468     0850  4                   END;                            ! if end_of_file or end_of_tape on read
;  469     0851  4
;  470     0852  3               END;                                        ! end of if write else read
;  471     0853  3
;  472     0854  2           END;                                            ! end of if map else error
;  473     0855  2
;  474     0856  1     END;                                                  ! end of routine


                                                .TITLE  ENDVOL
                                                .IDENT  \V04-000\

                                                .EXTRN  ADJTM, CHCK_IO_CLR_EXCP
                                                .EXTRN  COMPLETE_VIO, GTNEXT_VOL_READ
                                                .EXTRN  GTNEXT_VOL_WRIT
```

```
                                                .EXTRN  IO_DONE, LIB$CVT_DTB
                                                .EXTRN  READ_BLOCK, READ_BLOCK_REVERSE
                                                .EXTRN  RETURN_ALL_ERR, SPACE_TM
                                                .EXTRN  START_VIO, STOP_VIO
                                                .EXTRN  RESTORE_POS, REPOSITION
                                                .EXTRN  WRITE_HEADERS, WRITE_TM
                                                .EXTRN  WRITE_TRAILERS, CURRENT_UCB
                                                .EXTRN  CURRENT_WCB, HDR1
                                                .EXTRN  IO_PACKET, USER_STATUS
                                                .EXTRN  SYS$CMKRNL

                                                .PSECT  $CODE$,NOWRT,2

                              0000 00000        .ENTRY  END_OF_VOL, Save nothing            ; 0523
            5E            04  C2 00002          SUBL2   #4, SP                              ;
                    0000G CF  DD 00005          PUSHL   IO_PACKET                           ; 0566
7E          6E            2A  C1 00009          ADDL3   #42, PACKET, -(SP)                  ; 0573
16          9E            04  E1 0000D          BBC     #4, @(SP)+, 2$                      ;
04      0B  AB            06  E1 00011          BBC     #6, 11(CURRENT_VCB), 1$             ; 0581
                    0948  8F  BF 00016          CHMU    #2376                               ; 0583
   31464F45 8F      0000G DF  D1 0001A 1$:      CMPL    @HDR1, #826691397                   ; 0589
                          47  13 00023          BEQL    4$                                  ;
                          49  11 00025          BRB     5$                                  ; 0598
                          7E  D4 00027 2$:      CLRL    -(SP)                               ; 0625
                          5E  DD 00029          PUSHL   SP                                  ;
                    0000G CF  9F 0002B          PUSHAB  STOP_VIO                            ;
   00000000G 9F           03  FB 0002F          CALLS   #3, @#SYS$CMKRNL                    ;
50          6E            38  C1 00036          ADDL3   #56, PACKET, R0                     ; 0626
            52            60  3C 0003A          MOVZWL  (R0), IO_ERROR                      ;
   000001A4 8F            52  D1 0003D          CMPL    IO_ERROR, #420                      ; 0628
                          12  12 00044          BNEQ    3$                                  ;
            50      0000G CF  D0 00046          MOVL    CURRENT_UCB, R0                     ; 0631
                    00B0  C0  DD 0004B          PUSHL   176(R0)                             ;
                        0000G 30 0004F          BSBW    REPOSITION                          ;
            5E            04  C0 00052          ADDL2   #4, SP                              ;
                        018F  31 00055          BRW     18$                                 ; 0632
   000021D4 8F            52  D1 00058 3$:      CMPL    IO_ERROR, #8660                     ; 0639
                          32  12 0005F          BNEQ    7$                                  ;
   31464F45 8F      0000G DF  D1 00061          CMPL    @HDR1, #826691397                   ; 0650
                          04  12 0006A          BNEQ    5$                                  ;
                    0870  8F  BF 0006C 4$:      CHMU    #2160                               ; 0652
08      2D  AB            01  E1 00070 5$:      BBC     #1, 45(CURRENT_VCB), 6$             ; 0658
50          6E            38  C1 00075          ADDL3   #56, PACKET, R0                     ; 0661
            60            2C  D0 00079          MOVL    #44, (R0)                           ;
                          04     0007C          RET                                         ; 0665
                          6E  DD 0007D 6$:      PUSHL   PACKET                              ; 0666
                          01  DD 0007F          PUSHL   #1                                  ;
                          5E  DD 00081          PUSHL   SP                                  ;
                    0000V CF  9F 00083          PUSHAB  INSERT_TAIL                         ;
   00000000G 9F           04  FB 00087          CALLS   #4, @#SYS$CMKRNL                    ;
                    0000G CF  D4 0008E          CLRL    IO_PACKET                           ; 0667
                          04     00092          RET                                         ; 0665
   00000870 8F            52  D1 00093 7$:      CMPL    IO_ERROR, #2160                     ; 0672
                          1E  13 0009A          BEQL    8$                                  ;
   00000878 8F            52  D1 0009C          CMPL    IO_ERROR, #2168                     ; 0674
                          15  13 000A3          BEQL    8$                                  ;
50          6E            38  C1 000A5          ADDL3   #56, PACKET, R0                     ; 0677
```

```
               0000G  CF        60  D0 000A9       MOVL    (R0), USER_STATUS                      :
        50            6E        3C  C1 000AE       ADDL3   #60, PACKET, R0                        : 0678
               0000G  CF        60  D0 000B2       MOVL    (R0), USER_STATUS+4                    :
                              0142  31 000B7       BRW     20$                                    : 0679
               0000G  CF        00  FB 000BA  8$:  CALLS   #0, CHCK_IO_CLR_EXCP                   : 0692
        44        2D  AB        01  E1 000BF       BBC     #1, 45(CURRENT_VCB), 10$               : 0701
                      50  0000G CF  D0 000C4       MOVL    CURRENT_WCB, R0                        : 0702
                      09     0B A0  E9 000C9       BLBC    11(R0), 9$                             :
            00000878  8F        52  D1 000CD       CMPL    IO_ERROR, #2168                        : 0703
                                32  13 000D4       BEQL    10$                                    :
        50            6E        38  C1 000D6  9$:  ADDL3   #56, PACKET, R0                        : 0709
               0000G  CF        60  D0 000DA       MOVL    (R0), USER_STATUS                      :
        50            6E        3C  C1 000DF       ADDL3   #60, PACKET, R0                        : 0710
               0000G  CF        60  D0 000E3       MOVL    (R0), USER_STATUS+4                    :
                                6E  DD 000E8       PUSHL   PACKET                                 : 0711
                                01  DD 000EA       PUSHL   #1                                     :
                                5E  DD 000EC       PUSHL   SP                                     :
                          0000G CF  9F 000EE       PUSHAB  IO_DONE                                :
           00000000G  9F        04  FB 000F2       CALLS   #4, @#SYS$CMKRNL                       :
                          0000G CF  D4 000F9       CLRL    IO_PACKET                              : 0712
                                7E  D4 000FD       CLRL    -(SP)                                  : 0718
                                5E  DD 000FF       PUSHL   SP                                     :
                          0000G CF  9F 00101       PUSHAB  COMPLETE_VIO                           :
                              00FC  31 00105       BRW     21$                                    :
                      50  0000G CF  D0 00108 10$:  MOVL    CURRENT_WCB, R0                        : 0722
                      22     0B A0  E8 0010D       BLBS    11(R0), 11$                            :
        50            6E        3A  C1 00111       ADDL3   #58, PACKET, R0                        : 0725
               0000G  CF        60  B0 00115       MOVW    (R0), USER_STATUS+2                    :
        50            6E        3C  C1 0011A       ADDL3   #60, PACKET, R0                        : 0726
               0000G  CF        60  D0 0011E       MOVL    (R0), USER_STATUS+4                    :
                             0000V  30 00123       BSBW    NEXT_VOL_WRITE                         : 0727
                                6E  DD 00126       PUSHL   PACKET                                 : 0731
                                01  DD 00128       PUSHL   #1                                     :
                                5E  DD 0012A       PUSHL   SP                                     :
                          0000G CF  9F 0012C       PUSHAB  IO_DONE                                :
                              00BE  31 00130       BRW     19$                                    :
            00000878  8F        52  D1 00133 11$:  CMPL    IO_ERROR, #2168                        : 0744
                                22  12 0013A       BNEQ    12$                                    :
                                7E  D4 0013C       CLRL    -(SP)                                  : 0747
                                5E  DD 0013E       PUSHL   SP                                     :
                          0000G CF  9F 00140       PUSHAB  START_VIO                              :
           00000000G  9F        03  FB 00144       CALLS   #3, @#SYS$CMKRNL                       :
        50            6E        3A  C1 0014B       ADDL3   #58, PACKET, R0                        : 0748
               0000G  CF        60  B0 0014F       MOVW    (R0), USER_STATUS+2                    :
        50            6E        3C  C1 00154       ADDL3   #60, PACKET, R0                        : 0749
               0000G  CF        60  D0 00158       MOVL    (R0), USER_STATUS+4                    :
                                    04 0015D       RET                                            : 0746
                      52  0000G CF  D0 0015E 12$:  MOVL    CURRENT_UCB, R2                        : 0762
        52      0080  C2     30 AB  C3 00163       SUBL3   48(CURRENT_VCB), 176(R2), R2           : 0763
                                52  D7 0016A       DECL    BLOCKS1                                : 0762
                                03  18 0016C       BGEQ    13$                                    : 0765
                              009B  31 0016E       BRW     22$                                    :
                                01  DD 00171 13$:  PUSHL   #1                                     : 0771
                                01  DD 00173       PUSHL   #1                                     :
                                5E  DD 00175       PUSHL   SP                                     :
                          0000G CF  9F 00177       PUSHAB  ADJTM                                  :
           00000000G  9F        04  FB 0017B       CALLS   #4, @#SYS$CMKRNL                       :
```

```
                     7E         50  8F  9A 00182          MOVZBL  #80, -(SP)                         ; 0773
                             0000G  CF  DD 00186          PUSHL   HDR1                               ;
              0000G  CF             02  FB 0018A          CALLS   #2, READ_BLOCK                     ;
                     16             50  E9 0018F          BLBC    R0, 14$                            ;
           31464F45  8F      ^000G  DF  D1 00192          CMPL    @HDR1, #826691397                  ; 0775
                                    0F  13 0019B          BEQL    15$                                ;
           31564F45  8F      0000G  DF  D1 0019D          CMPL    @HDR1, #827739973                  ; 0777
                                    04  13 001A6          BEQL    15$                                ;
                              0224  8F  BF 001A8  14$:    CHMU    #548                               ; 0779
                                04  AE  9F 001AC  15$:    PUSHAB  BLOCKS2                            ; 0786
          7E  0000G  CF                 36  C1 001AF          ADDL3   #54, HDR1, -(SP)                   ;
                                    06  DD 001B5          PUSHL   #6                                 ;
          00000000G  9F             03  FB 001B7          CALLS   #3, @#LIB$CVT_DTB                  ;
                 04  AE             52  D1 001BE          CMPL    BLOCKS1, BLOCKS2                   ; 0788
                                    11  13 001C2          BEQL    16$                                ;
                     50         3C  AB  D0 001C4          MOVL    60(CURRENT_VCB), R0                ; 0790
          51         52       01AC  C0  C1 001C8          ADDL3   428(R0), BLOCKS1, R1               ; 0792
  01AC    C0         51         04  AE  C3 001CE          SUBL3   BLOCKS2, R1, 428(R0)               ;
           31464F45  8F      0000G  DF  D1 001D5  16$:    CMPL    @HDR1, #826691397                  ; 0798
                                    04  12 001DE          BNEQ    17$                                ;
                              0870  8F  BF 001E0          CHMU    #2160                              ; 0800
                                 0000V  30 001E4  17$:    BSBW    NEXT_VOL_READ                      ; 0804
                                    6E  DD 001E7  18$:    PUSHL   PACKET                             ; 0808
                                    01  DD 001E9          PUSHL   #1                                 ;
                                    5E  DD 001EB          PUSHL   SP                                 ;
                             0000V  CF  9F 001ED          PUSHAB  INSERT_HEAD                        ;
          00000000G  9F             04  FB 001F1  19$:    CALLS   #4, @#SYS$CMKRNL                   ;
                             0000G  CF  D4 001F8          CLRL    IO_PACKET                          ; 0809
                                    7E  D4 001FC  20$:    CLRL    -(SP)                              ; 0810
                                    5E  DD 001FE          PUSHL   SP                                 ;
                             0000G  CF  9F 00200          PUSHAB  START_VIO                          ;
          00000000G  9F             03  FB 00204  21$:    CALLS   #3, @#SYS$CMKRNL                   ;
                                        04 0020B          RET                                        ; 0767
                     52      0000G  CF  D0 0020C  22$:    MOVL    HDR1, R2                           ; 0824
                     7E         50  8F  9A 00211  23$:    MOVZBL  #80, -(SP)                         ;
                                    52  DD 00215          PUSHL   R2                                 ;
              0000G  CF             02  FB 00217          CALLS   #2, READ_BLOCK_REVERSE             ;
                     0B             50  E9 0021C          BLBC    R0, 24$                            ;
           314C4F56  8F      0000G  DF  D1 0021F          CMPL    @HDR1, #827084630                  ; 0826
                                    04  12 00228          BNEQ    25$                                ;
                              0224  8F  BF 0022A  24$:    CHMU    #548                               ; 0828
                     52      0000G  CF  D0 0022E  25$:    MOVL    HDR1, R2                           ; 0830
           31524448  8F             62  D1 00233          CMPL    (R2), #827475016                   ;
                                    D5  12 0023A          BNEQ    23$                                ;
                                    51  D4 0023C          CLRL    TM                                 ; 0838
                     50      0000G  CF  D0 0023E          MOVL    CURRENT_UCB, R0                    ; 0839
                             00B0   C0  DD 00243          PUSHL   176(R0)                            ;
                                    51  DD 00247          PUSHL   TM                                 ;
                                    02  DD 00249          PUSHL   #2                                 ;
                                    5E  DD 0024B          PUSHL   SP                                 ;
                             0000G  CF  9F 0024D          PUSHAB  RESTORE_POS                        ;
          00000000G  9F             05  FB 00251          CALLS   #5, @#SYS$CMKRNL                   ;
                     50      0000G  CF  D0 00258          MOVL    HDR1, R0                           ; 0840
                              0938  8F  BF 0025D          CHMU    #2360                              ; 0845
                                        04 00261          RET                                        ; 0856

; Routine Size:  610 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  475          0857  1
```

```
;  477          0858  1 ROUTINE INSERT_HEAD (PACKET) : COMMON_CALL NOVALUE =
;  478          0859  1
;  479          0860  1 !++
;  480          0861  1 !
;  481          0862  1 ! FUNCTIONAL DESCRIPTION:
;  482          0863  1 !       this routine inserts io packet at the header of the blocked io queue
;  483          0864  1 !
;  484          0865  1 ! CALLING SEQUENCE:
;  485          0866  1 !       insert_head(arg1), called in kernel mode
;  486          0867  1 !
;  487          0868  1 ! INPUT PARAMETERS:
;  488          0869  1 !       arg1 - address of io request packet
;  489          0870  1 !
;  490          0871  1 ! IMPLICIT INPUTS:
;  491          0872  1 !       current_vcb     - address of current volume control block
;  492          0873  1 !
;  493          0874  1 ! OUTPUT PARAMETERS:
;  494          0875  1 !       none
;  495          0876  1 !
;  496          0877  1 ! IMPLICIT OUTPUTS:
;  497          0878  1 !       none
;  498          0879  1 !
;  499          0880  1 ! ROUTINE VALUE:
;  500          0881  1 !       none
;  501          0882  1 !
;  502          0883  1 ! SIDE EFFECTS:
;  503          0884  1 !       packet inserted at head of queue
;  504          0885  1 !
;  505          0886  1 !--
;  506          0887  1
;  507          0888  2     BEGIN
;  508          0889  2
;  509          0890  2     EXTERNAL REGISTER
;  510          0891  2         COMMON_REG;
;  511          0892  2
;  512          0893  2     ! clobber nmap until this function is in driver
;  513          0894  2     !
;  514          0895  2     INSQUE(.PACKET, CURRENT_VCB[VCB$L_BLOCKFL]);
;  515          0896  1     END;                                    ! end of routine


                                        0000 00000 INSERT_HEAD:
                                                   .WORD   Save nothing                          ; 0858
                              6B    04  BC  0E 00002         INSQUE  @PACKET, (CURRENT_VCB)      ; 0895
                                        04 00006         RET                                   ; 0896

; Routine Size:  7 bytes,    Routine Base:  $CODE$ + 0262
```

```
;  517          0897  1 ROUTINE INSERT_TAIL (PACKET) : COMMON_CALL NOVALUE =
;  518          0898  1
;  519          0899  1 !++
;  520          0900  1 !
;  521          0901  1 ! FUNCTIONAL DESCRIPTION:
;  522          0902  1 !       this routine inserts the packet in the tail of the blocked io request
;  523          0903  1 !       queue
;  524          0904  1 !
;  525          0905  1 ! CALLING SEQUENCE:
;  526          0906  1 !       insert_tail(arg1), called in kernel mode
;  527          0907  1 !
;  528          0908  1 ! INPUT PARAMETERS:
;  529          0909  1 !       none
;  530          0910  1 !
;  531          0911  1 ! IMPLICIT INPUTS:
;  532          0912  1 !       current_vcb     - address of current volume control block
;  533          0913  1 !
;  534          0914  1 ! OUTPUT PARAMETERS:
;  535          0915  1 !       none
;  536          0916  1 !
;  537          0917  1 ! IMPLICIT OUTPUTS:
;  538          0918  1 !       none
;  539          0919  1 !
;  540          0920  1 ! ROUTINE VALUE:
;  541          0921  1 !       none
;  542          0922  1 !
;  543          0923  1 ! SIDE EFFECTS:
;  544          0924  1 !       none
;  545          0925  1 !
;  546          0926  1 !--
;  547          0927  1
;  548          0928  2     BEGIN
;  549          0929  2
;  550          0930  2     EXTERNAL REGISTER
;  551          0931  2         COMMON_REG;
;  552          0932  2
;  553          0933  2     INSQUE(.PACKET, .CURRENT_VCB[VCB$L_BLOCKBL]);
;  554          0934  1     END;                                    ! end of routine



                                        0000 00000 INSERT_TAIL:
                                                   .WORD   Save nothing                          ; 0897
                            04   BB      04   BC  0E 00002       INSQUE  @PACKET, @4(CURRENT_VCB)  ; 0933
                                                04 00007       RET                               ; 0934

; Routine Size:  8 bytes,    Routine Base:  $CODE$ + 0269

;  555          0935  1
```

```
;  557          0936  1 GLOBAL ROUTINE NEXT_VOL_WRITE : L$NEXT_VOL_WRIT NOVALUE =
;  558          0937  1
;  559          0938  1 !++
;  560          0939  1 !
;  561          0940  1 ! FUNCTIONAL DESCRIPTION:
;  562          0941  1 !       This routine writes end of volume trailers, gets the next volume
;  563          0942  1 !       for write, and writes the header labels.
;  564          0943  1 !
;  565          0944  1 ! CALLING SEQUENCE:
;  566          0945  1 !       NEXT_VOL_WRITE()
;  567          0946  1 !
;  568          0947  1 ! INPUT PARAMETERS:
;  569          0948  1 !       none
;  570          0949  1 !
;  571          0950  1 ! IMPLICIT INPUTS:
;  572          0951  1 !       blocked io queue
;  573          0952  1 !
;  574          0953  1 ! OUTPUT PARAMETERS:
;  575          0954  1 !       none
;  576          0955  1 !
;  577          0956  1 ! IMPLICIT OUTPUTS:
;  578          0957  1 !       io requeue to next volume
;  579          0958  1 !
;  580          0959  1 ! ROUTINE VALUE:
;  581          0960  1 !       none
;  582          0961  1 !
;  583          0962  1 ! SIDE EFFECTS:
;  584          0963  1 !       none
;  585          0964  1 !
;  586          0965  1 ! USER ERRORS:
;  587          0966  1 !       none
;  588          0967  1 !
;  589          0968  1 !--
;  590          0969  1
;  591          0970  2     BEGIN
;  592          0971  2
;  593          0972  2     EXTERNAL REGISTER
;  594          0973  2         COMMON_REG;
;  595          0974  2
;  596          0975  2     WRITE_TRAILERS('V');                        ! write trailers
;  597          0976  2     WRITE_TM();
;  598          0977  2     WRITE_TM();
;  599          0978  2
;  600          0979  2     GTNEXT_VOL_WRIT();                          ! get the next volume for write
;  601          0980  2     WRITE_READERS();                            ! write hdr1 and hdr2
;  602          0981  2
;  603          0982  2     ! Close out header label set with tape mark
;  604          0983  2     !
;  605          0984  2
;  606          0985  2     WRITE_TM();
;  607          0986  1     END;


                                7E       56    8F   9A 00000 NEXT_VOL_WRITE::
```

```
                                                MOVZBL  #86, -(SP)                      ; 0975
                        0000G 30 00004          BSBW    WRITE_TRAILERS                  ;
                     5E   04  C0 00007          ADDL2   #4, SP                          ;
                        0000G 30 0000A          BSBW    WRITE_TM                        ; 0976
                        0000G 30 0000D          BSBW    WRITE_TM                        ; 0977
                        0000G 30 00010          BSBW    GTNEXT_VOL_WRIT                 ; 0979
                        0000G 30 00013          BSBW    WRITE_HEADERS                   ; 0980
                        0000G 31 00016          BRW     WRITE_TM                        ; 0985

; Routine Size:  25 bytes,    Routine Base:  $CODE$ + 0271

;  608          0987  1
```

```
;  610          0988  1 GLOBAL ROUTINE NEXT_VOL_READ : L$NEXT_VOL_READ NOVALUE =
;  611          0989  1
;  612          0990  1 !++
;  613          0991  1 !
;  614          0992  1 ! FUNCTIONAL DESCRIPTION:
;  615          0993  1 !       This routine positions the current volume at the very end, gets the
;  616          0994  1 !       next volume for read, verifies the headers and positions the tape at
;  617          0995  1 !       the beginning of the data section.
;  618          0996  1 !
;  619          0997  1 ! CALLING SEQUENCE:
;  620          0998  1 !       NEXT_VOL_READ()
;  621          0999  1 !
;  622          1000  1 ! INPUT PARAMETERS:
;  623          1001  1 !       none
;  624          1002  1 !
;  625          1003  1 ! IMPLICIT INPUTS:
;  626          1004  1 !       blocked io queue
;  627          1005  1 !
;  628          1006  1 ! OUTPUT PARAMETERS:
;  629          1007  1 !       none
;  630          1008  1 !
;  631          1009  1 ! IMPLICIT OUTPUTS:
;  632          1010  1 !       requeue blocked io to next volume
;  633          1011  1 !
;  634          1012  1 ! ROUTINE VALUE:
;  635          1013  1 !       none
;  636          1014  1 !
;  637          1015  1 ! SIDE EFFECTS:
;  638          1016  1 !       none
;  639          1017  1 !
;  640          1018  1 ! USER ERRORS:
;  641          1019  1 !       none
;  642          1020  1 !
;  643          1021  1 !--
;  644          1022  1
;  645          1023  2     BEGIN
;  646          1024  2
;  647          1025  2     EXTERNAL REGISTER
;  648          1026  2         COMMON_REG;
;  649          1027  2
;  650          1028  2     ! If user labels are read, a tape mark could be read, which makes TM=0
;  651          1029  2     !
;  652          1030  2
;  653          1031  2     SPACE_TM(IF .CURRENT_VCB[VCB$B_TM] EQL 0 THEN 1 ELSE 2);
;  654          1032  2
;  655          1033  2     GTNEXT_VOL_READ();                          ! get the next volume for read
;  656          1034  2
;  657          1035  2     IF .CURRENT_VCB[VCB$B_TM] EQL 0
;  658          1036  2     THEN
;  659          1037  2         SPACE_TM(1);
;  660          1038  2
;  661          1039  1     END;
```

```
                                 2E    AB  95 00000 NEXT_VOL_READ::
                                                     TSTB    46(CURRENT_VCB)                 ; 1031
                                      04  12 00003          BNEQ    1$
                                      01  DD 00005          PUSHL   #1
                                      02  11 00007          BRB     2$
                                      02  DD 00009 1$:      PUSHL   #2
                           0000G      CF  01  FB 0000B 2$:   CALLS   #1, SPACE_TM
                                    0000G 30 00010          BSBW    GTNEXT_VOL_READ         ; 1033
                                 2E    AB  95 00013          TSTB    46(CURRENT_VCB)         ; 1035
                                      07  12 00016          BNEQ    3$
                                      01  DD 00018          PUSHL   #1                      ; 1037
                           0000G      CF  01  FB 0001A      CALLS   #1, SPACE_TM
                                          05 0001F 3$:      RSB                             ; 1039

; Routine Size:  32 bytes,    Routine Base:  $CODE$ + 028A

;  662          1040  1
;  663          1041  1 END
;  664          1042  1
;  665          1043  0 ELUDOM



;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $CODE$                        682  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                                 Library Statistics
;
;                                          -------- Symbols --------      Pages       Processing
;       File                               Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1            18619        29         0        1000        00:01.8



;                                       COMMAND QUALIFIERS
;
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:ENDVOL/OBJ=OBJ$:ENDVOL MSRC$:ENDVOL/UPDATE=(ENH$:ENDVOL)

; Size:          682 code + 0 data bytes
; Run Time:        00:17.1
; Elapsed Time:    00:50.7
; Lines/CPU Min:    3661
```

```
; Lexemes/CPU-Min: 18540
; Memory Used:  211 pages
; Compilation Complete
```

FREEPG
LIS

DEACCS
LIS

FIND
LIS

DATECN
LIS

FRMOD1
LIS

CREATE
LIS

GETREQ
LIS

EXPIRE
LIS

CNTRL
LIS

FRMHDR
LIS

HEADER
LIS

COMLABPRC
LIS

ENDVOL
LIS

GETFIB
LIS