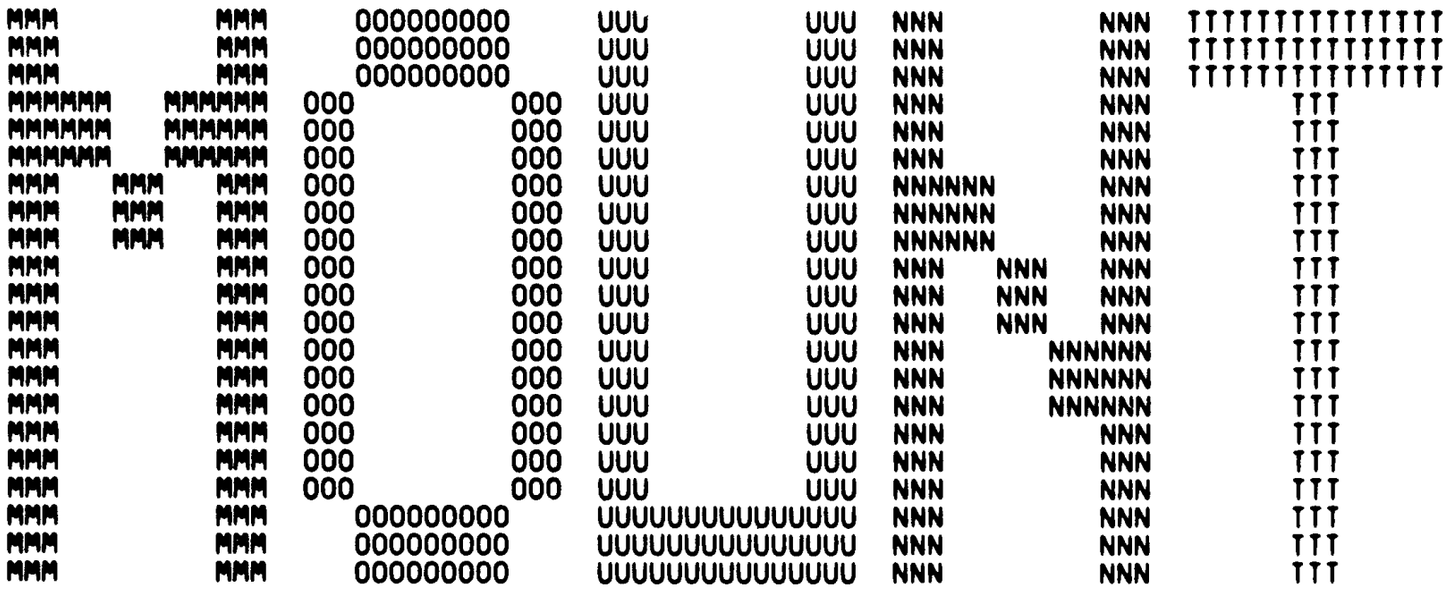
MMM MMM 000000000 UUU UUU NNN NNN TTTTTTTTTTTTTTT
MMM MMM 000000000 UUU UUU NNN NNN TTTTTTTTTTTTTTT
MMM MMM 000000000 UUU UUU NNN NNN TTTTTTTTTTTTTTT
MMMMMMM MMMMMMM 000 000 UUU UUU NNN NNN TTT
MMMMMMM MMMMMMM 000 000 UUU UUU NNN NNN TTT
MMMMMMM MMMMMMM 000 000 UUU UUU NNN NNN TTT
MMM MMM MMM 000 000 UUU UUU NNNNNN NNN TTT
MMM MMM MMM 000 000 UUU UUU NNNNNN NNN TTT
MMM MMM MMM 000 000 UUU UUU NNNNNN NNN TTT
MMM MMM 000 000 UUU UUU NNN NNN NNN TTT
MMM MMM 000 000 UUU UUU NNN NNN NNN TTT
MMM MMM 000 000 UUU UUU NNN NNN NNN TTT
MMM MMM 000 000 UUU UUU NNN NNNNNN TTT
MMM MMM 000 000 UUU UUU NNN NNNNNN TTT
MMM MMM 000 000 UUU UUU NNN NNNNNN TTT
MMM MMM 000 000 UUU UUU NNN NNN TTT
MMM MMM 000 000 UUU UUU NNN NNN TTT
MMM MMM 000 000 UUU UUU NNN NNN TTT
MMM MMM 000000000 UUUUUUUUUUUUUUUU NNN NNN TTT
MMM MMM 000000000 UUUUUUUUUUUUUUUU NNN NNN TTT
MMM MMM 000000000 UUUUUUUUUUUUUUUU NNN NNN TTT
MMM MMM 000000000 UUUUUUUUUUUUUUUU NNN NNN TTT

FILEID**ERASE

```
EEEEEEEEEE  RRRRRRRR      AAAAAA      SSSSSSSS  EEEEEEEEEE
EEEEEEEEEE  RRRRRRRR      AAAAAA      SSSSSSSS  EEEEEEEEEE
EE          RR      RR  AA      AA  SS          EE
EE          RR      RR  AA      AA  SS          EE
EE          RR      RR  AA      AA  SS          EE
EE          RR      RR  AA      AA  SS          EE
EEEEEEEE    RRRRRRRR    AA      AA    SSSSSS    EEEEEEEE
EEEEEEEE    RRRRRRRR    AA      AA    SSSSSS    EEEEEEEE
EE          RR  RR      AAAAAAAAAA          SS  EE
EE          RR  RR      AAAAAAAAAA          SS  EE
EE          RR    RR    AA      AA          SS  EE                ....
EE          RR    RR    AA      AA          SS  EE                ....
EEEEEEEEEE  RR      RR  AA      AA  SSSSSSSS    EEEEEEEEEE        ....
EEEEEEEEEE  RR      RR  AA      AA  SSSSSSSS    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
```

GET
V04

```
;     1          0001  0 MODULE ERASE (
;     2          0002  0                 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 !*                                                                           *
;    10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    12          0012  1 !*  ALL RIGHTS RESERVED.                                                      *
;    13          0013  1 !*                                                                           *
;    14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    19          0019  1 !*  TRANSFERRED.                                                              *
;    20          0020  1 !*                                                                           *
;    21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    23          0023  1 !*  CORPORATION.                                                              *
;    24          0024  1 !*                                                                           *
;    25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    27          0027  1 !*                                                                           *
;    28          0028  1 !*                                                                           *
;    29          0029  1 !*****************************************************************************
;    30          0030  1
;    31          0031  1 !++
;    32          0032  1 !
;    33          0033  1 ! FACILITY:  MOUNT System Service
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !
;    37          0037  1 !       This module contains the routines that perform the Data Security
;    38          0038  1 !       Erase (DSE) on a portion of a disk volume.
;    39          0039  1 !
;    40          0040  1 ! ENVIRONMENT:
;    41          0041  1 !
;    42          0042  1 !       STARLET operating system, including privileged system services
;    43          0043  1 !       and internal exec routines.
;    44          0044  1 !
;    45          0045  1 !--
;    46          0046  1 !
;    47          0047  1 !
;    48          0048  1 ! AUTHOR:  Steven T. Jeffreys,   CREATION DATE:  23-Mar-1983
;    49          0049  1 !
;    50          0050  1 ! MODIFIED BY:
;    51          0051  1 !
;    52          0052  1 !       V03-003 ACG0378         Andrew C. Goldstein,    6-Dec-1983  18:28
;    53          0053  1 !               Move to MOUNT facility
;    54          0054  1 !
;    55          0055  1 !       V03-002 STJ3104         Steven T. Jeffreys,     03-Jun-1983
;    56          0056  1 !               - Removed reference to VMSD2.
;    57          0057  1 !
```

```
;    58          0058  1 !       V03-001 STJ3082         Steven T. Jeffreys,     30-Mar-1983
;    59          0059  1 !               - Added CHANNEL parameter to ERASE_BLOCKS and DO_ERASE.
;    60          0060  1 !                 This makes for a cleaner interface with callers outside
;    61          0061  1 !                 of the Files-11 ACP.  (eg. REBUILD and ANALYZE/DISK)
;    62          0062  1 !               - Use VMSD2 to enable/disable erase. (Temporary)
;    63          0063  1 !**
;    64          0064  1
;    65          0065  1
;    66          0066  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    67          0067  1 REQUIRE 'SRC$:MOUDEF.B32';
;    68          0599  1
;    69          0600  1 !
;    70          0601  1 ! Table of contents.
;    71          0602  1 !
;    72          0603  1
;    73          0604  1 FORWARD ROUTINE
;    74          0605  1         ERASE_BLOCKS,                           ! Top level DSE routine
;    75          0606  1         DO_ERASE;                               ! Issues the erase $QIO
```

```
;    77          0607  1 GLOBAL ROUTINE ERASE_BLOCKS (START_LBN, BLOCK_COUNT, CHANNEL) =
;    78          0608  1
;    79          0609  1 !++
;    80          0610  1 !
;    81          0611  1 ! FUNCTIONAL DESCRIPTION:
;    82          0612  1 !
;    83          0613  1 !       Perform a data security erase (DSE) on a single contiguous extent
;    84          0614  1 !       on the disk.  This routine is recursive.
;    85          0615  1 !
;    86          0616  1 !
;    87          0617  1 !       The DSE is done by calling the erase pattern generator system service,
;    88          0618  1 !       $ERAPAT, and writing the pattern returned to the disk.  This is repeated
;    89          0619  1 !       until $ERAPAT returns a status of SS$_NOTRAN, which indicates that the
;    90          0620  1 !       DSE is complete.
;    91          0621  1 !
;    92          0622  1 !       The $ERAPAT code is loadable, and may vary from site to site.
;    93          0623  1 !       However, the default $ERAPAT code is very simple, and by checking to see
;    94          0624  1 !       if the default $ERAPAT code is still being used, we may save the
;    95          0625  1 !       overhead of calling $ERAPAT.  If the flag SGN$V_LOADERAPAT in the cell
;    96          0626  1 !       SGN$GL_LOADFLAGS is set, it indicates that an alternate $ERAPAT has been
;    97          0627  1 !       loaded, and that we should call the $ERAPAT routine instead of taking
;    98          0628  1 !       the shortcut.
;    99          0629  1 !
;   100          0630  1 !       Note that the flag SGN$V_LOADERAPAT corresponds to the SYSGEN parameter
;   101          0631  1 !       LOADERAPAT.  Since the SYSGEN parameter may be changed on the running
;   102          0632  1 !       system, it implies that until the system is rebooted, it is possible
;   103          0633  1 !       that the site-specific $ERAPAT will not be called, and the default
;   104          0634  1 !       erase pattern (0), will be used in its place.  We do not believe this
;   105          0635  1 !       to be a significant security risk.
;   106          0636  1 !
;   107          0637  1 !       Note that the default $ERAPAT code defines a one-step DSE procedure
;   108          0638  1 !       for disks, and the erase pattern is 0.
;   109          0639  1 !
;   110          0640  1 !       This routine assumes that I/O transfers of an arbitrary length can be
;   111          0641  1 !       done to any disk device with but a single QIO.
;   112          0642  1 !
;   113          0643  1 ! CALLING SEQUENCE:
;   114          0644  1 !       ERASE_BLOCKS (ARG1, ARG2, ARG3)
;   115          0645  1 !
;   116          0646  1 ! INPUT PARAMETERS:
;   117          0647  1 !       ARG1: LBN of first block to be erased
;   118          0648  1 !       ARG2: number of blocks to erase
;   119          0649  1 !       ARG3: I/O channel to the device
;   120          0650  1 !
;   121          0651  1 ! IMPLICIT INPUTS:
;   122          0652  1 !       CURRENT_VCB: VCB of volume
;   123          0653  1 !       CURRENT_UCB: UCB of volume
;   124          0654  1 !
;   125          0655  1 ! OUTPUT PARAMETERS:
;   126          0656  1 !       None.
;   127          0657  1 !
;   128          0658  1 ! IMPLICIT OUTPUTS:
;   129          0659  1 !       LOC_LBN: placement LBN of allocation or 0
;   130          0660  1 !
;   131          0661  1 ! ROUTINE VALUE:
;   132          0662  1 !       1 if successful erase
;   133          0663  1 !       <a system status code> if an error was encountered.
```

```
;  134     0664  1 !
;  135     0665  1 ! SIDE EFFECTS:
;  136     0666  1 !       None.
;  137     0667  1 !
;  138     0668  1 !--
;  139     0669  1
;  140     0670  2 BEGIN                                           ! Start of ERASE_BLOCKS
;  141     0671  2
;  142     0672  2 EXTERNAL
;  143     0673  2         SGN$GL_LOADFLAGS: BITVECTOR ADDRESSING_MODE (GENERAL);
;  144     0674  2                                                 ! System flags bitvector
;  145     0675  2
;  146     0676  2 EXTERNAL LITERAL
;  147     0677  2         SGN$V_LOADERAPAT;                       ! System flag
;  148     0678  2
;  149     0679  2 LOCAL
;  150     0680  2         ERASE_PASS      : LONG,                 ! Count of erase passes
;  151     0681  2         ERASE_PATTERN   : LONG,                 ! Pattern to write to the disk
;  152     0682  2         ERASE_STATUS    : LONG,                 ! Intermediate status
;  153     0683  2         STATUS          : LONG;                 ! Store status of operation
;  154     0684  2
;  155     0685  2 !
;  156     0686  2 ! Determine if default $ERAPAT routine is present.
;  157     0687  2 !
;  158     0688  2 IF NOT .SGN$GL_LOADFLAGS [SGN$V_LOADERAPAT]
;  159     0689  2 THEN
;  160     0690  3     BEGIN
;  161     0691  3     !
;  162     0692  3     ! The default $ERAPAT is present.  That implies that the
;  163     0693  3     ! DSE is a one-pass operation that zeroes each block.
;  164     0694  3     !
;  165     0695  3     STATUS = SS$_NOTRAN;
;  166     0696  3     ERASE_STATUS = DO_ERASE (.START_LBN, .BLOCK_COUNT, 0, .CHANNEL);
;  167     0697  3     END
;  168     0698  2 ELSE
;  169     0699  3     BEGIN
;  170     0700  3     !
;  171     0701  3     ! A site-specific $ERAPAT has been loaded.
;  172     0702  3     !
;  173     0703  3     ERASE_PASS = 1;
;  174     0704  3     STATUS = $ERAPAT (TYPE=ERA$K_DISK, COUNT=.ERASE_PASS, PATADR=ERASE_PATTERN);
;  175     0705  3     ERASE_STATUS = .STATUS;
;  176     0706  3     WHILE .STATUS AND (.STATUS NEQ SS$_NOTRAN) DO
;  177     0707  4         BEGIN
;  178     0708  4         !
;  179     0709  4         ! Write the erase pattern to the disk and call $ERAPAT
;  180     0710  4         ! to generate the next data security erase pattern.
;  181     0711  4         ! Each time we call $ERAPAT, increment the ERASE_PASS.
;  182     0712  4         ! When $ERAPAT returns SS$_NOTRAN, the DSE is complete.
;  183     0713  4         ! Note that if the last attempt to erase the data succeeds,
;  184     0714  4         ! we assume that the operation was a success.
;  185     0715  4         !
;  186     0716  4         ERASE_STATUS = DO_ERASE (.START_LBN, .BLOCK_COUNT, .ERASE_PATTERN, .CHANNEL);
;  187     0717  4         ERASE_PASS = .ERASE_PASS + 1;
;  188     0718  4         STATUS = $ERAPAT (TYPE=ERA$K_DISK, COUNT=.ERASE_PASS, PATADR=ERASE_PATTERN);
;  189     0719  3         END;
;  190     0720  2     END;
```

```
;  191          0721  2 !
;  192          0722  2 ! Return the most significant status value.  At this point, ERASE_STATUS
;  193          0723  2 ! is the status of the write to disk, and STATUS is the last of the last
;  194          0724  2 ! call to $ERAPAT.  If the write had an error, return that status, other
;  195          0725  2 ! wise return the status of the last call to $ERAPAT (which should be
;  196          0726  2 ! SS$_NOTRAN).
;  197          0727  2 !
;  198          0728  2 IF NOT .ERASE_STATUS
;  199          0729  2 THEN
;  200          0730  2     .ERASE_STATUS
;  201          0731  2 ELSE
;  202          0732  2     .STATUS
;  203          0733  2
;  204          0734  1 END;                                    ! End of ERASE_BLOCKS

                                                                .TITLE  ERASE
                                                                .IDENT  \V04-000\

                                                                .EXTRN  SGN$GL_LOADFLAGS
                                                                .EXTRN  SGN$V_LOADERAPAT
                                                                .EXTRN  SYS$ERAPAT

                                                                .PSECT  $CODE$,NOWRT,2

                                                 003C 00000     .ENTRY  ERASE_BLOCKS, Save R2,R3,R4,R5          ; 0607
                                    55 00000000G   00   9E 00002 MOVAB   SYS$ERAPAT, R5
                                    5E             04   C2 00009 SUBL2   #4, SP
                          18 00000000G   00 00000000G   8F   E0 0000C BBS     #SGN$V_LOADERAPAT, SGN$GL_LOADFLAGS, 1$ ; 0688
                                    54       0629       8F   3C 00018 MOVZWL  #1577, STATUS                  ; 0695
                                             0C         AC   DD 0001D PUSHL   CHANNEL                         ; 0696
                                                        7E   D4 00020 CLRL    -(SP)
                                       7E       04      AC   7D 00022 MOVQ    START_LBN, -(SP)
                                0000V  CF               04   FB 00026 CALLS   #4, DO_ERASE
                                    53                  50   D0 0002B MOVL    R0, ERASE_STATUS
                                                        40   11 0002E BRB     3$                               ; 0688
                                    52                  01   D0 00030 1$:    MOVL    #1, ERASE_PASS           ; 0703
                                          4004          8F   BB 00033       PUSHR   #^M<R2,SP>               ; 0704
                                                        02   DD 00037       PUSHL   #2
                                    65                  03   FB 00039       CALLS   #3, SYS$ERAPAT
                                    54                  50   D0 0003C       MOVL    R0, STATUS
                                    53                  54   D0 0003F       MOVL    STATUS, ERASE_STATUS        ; 0705
                                    2B                  54   E9 00042 2$:    BLBC    STATUS, 3$                  ; 0706
                             00000629   8F              54   D1 00045       CMPL    STATUS, #1577
                                                        22   13 0004C       BEQL    3$
                                             0C         AC   DD 0004E       PUSHL   CHANNEL                     ; 0716
                                             04         AE   DD 00051       PUSHL   ERASE_PATTERN
                                       7E       04      AC   7D 00054       MOVQ    START_LBN, -(SP)
                                0000V  CF               04   FB 00058       CALLS   #4, DO_ERASE
                                    53                  50   D0 0005D       MOVL    R0, ERASE_STATUS
                                                        52   D6 00060       INCL    ERASE_PASS              ; 0717
                                          4004          8F   BB 00062       PUSHR   #^M<R2,SP>              ; 0718
                                                        02   DD 00066       PUSHL   #2
                                    65                  03   FB 00068       CALLS   #3, SYS$ERAPAT
                                    54                  50   D0 0006B       MOVL    R0, STATUS
                                                        D2   11 0006E       BRB     2$                      ; 0706
                                    04                  53   E8 00070 3$:    BLBS    ERASE_STATUS, 4$        ; 0728
```

```
                50              53  D0 00073          MOVL    ERASE_STATUS, R0                        ; 0730
                                    04 00076          RET                                             ;
                50              54  D0 00077 4$:      MOVL    STATUS, R0                              ; 0732
                                    04 0007A          RET                                             ; 0734

; Routine Size:  123 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  206     0735  1 ROUTINE DO_ERASE (START_LBN, BLOCK_COUNT, ERASE_PATTERN, CHANNEL) =
;  207     0736  1
;  208     0737  1 !++
;  209     0738  1 !
;  210     0739  1 ! FUNCTIONAL DESCRIPTION:
;  211     0740  1 !
;  212     0741  1 !       Helper routine to ERASE_BLOCKS.  Write the erase pattern to the disk,
;  213     0742  1 !       making sure every block gets written.  If a bad block is encountered,
;  214     0743  1 !       write around it.
;  215     0744  1 !
;  216     0745  1 !       This routine assumes that I/O transfers of an arbitrary length can be
;  217     0746  1 !       done to any disk device with but a single QIO.
;  218     0747  1 !
;  219     0748  1 !       Even though the nature of this routine lends itself to a recursive
;  220     0749  1 !       implementation, it was done iteratively to minimize stack usage.
;  221     0750  1 !
;  222     0751  1 ! CALLING SEQUENCE:
;  223     0752  1 !       DO_ERASE (ARG1, ARG2, ARG3, ARG4)
;  224     0753  1 !
;  225     0754  1 ! INPUT PARAMETERS:
;  226     0755  1 !       ARG1: LBN of first block to be erased
;  227     0756  1 !       ARG2: number of blocks to erase
;  228     0757  1 !       ARG3: 4 byte erase pattern
;  229     0758  1 !       ARG4: I/O channel to the device
;  230     0759  1 !
;  231     0760  1 ! IMPLICIT INPUTS:
;  232     0761  1 !       None.
;  233     0762  1 !
;  234     0763  1 ! OUTPUT PARAMETERS:
;  235     0764  1 !       None.
;  236     0765  1 !
;  237     0766  1 ! IMPLICIT OUTPUTS:
;  238     0767  1 !       None.
;  239     0768  1 !
;  240     0769  1 ! ROUTINE VALUE:
;  241     0770  1 !       1 if successful erase
;  242     0771  1 !       <a system status code> if an error was encountered.
;  243     0772  1 !
;  244     0773  1 ! SIDE EFFECTS:
;  245     0774  1 !       The count of erase I/O operations is incremented.
;  246     0775  1 !
;  247     0776  1 !--                                                                    ; F
;  248     0777  1
;  249     0778  2 BEGIN                                           ! Start of DO_ERASE
;  250     0779  2
;  251     0780  2 EXTERNAL
;  252     0781  2         SGN$GL_VMSD2    : LONG ADDRESSING_MODE (GENERAL), !*** TEMPORARY
;  253     0782  2         PMS$GL_ERASEIO  : LONG ADDRESSING_MODE (GENERAL);
;  254     0783  2                                                 ! Count of erase I/O operations
;  255     0784  2
;  256     0785  2 LOCAL
;  257     0786  2         LBN             : LONG,                 ! local copy of START_LBN
;  258     0787  2         COUNT           : LONG,                 ! local copy of BLOCK_COUNT
;  259     0788  2         BLOCKS_ERASED   : LONG,                 ! # of blocks actually erased
;  260     0789  2         IOSTS           : BBLOCK [8],           ! I/O status block
;  261     0790  2         ERASE_STATUS    : LONG,                 ! Routine status
;  262     0791  2         STATUS          : LONG;                 ! Store status of operation
```

```
;  263          0792  2
;  264          0793  2 !
;  265          0794  2 ! Erase the specified portion of the disk.  If the erase fails, retry
;  266          0795  2 ! the operation starting from 1 block past the point of failure.  Only
;  267          0796  2 ! the status coded from the first such error is returned.
;  268          0797  2 !
;  269          0798  2 !
;  270          0799  2 ! Set things up for the loop.  The loop will terminate when no more
;  271          0800  2 ! blocks need to be erased.
;  272          0801  2 !
;  273          0802  2 ERASE_STATUS = SS$_NORMAL;                       ! Assume no errors
;  274          0803  2 STATUS = SS$_NORMAL;                             ! Ditto
;  275          0804  2 LBN = .START_LBN;                                ! Copy starting LBN
;  276          0805  2 COUNT = .BLOCK_COUNT;                           ! Copy # of blocks to erase
;  277          0806  2 WHILE (.COUNT GTR 0) DO
;  278          0807  3     BEGIN                                       ! Start of erase loop
;  279          0808  3     !
;  280          0809  3     ! Issue an erase $QIOW to the volume.  The IO$M_ERASE modifier turns
;  281          0810  3     ! an ordinary write into a special low-overhead write.
;  282          0811  3     !
;  283        P 0812  3     STATUS = $QIOW      (CHAN = .CHANNEL,
;  284        P 0813  3                          FUNC = (IO$_WRITELBLK OR IO$M_ERASE),
;  285        P 0814  3                          EFN  = MOUNT_EFN,
;  286        P 0815  3                          IOSB = IOSTS,
;  287        P 0816  3                          P1   = ERASE_PATTERN,
;  288        P 0817  3                          P2   = (.COUNT * 512),
;  289        P 0818  3                          P3   = .LBN
;  290          0819  3                         );
;  291          0820  3
;  292          0821  3     PMS$GL_ERASEIO = .PMS$GL_ERASEIO + 1;       ! Bump erase I/O counter
;  293          0822  3
;  294          0823  3     IF .STATUS
;  295          0824  3     THEN
;  296          0825  4         BEGIN
;  297          0826  4         !
;  298          0827  4         ! The call to $QIOW succeeded.
;  299          0828  4         ! Decrement the count of blocks to erase by the number of blocks
;  300          0829  4         ! actually erased.  Advance the starting lbn to 1 block past the
;  301          0830  4         ! last block actually erased.
;  302          0831  4         !
;  303          0832  4         BLOCKS_ERASED = .IOSTS [2,0,32,0] / 512;
;  304          0833  4         COUNT = (.COUNT - .BLOCKS_ERASED) - 1;
;  305          0834  4         LBN = (.LBN + .BLOCKS_ERASED) + 1;
;  306          0835  4         !
;  307          0836  4         ! If an error was encountered during the erase, and it is the first
;  308          0837  4         ! such error, save the error status code so that if may be returned
;  309          0838  4         ! to the caller.
;  310          0839  4         !
;  311          0840  5         IF  (NOT (STATUS = .IOSTS [0,0,16,0]))
;  312          0841  4         AND .ERASE_STATUS
;  313          0842  4         THEN
;  314          0843  4             ERASE_STATUS = .STATUS;             ! Save first error status
;  315          0844  4         END
;  316          0845  3     ELSE
;  317          0846  4         BEGIN
;  318          0847  4         !
;  319          0848  4         ! The erase $QIOW failed outright.  Terminate the erase operation.
```

```
;  320     0849  4          !
;  321     0850  4          ERASE_STATUS = .STATUS;                 ! Save exit status
;  322     0851  4          COUNT = 0;                              ! Insure loop termination
;  323     0852  3          END;
;  324     0853  2      END;                                        ! End of erase loop
;  325     0854  2
;  326     0855  2 !
;  327     0856  2 ! Return the erase status.  Only the first error encountered is reported.
;  328     0857  2 !
;  329     0858  2 RETURN .ERASE_STATUS
;  330     0859  2
;  331     0860  1 END;                                            ! End of DO_ERASE


                                                .EXTRN  SGN$GL_VMSD2, PMS$GL_ERASEIO
                                                .EXTRN  SYS$QIOW

                                    003C 00000 DO_ERASE:
                                                .WORD   Save R2,R3,R4,R5                ; 0735
                                5E          08  C2 00002          SUBL2   #8, SP
                                55          01  D0 00005          MOVL    #1, ERASE_STATUS            ; 0802
                                50          01  D0 00008          MOVL    #1, STATUS                  ; 0803
                                53      04  AC  D0 0000B          MOVL    START_LBN, LBN              ; 0804
                                52      08  AC  D0 0000F          MOVL    BLOCK_COUNT, COUNT          ; 0805
                                            52  D5 00013 1$:      TSTL    COUNT                       ; 0806
                                            57  15 00015          BLEQ    3$
                                            7E  7C 00017          CLRQ    -(SP)                       ; 0819
                                            7E  D4 00019          CLRL    -(SP)
                                            53  DD 0001B          PUSHL   LBN
                                7E          52  09 78 0001D          ASHL    #9, COUNT, -(SP)
                                        0C  AC  9F 00021          PUSHAB  ERASE_PATTERN
                                            7E  7C 00024          CLRQ    -(SP)
                                        20  AE  9F 00026          PUSHAB  IOSTS
                                    7E      0420  8F 3C 00029          MOVZWL  #1056, -(SP)
                                            10  AC DD 0002E          PUSHL   CHANNEL
                                            1A  DD 00031          PUSHL   #26
                                00000000G   00  0C FB 00033          CALLS   #12, SYS$QIOW
                                        00000000G   00  D6 0003A          INCL    PMS$GL_ERASEIO      ; 0821
                                    24          50  E9 00040          BLBC    STATUS, 2$              ; 0823
                            54          02  AE 00000200  8F C7 00043          DIVL3   #512, IOSTS+2, BLOCKS_ERASED    ; 0832
                            51              52          54  C3 0004C          SUBL3   BLOCKS_ERASED, COUNT, R1        ; 0833
                                            52      FF  A1  9E 00050          MOVAB   -1(R1), COUNT
                                            53      01 A443 9E 00054          MOVAB   1(BLOCKS_ERASED)[LBN], LBN      ; 0834
                                            50          6E  3C 00059          MOVZWL  IOSTS, STATUS       ; 0840
                                            B4          50  E8 0005C          BLBS    STATUS, 1$
                                            B1          55  E9 0005F          BLBC    ERASE_STATUS, 1$    ; 0841
                                            55          50  D0 00062          MOVL    STATUS, ERASE_STATUS    ; 0843
                                                        AC  11 00065          BRB     1$                  ; 0823
                                            55          50  D0 00067 2$:      MOVL    STATUS, ERASE_STATUS    ; 0850
                                                        52  D4 0006A          CLRL    COUNT               ; 0851
                                                        A5  11 0006C          BRB     1$                  ; 0806
                                            50          55  D0 0006E 3$:      MOVL    ERASE_STATUS, R0    ; 0858
                                                            04 00071          RET                         ; 0860

; Routine Size:  114 bytes,    Routine Base:  $CODE$ + 007B
```

```
;  332          0861  1
;  333          0862  1 END
;  334          0863  0 ELUDOM


;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $CODE$                          237  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        17         0      1000          00:01.8



;                                   COMMAND QUALIFIERS

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

; Size:          237 code + 0 data bytes
; Run Time:          00:13.2
; Elapsed Time:      00:26.6
; Lines/CPU Min:      3931
; Lexemes/CPU-Min:  31548
; Memory Used:   102 pages
; Compilation Complete
```

CHKHM2
LIS

CHNUCB
LIS

GETUIC
LIS

ERASE
LIS

LEFTONE
LIS

MOUDK2
LIS

CHKHM1
LIS

CHKSM2
LIS

CLUSTRMNT
LIS

INIFC2
LIS

MOUDK1
LIS

MAKRVT
LIS

MAKLOG
LIS