H 5

```
FFFFFFFFFFFFFFF        111          111    XXX         XXX
FFFFFFFFFFFFFFF        111          111    XXX         XXX
FFFFFFFFFFFFFFF        111          111    XXX         XXX
FFF                 111111       111111    XXX         XXX
FFF                 111111       111111    XXX         XXX
FFF                 111111       111111    XXX         XXX
FFF                    111          111       XXX   XXX
FFF                    111          111       XXX   XXX
FFF                    111          111       XXX   XXX
FFFFFFFF.FFF           111          111          XXX
FFFFFFFFFFFF           111          111          XXX
FFFFFFFFFFFF           111          111          XXX
FFF                    111          111       XXX   XXX
FFF                    111          111       XXX   XXX
FFF                    111          111       XXX   XXX
FFF                    111          111    XXX         XXX
FFF                    111          111    XXX         XXX
FFF                    111          111    XXX         XXX
FFF                 111111111    111111111 XXX         XXX
FFF                 111111111    111111111 XXX         XXX
FFF                 111111111    111111111 XXX         XXX
```

```
_$25
Symb
----
IOC1
IO_C
IO_C
IO_D
IO_F
IO_S
KILL

KILL
KILL
LB_E
LB_D
LB_F
LB_H
LB_L
LOCA
LOCA
LOCK

LOCK
LOCK
LOCK
LOC_
LOC_
L_CO
L_CO
L_DA
L_DA
MAIN
MAKE
MAKE
MAKE
MAKE
MAKE
MAKE

MAKE
MAKE
MAP_
MAP_

MAP_
MARK
MARK
MARK
MARK
MARK
```

FILEID**ERASE

1 2

EXT

```
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
```

```
;     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:  F11ACP Structure Level 2
;    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 !       V03-007 CDS0004         Christian D. Saether    20-Jun-1984
;    53          0053  1 !               Raise the process diocnt around the call to $qio so
;    54          0054  1 !               it will not fail for lack of quota.  Also raise ASTCNT
;    55          0055  1 !               so it will not fail for that reason.
;    56          0056  1 !
;    57          0057  1 !       V03-006 ACG0408         Andrew C. Goldstein,    23-Mar-1984  14:44
```

```
;    58          0058  1 !                 Add AST parameter so that impure storage is fully based
;    59          0059  1 !
;    60          0060  1 !       V03-005 CDS0003         Christian D. Saether    30-Dec-1983
;    61          0061  1 !                 Use L_NORM linkage and BIND_COMMON macro.
;    62          0062  1 !
;    63          0063  1 !       V03-004 CDS0002         Christian D. Saether    27-Sep-1983
;    64          0064  1 !                 Enhance privileges for erase.
;    65          0065  1 !
;    66          0066  1 !       V03-003 STJ3104         Steven T. Jeffreys,     03-Jun-1983
;    67          0067  1 !                 - Removed reference to VMSD2.
;    68          0068  1 !
;    69          0069  1 !       V03-002 CDS0001         Christian D. Saether    12-May-1983
;    70          0070  1 !                 Make erase qio asychronous.
;    71          0071  1 !
;    72          0072  1 !       V03-001 STJ3082         Steven T. Jeffreys,     30-Mar-1983
;    73          0073  1 !                 - Added CHANNEL parameter to ERASE_BLOCKS and DO_ERASE.
;    74          0074  1 !                   This makes for a cleaner interface with callers outside
;    75          0075  1 !                   of the Files-11 ACP.  (eg. REBUILD and ANALYZE/DISK)
;    76          0076  1 !                 - Use VMSD2 to enable/disable erase. (Temporary)
;    77          0077  1 !**
;    78          0078  1
;    79          0079  1
;    80          0080  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    81          0081  1 REQUIRE 'SRC$:FCPDEF.B32';
;    82          1072  1
;    83          1073  1 !
;    84          1074  1 ! Table of contents.
;    85          1075  1 !
;    86          1076  1
;    87          1077  1 FORWARD ROUTINE
;    88          1078  1         ERASE_BLOCKS    : L_NORM,               ! Top level DSE routine
;    89          1079  1         DO_ERASE        : L_NORM;               ! Issues the erase $QIO
```

```
;    91          1080  1 GLOBAL ROUTINE ERASE_BLOCKS (START_LBN, BLOCK_COUNT, CHANNEL) : L_NORM =
;    92          1081  1
;    93          1082  1 !++
;    94          1083  1 !
;    95          1084  1 ! FUNCTIONAL DESCRIPTION:
;    96          1085  1 !
;    97          1086  1 !       Perform a data security erase (DSE) on a single contiguous extent
;    98          1087  1 !       on the disk.  This routine is recursive.
;    99          1088  1 !
;   100          1089  1 !
;   101          1090  1 !       The DSE is done by calling the erase pattern generator system service,
;   102          1091  1 !       $ERAPAT, and writing the pattern returned to the disk.  This is repeated
;   103          1092  1 !       until $ERAPAT returns a status of SS$_NOTRAN, which indicates that the
;   104          1093  1 !       DSE is complete.
;   105          1094  1 !
;   106          1095  1 !       The $ERAPAT code is loadable, and may vary from site to site.
;   107          1096  1 !       However, the default $ERAPAT code is very simple, and by checking to see
;   108          1097  1 !       if the default $ERAPAT code is still being used, we may save the
;   109          1098  1 !       overhead of calling $ERAPAT.  If the flag SGN$V_LOADERAPAT in the cell
;   110          1099  1 !       SGN$GL_LOADFLAGS is set, it indicates that an alternate $ERAPAT has been
;   111          1100  1 !       loaded, and that we should call the $ERAPAT routine instead of taking
;   112          1101  1 !       the shortcut.
;   113          1102  1 !
;   114          1103  1 !       Note that the flag SGN$V_LOADERAPAT corresponds to the SYSGEN parameter
;   115          1104  1 !       LOADERAPAT.  Since the SYSGEN parameter may be changed on the running
;   116          1105  1 !       system, it implies that until the system is rebooted, it is possible
;   117          1106  1 !       that the site-specific $ERAPAT will not be called, and the default
;   118          1107  1 !       erase pattern (0), will be used in its place.  We do not believe this
;   119          1108  1 !       to be a significant security risk.
;   120          1109  1 !
;   121          1110  1 !       Note that the default $ERAPAT code defines a one-step DSE procedure
;   122          1111  1 !       for disks, and the erase pattern is 0.
;   123          1112  1 !
;   124          1113  1 !       This routine assumes that I/O transfers of an arbitrary length can be
;   125          1114  1 !       done to any disk device with but a single QIO.
;   126          1115  1 !
;   127          1116  1 ! CALLING SEQUENCE:
;   128          1117  1 !       ERASE_BLOCKS (ARG1, ARG2, ARG3)
;   129          1118  1 !
;   130          1119  1 ! INPUT PARAMETERS:
;   131          1120  1 !       ARG1: LBN of first block to be erased
;   132          1121  1 !       ARG2: number of blocks to erase
;   133          1122  1 !       ARG3: I/O channel to the device
;   134          1123  1 !
;   135          1124  1 ! IMPLICIT INPUTS:
;   136          1125  1 !       CURRENT_VCB: VCB of volume
;   137          1126  1 !       CURRENT_UCB: UCB of volume
;   138          1127  1 !
;   139          1128  1 ! OUTPUT PARAMETERS:
;   140          1129  1 !       None.
;   141          1130  1 !
;   142          1131  1 ! IMPLICIT OUTPUTS:
;   143          1132  1 !       LOC_LBN: plcement LBN of allocation or 0
;   144          1133  1 !
;   145          1134  1 ! ROUTINE VALUE:
;   146          1135  1 !       1 if successful erase
;   147          1136  1 !       <a system status code> if an error was encountered.
```

```
;  148          1137  1 !
;  149          1138  1 ! SIDE EFFECTS:
;  150          1139  1 !       None.
;  151          1140  1 !
;  152          1141  1 !--
;  153          1142  1
;  154          1143  2 BEGIN                                           ! Start of ERASE_BLOCKS
;  155          1144  2
;  156          1145  2 EXTERNAL
;  157          1146  2         SGN$GL_LOADFLAGS: BITVECTOR ADDRESSING_MODE (GENERAL);
;  158          1147  2                                                 ! System flags bitvector
;  159          1148  2
;  160          1149  2 EXTERNAL LITERAL
;  161          1150  2         SGN$V_LOADERAPAT;                       ! System flag
;  162          1151  2
;  163          1152  2 BASE_REGISTER;
;  164          1153  2
;  165          1154  2 LOCAL
;  166          1155  2         ERASE_PASS      : LONG,                 ! Count of erase passes
;  167          1156  2         ERASE_PATTERN   : LONG,                 ! Pattern to write to the disk
;  168          1157  2         ERASE_STATUS    : LONG,                 ! Intermediate status
;  169          1158  2         STATUS          : LONG;                 ! Store status of operation
;  170          1159  2
;  171          1160  2 !
;  172          1161  2 ! Determine if default $ERAPAT routine is present.
;  173          1162  2 !
;  174          1163  2 IF NOT .SGN$GL_LOADFLAGS [SGN$V_LOADERAPAT]
;  175          1164  2 THEN
;  176          1165  3     BEGIN
;  177          1166  3     !
;  178          1167  3     ! The default $ERAPAT is present.  That implies that the
;  179          1168  3     ! DSE is a one-pass operation that zeroes each block.
;  180          1169  3     !
;  181          1170  3     STATUS = SS$_NOTRAN;
;  182          1171  3     ERASE_STATUS = DO_ERASE (.START_LBN, .BLOCK_COUNT, 0, .CHANNEL);
;  183          1172  3     END
;  184          1173  2 ELSE
;  185          1174  3     BEGIN
;  186          1175  3     !
;  187          1176  3     ! A site-specific $ERAPAT has been loaded.
;  188          1177  3     !
;  189          1178  3     ERASE_PASS = 1;
;  190          1179  3     STATUS = $ERAPAT (TYPE=ERA$K_DISK, COUNT=.ERASE_PASS, PATADR=ERASE_PATTERN);
;  191          1180  3     ERASE_STATUS = .STATUS;
;  192          1181  3     WHILE .STATUS AND (.STATUS NEQ SS$_NOTRAN) DO
;  193          1182  4         BEGIN
;  194          1183  4         !
;  195          1184  4         ! Write the erase pattern to the disk and call $ERAPAT
;  196          1185  4         ! to generate the next data security erase pattern.
;  197          1186  4         ! Each time we call $ERAPAT, increment the ERASE_PASS.
;  198          1187  4         ! When $ERAPAT returns SS$_NOTRAN, the DSE is complete.
;  199          1188  4         ! Note that if the last attempt to erase the data succeeds,
;  200          1189  4         ! we assume that the operation was a success.
;  201          1190  4         !
;  202          1191  4         ERASE_STATUS = DO_ERASE (.START_LBN, .BLOCK_COUNT, .ERASE_PATTERN, .CHANNEL);
;  203          1192  4         ERASE_PASS = .ERASE_PASS + 1;
;  204          1193  4         STATUS = $ERAPAT (TYPE=ERA$K_DISK, COUNT=.ERASE_PASS, PATADR=ERASE_PATTERN);
```

```
;  205          1194  3          END;
;  206          1195  2      END;
;  207          1196  2 !
;  208          1197  2 ! Return the most significant status value.  At this point, ERASE_STATUS
;  209          1198  2 ! is the status of the write to disk, and STATUS is the last of the last
;  210          1199  2 ! call to $ERAPAT.  If the write had an error, return that status, other
;  211          1200  2 ! wise return the status of the last call to $ERAPAT (which should be
;  212          1201  2 ! SS$_NOTRAN).
;  213          1202  2 !
;  214          1203  2 IF NOT .ERASE_STATUS
;  215          1204  2 THEN
;  216          1205  2      .ERASE_STATUS
;  217          1206  2 ELSE
;  218          1207  2      .STATUS
;  219          1208  2
;  220          1209  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          ; 1080
                                        55 00000000G  00  9E 00002    MOVAB   SYS$ERAPAT, R5
                                                  5E  04  C2 00009    SUBL2   #4, SP
                                18 00000000G  54  00 00000000G  8F  E0 0000C    BBS     #SGN$V_LOADERAPAT, SGN$GL_LOADFLAGS, 1$  ; 1163
                                                  54      0629  8F  3C 00018    MOVZWL  #1577, STATUS                   ; 1170
                                                          0C  AC  DD 0001D    PUSHL   CHANNEL                         ; 1171
                                                              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$                      ; 1163
                                                  52              01  D0 00030 1$: MOVL    #1, ERASE_PASS          ; 1178
                                                          4004  8F  BB 00033    PUSHR   #^M<R2,SP>              ; 1179
                                                                02  DD 00037    PUSHL   #2
                                                  65            03  FB 00039    CALLS   #3, SYS$ERAPAT
                                                  54            50  D0 0003C    MOVL    R0, STATUS
                                                  53            54  D0 0003F    MOVL    STATUS, ERASE_STATUS    ; 1180
                                                  2B            54  E9 00042 2$: BLBC    STATUS, 3$              ; 1181
                                        00000629  8F            54  D1 00045    CMPL    STATUS, #1577
                                                                22  13 0004C    BEQL    3$
                                                          0C  AC  DD 0004E    PUSHL   CHANNEL                 ; 1191
                                                          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          ; 1192
                                                          4004  8F  BB 00062    PUSHR   #^M<R2,SP>          ; 1193
                                                                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$                              ; 1181
04                  53  E8 00070 3$:         BLBS    ERASE_STATUS, 4$                ; 1203
50                  53  D0 00073             MOVL    ERASE_STATUS, R0                ; 1205
                        04 00076             RET                                     ;
50                  54  D0 00077 4$:         MOVL    STATUS, R0                      ; 1207
                        04 0007A             RET                                     ; 1209

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

```
;  222          1210  1 ROUTINE DO_ERASE (START_LBN, BLOCK_COUNT, ERASE_PATTERN, CHANNEL) : L_NORM =
;  223          1211  1
;  224          1212  1 !++
;  225          1213  1 !
;  226          1214  1 ! FUNCTIONAL DESCRIPTION:
;  227          1215  1 !
;  228          1216  1 !       Helper routine to ERASE_BLOCKS.  Write the erase pattern to the disk,
;  229          1217  1 !       making sure every block gets written.  If a bad block is encountered,
;  230          1218  1 !       write around it.
;  231          1219  1 !
;  232          1220  1 !       This routine assumes that I/O transfers of an arbitrary length can be
;  233          1221  1 !       done to any disk device with but a single QIO.
;  234          1222  1 !
;  235          1223  1 !       Even though the nature of this routine lends itself to a recursive
;  236          1224  1 !       implementation, it was done iteratively to minimize stack usage.
;  237          1225  1 !
;  238          1226  1 ! CALLING SEQUENCE:
;  239          1227  1 !       DO_ERASE (ARG1, ARG2, ARG3, ARG4)
;  240          1228  1 !
;  241          1229  1 ! INPUT PARAMETERS:
;  242          1230  1 !       ARG1: LBN of first block to be erased
;  243          1231  1 !       ARG2: number of blocks to erase                                          ; R
;  244          1232  1 !       ARG3: 4 byte erase pattern
;  245          1233  1 !       ARG4: I/O channel to the device
;  246          1234  1 !
;  247          1235  1 ! IMPLICIT INPUTS:
;  248          1236  1 !       None.
;  249          1237  1 !
;  250          1238  1 ! OUTPUT PARAMETERS:
;  251          1239  1 !       None.
;  252          1240  1 !
;  253          1241  1 ! IMPLICIT OUTPUTS:
;  254          1242  1 !       None.
;  255          1243  1 !
;  256          1244  1 ! ROUTINE VALUE:
;  257          1245  1 !       1 if successful erase
;  258          1246  1 !       <a system status code> if an error was encountered.
;  259          1247  1 !
;  260          1248  1 ! SIDE EFFECTS:
;  261          1249  1 !       The count of erase I/O operations is incremented.
;  262          1250  1 !
;  263          1251  1 !--
;  264          1252  1
;  265          1253  2 BEGIN                                           ! Start of DO_ERASE
;  266          1254  2
;  267          1255  2 EXTERNAL
;  268          1256  2         CTL$GL_PCB      : ADDRESSING_MODE (GENERAL),
;  269          1257  2         CTL$GL_PHD      : ADDRESSING_MODE (GENERAL),
;  270          1258  2         PMS$GL_ERASEIO  : LONG ADDRESSING_MODE (GENERAL);
;  271          1259  2
;  272          1260  2                                               ! Count of erase I/O operations
;  273          1261  2
;  274          1262  2 BASE_REGISTER;
;  275          1263  2
;  276          1264  2 EXTERNAL ROUTINE
;  277          1265  2         WAIT_FOR_AST    : L_NORM,
;  278          1266  2         CONTINUE_THREAD : L_NORM;
```

```
;  279          1267  2
;  280          1268  2 LOCAL
;  281          1269  2         LBN             : LONG,                 ! local copy of START_LBN
;  282          1270  2         COUNT           : LONG,                 ! local copy of BLOCK_COUNT
;  283          1271  2         BLOCKS_ERASED   : LONG,                 ! # of blocks actually erased
;  284          1272  2         IOSTS           : BBLOCK [8],           ! I/O status block
;  285          1273  2         ERASE_STATUS    : LONG,                 ! Routine status
;  286          1274  2         STATUS          : LONG;                 ! Store status of operation
;  287          1275  2
;  288          1276  2 !
;  289          1277  2 ! Erase the specified portion of the disk.  If the erase fails, retry
;  290          1278  2 ! the operation starting from 1 block past the point of failure.  Only
;  291          1279  2 ! the status coded from the first such error is returned.
;  292          1280  2 !
;  293          1281  2 !
;  294          1282  2 ! Set things up for the loop.  The loop will terminate when no more
;  295          1283  2 ! blocks need to be erased.
;  296          1284  2 !
;  297          1285  2 ERASE_STATUS = SS$_NORMAL;                        ! Assume no errors
;  298          1286  2 STATUS = SS$_NORMAL;                             ! Ditto
;  299          1287  2 LBN = .START_LBN;                                ! Copy starting LBN
;  300          1288  2 COUNT = .BLOCK_COUNT;                            ! Copy # of blocks to erase
;  301          1289  2 WHILE (.COUNT GTR 0) DO
;  302          1290  3     BEGIN                                        ! Start of erase loop
;  303          1291  3
;  304          1292  3     LOCAL
;  305          1293  3         PTR     : REF BBLOCK,
;  306          1294  3         SAVE_PRIV : VECTOR [4];
;  307          1295  3
;  308          1296  3     PTR = .CTL$GL_PCB;
;  309          1297  3     PTR [PCB$W_DIOCNT] = .PTR [PCB$W_DIOCNT] + 1;
;  310          1298  3     PTR [PCB$W_ASTCNT] = .PTR [PCB$W_ASTCNT] + 1;
;  311          1299  3     SAVE_PRIV [0] = .(PTR [PCB$Q_PRIV]);
;  312          1300  3     SAVE_PRIV [1] = .(PTR [PCB$Q_PRIV]+4);
;  313          1301  3     BBLOCK [PTR [PCB$Q_PRIV], PRV$V_LOG_IO] = 1;
;  314          1302  3     BBLOCK [PTR [PCB$Q_PRIV], PRV$V_BYPASS] = 1;
;  315          1303  3     PTR = .CTL$GL_PHD;
;  316          1304  3     SAVE_PRIV [2] = .(PTR [PHD$Q_PRIVMSK]);
;  317          1305  3     SAVE_PRIV [3] = .(PTR [PHD$Q_PRIVMSK]+4);
;  318          1306  3     BBLOCK [PTR [PHD$Q_PRIVMSK], PRV$V_LOG_IO] = 1;
;  319          1307  3     BBLOCK [PTR [PHD$Q_PRIVMSK], PRV$V_BYPASS] = 1;
;  320          1308  3
;  321          1309  3     !
;  322          1310  3     ! Issue an erase $QIO to the volume.  The IO$M_ERASE modifier turns
;  323          1311  3     ! an ordinary write into a special low-overhead write.
;  324          1312  3     !
;  325        P 1313  3     STATUS = $QIO       (CHAN = .CHANNEL,
;  326        P 1314  3                          FUNC = (IO$_WRITELBLK OR IO$M_ERASE),
;  327        P 1315  3                          IOSB = IOSTS,
;  328        P 1316  3                          ASTADR = CONTINUE_THREAD,
;  329        P 1317  3                          ASTPRM = .BASE,
;  330        P 1318  3                          P1   = ERASE_PATTERN,
;  331        P 1319  3                          P2   = (.COUNT * 512),
;  332        P 1320  3                          P3   = .LBN
;  333          1321  3                         );
;  334          1322  3
;  335          1323  3     PMS$GL_ERASEIO = .PMS$GL_ERASEIO + 1;       ! Bump erase I/O counter
```

```
;  336          1324  3
;  337          1325  3      (PTR [PHD$Q_PRIVMSK]) = .SAVE_PRIV [2];
;  338          1326  3      (PTR [PHD$Q_PRIVMSK]+4) = .SAVE_PRIV [3];
;  339          1327  3      PTR = .CTL$GL_PCB;
;  340          1328  3      PTR [PCB$W_ASTCNT] = .PTR [PCB$W_ASTCNT] - 1;
;  341          1329  3      PTR [PCB$W_DIOCNT] = .PTR [PCB$W_DIOCNT] - 1;
;  342          1330  3      (PTR [PCB$Q_PRIV]) = .SAVE_PRIV [0];
;  343          1331  3      (PTR [PCB$Q_PRIV]+4) = .SAVE_PRIV [1];
;  344          1332  3
;  345          1333  3      IF .STATUS
;  346          1334  3      THEN
;  347          1335  4          BEGIN
;  348          1336  4          WAIT_FOR_AST();
;  349          1337  4          STATUS = .IOSTS;
;  350          1338  3          END;
;  351          1339  3
;  352          1340  3      IF .STATUS
;  353          1341  3      THEN
;  354          1342  4          BEGIN
;  355          1343  4          !
;  356          1344  4          ! The call to $QIO succeeded.
;  357          1345  4          ! Decrement the count of blocks to erase by the number of blocks
;  358          1346  4          ! actually erased.  Advance the starting lbn to 1 block past the
;  359          1347  4          ! last block actually erased.
;  360          1348  4          !
;  361          1349  4          BLOCKS_ERASED = .IOSTS [2,0,32,0] / 512;
;  362          1350  4          COUNT = (.COUNT - .BLOCKS_ERASED) - 1;
;  363          1351  4          LBN = (.LBN + .BLOCKS_ERASED) + 1;
;  364          1352  4          !
;  365          1353  4          ! If an error was encountered during the erase, and it is the first
;  366          1354  4          ! such error, save the error status code so that if may be returned
;  367          1355  4          ! to the caller.
;  368          1356  4          !
;  369          1357  5          IF  (NOT (STATUS = .IOSTS [0,0,16,0]))
;  370          1358  4          AND .ERASE_STATUS
;  371          1359  4          THEN
;  372          1360  4              ERASE_STATUS = .STATUS;             ! Save first error status
;  373          1361  4          END
;  374          1362  3      ELSE
;  375          1363  4          BEGIN
;  376          1364  4          !
;  377          1365  4          ! The erase $QIO failed outright.  Terminate the erase operation.
;  378          1366  4          !
;  379          1367  4          ERASE_STATUS = .STATUS;                 ! Save exit status
;  380          1368  4          COUNT = 0;                              ! Insure loop termination
;  381          1369  3          END;
;  382          1370  2      END;                                        ! End of erase loop
;  383          1371  2
;  384          1372  2 !
;  385          1373  2 ! Return the erase status.  Only the first error encountered is reported.
;  386          1374  2 !
;  387          1375  2 RETURN .ERASE_STATUS
;  388          1376  2
;  389          1377  1 END;                                            ! End of DO_ERASE
```

```
                                        .EXTRN  CTL$GL_PCB, CTL$GL_PHD
                                        .EXTRN  PMS$GL_ERASEIO, WAIT_FOR_AST
                                        .EXTRN  CONTINUE_THREAD
                                        .EXTRN  SYS$QIO

                        01FC 00000 DO_ERASE:
                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8                       ; 1210
                58 00000000G    00  9E 00002         MOVAB   CTL$GL_PCB, R8
                5E              18  C2 00009         SUBL2   #24, SP                             ;
                57              01  D0 0000C         MOVL    #1, ERASE_STATUS                    ; 1285
                55              01  D0 0000F         MOVL    #1, STATUS                          ; 1286
                54        04    AC  D0 00012         MOVL    START_LBN, LBN                      ; 1287
                52        08    AC  D0 00016         MOVL    BLOCK_COUNT, COUNT                  ; 1288
                                52  D5 0001A 1$:     TSTL    COUNT                               ; 1289
                                03  14 0001C         BGTR    2$                                  ;
                              00A7  31 0001E         BRW     5$                                  ;
                53              68  D0 00021 2$:     MOVL    CTL$GL_PCB, PTR                     ; 1296
                          3E    A3  B6 00024         INCW    62(PTR)                             ; 1297
                          38    A3  B6 00027         INCW    56(PTR)                             ; 1298
                6E        0084  C3  7D 0002A         MOVQ    132(PTR), SAVE_PRIV                 ; 1299
          0084  C3 20000080     8F  C8 0002F         BISL2   #536871040, 132(PTR)                ; 1302
                53 00000000G    00  D0 00038         MOVL    CTL$GL_PHD, PTR                     ; 1303
            08  AE              63  7D 0003F         MOVQ    (PTR), SAVE_PRIV+8                  ; 1304
                63 20000080     8F  C8 00043         BISL2   #536871040, (PTR)                   ; 1307
                                7E  7C 0004A         CLRQ    -(SP)                               ; 1321
                                7E  D4 0004C         CLRL    -(SP)                               ;
                                54  DD 0004E         PUSHL   LBN                                 ;
          7E                    52  09 78 00050         ASHL    #9, COUNT, -(SP)                ;
                          0C    AC  9F 00054         PUSHAB  ERASE_PATTERN                       ;
                                5A  DD 00057         PUSHL   BASE                                ;
                          0000G CF  9F 00059         PUSHAB  CONTINUE_THREAD                     ;
                            30  AE  9F 0005D         PUSHAB  IOSTS                               ;
                        7E  0420  8F  3C 00060         MOVZWL  #1056, -(SP)                    ;
                            10  AC  DD 00065         PUSHL   CHANNEL                             ;
                                7E  D4 00068         CLRL    -(SP)                               ;
          00000000G     00        0C  FB 0006A         CALLS   #12, SYS$QIO                    ;
                        55        50  D0 00071         MOVL    R0, STATUS                      ;
                    00000000G     00  D6 00074         INCL    PMS$GL_ERASEIO                  ; 1323
                        63    08  AE  7D 0007A         MOVQ    SAVE_PRIV+8, (PTR)              ; 1325
                        53        68  D0 0007E         MOVL    CTL$GL_PCB, PTR                 ; 1327
                            38  A3  B7 00081         DECW    56(PTR)                         ; 1328
                            3E  A3  B7 00084         DECW    62(PTR)                         ; 1329
                0084    C3        6E  7D 00087         MOVQ    SAVE_PRIV, 132(PTR)             ; 1330
                        31        55  E9 0008C         BLBC    STATUS, 3$                      ; 1333
                0000G   CF        00  FB 0008F         CALLS   #0, WAIT_FOR_AST                ; 1336
                        55    10  AE  D0 00094         MOVL    IOSTS, STATUS                   ; 1337
                        25        55  E9 00098         BLBC    STATUS, 3$                      ; 1340
          56        12  AE 00000200  8F  C7 0009B         DIVL3   #512, IOSTS+2, BLOCKS_ERASED    ; 1349
          50            52        56  C3 000A4         SUBL3   BLOCKS_ERASED, COUNT, R0        ; 1350
                        52    FF  A0  9E 000A8         MOVAB   -1(R0), COUNT                   ;
                        54    01 A644 9E 000AC         MOVAB   1(BLOCKS_ERASED)[LBN], LBN      ; 1351
                        55    10  AE  3C 000B1         MOVZWL  IOSTS, STATUS                   ; 1357
                        0D        55  E8 000B5         BLBS    STATUS, 4$                      ;
                        0A        57  E9 000B8         BLBC    ERASE_STATUS, 4$                ; 1358
                        57        55  D0 000BB         MOVL    STATUS, ERASE_STATUS            ; 1360
                                  05  11 000BE         BRB     4$                              ; 1340
                        57        55  D0 000C0 3$:     MOVL    STATUS, ERASE_STATUS            ; 1367
```

```
                                        52  D4 000C3          CLRL    COUNT                                   ; 1368
                                      FF52  31 000C5 4$:      BRW     1$                                      ; 1289
                                50      57  D0 000C8 5$:      MOVL    ERASE_STATUS, R0                        ; 1375
                                            04 000CB          RET                                             ; 1377

; Routine Size:  204 bytes,    Routine Base:  $CODE$ + 007B

;  390          1378  1
;  391          1379  1 END
;  392          1380  0 ELUDOM



;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $CODE$                         327  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         32         0        1000        00:02.0



;                                       COMMAND QUALIFIERS

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

; Size:          327 code + 0 data bytes
; Run Time:        00:13.9
; Elapsed Time:    00:33.1
; Lines/CPU Min:     5956
; Lexemes/CPU-Min: 25407
; Memory Used:  155 pages
; Compilation Complete
```

EXTIDX.LIS

GTLCAT.LIS

GETLOC.LIS

EXTEND.LIS

EXTHDR.LIS

FILUTL.LIS

GETREQ.LIS

EXTCONTIG.LIS

ERASE.LIS

GETTIM.LIS

FIND.LIS

GETFIB.LIS

INIFC2.LIS

INIFCP.LIS

EXTFCB.LIS

FILESERV.LIS

FILESIZE.LIS

GETPTR.LIS