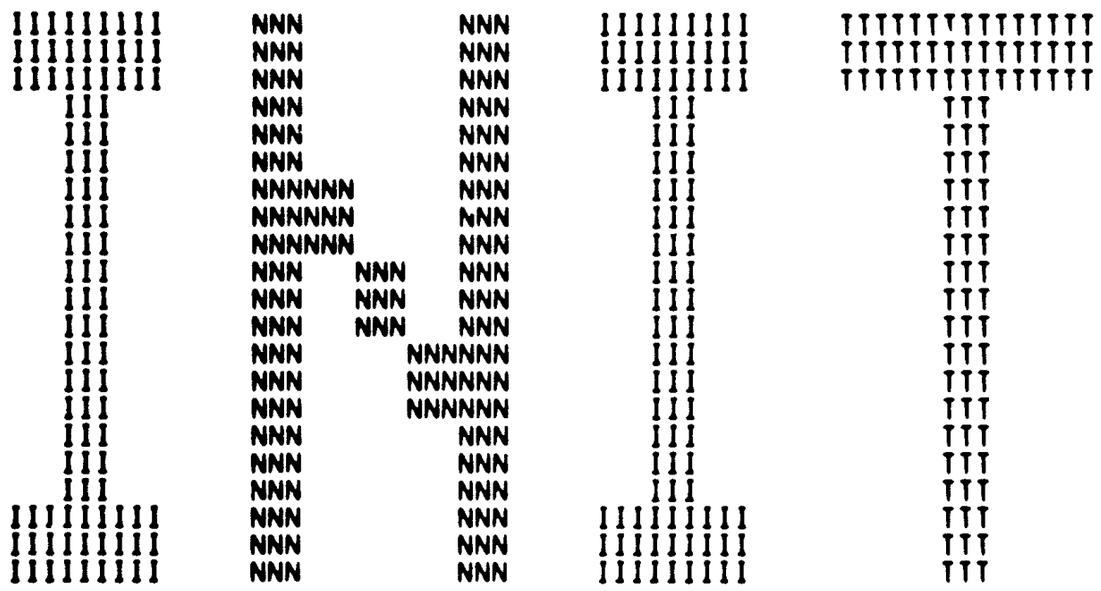
IIIIIIIII NNN NNN IIIIIIIII TTTTTTTTTTTTTTT
IIIIIIIII NNN NNN IIIIIIIII TTTTTTTTTTTTTTT
IIIIIIIII NNN NNN IIIIIIIII TTTTTTTTTTTTTTT
III NNN NNN III TTT
III NNN NNN III TTT
III NNN NNN III TTT
III NNNNNN NNN III TTT
III NNNNNN NNN III TTT
III NNNNNN NNN III TTT
III NNN NNN NNN III TTT
III NNN NNN NNN III TTT
III NNN NNN III TTT
III NNN NNNNNN III TTT
III NNN NNNNNN III TTT
III NNN NNNNNN III TTT
III NNN NNN III TTT
III NNN NNN III TTT
III NNN NNN III TTT
IIIIIIIII NNN NNN IIIIIIIII TTT
IIIIIIIII NNN NNN IIIIIIIII TTT
IIIIIIIII NNN NNN IIIIIIIII TTT

```
    IIIIII    NN      NN    IIIIII    DDDDDDDD      SSSSSSSS  KK      KK
    IIIIII    NN      NN    IIIIII    DDDDDDDD      SSSSSSSS  KK      KK
      II      NN      NN      II      DD      DD    SS        KK      KK
      II      NN      NN      II      DD      DD    SS        KK      KK
      II      NNNN    NN      II      DD      DD    SS        KK    KK
      II      NNNN    NN      II      DD      DD    SS        KK    KK
      II      NN  NN  NN      II      DD      DD      SSSSSS  KKKKKK
      II      NN  NN  NN      II      DD      DD      SSSSSS  KKKKKK
      II      NN    NNNN      II      DD      DD            SS  KK    KK
      II      NN    NNNN      II      DD      DD            SS  KK    KK
      II      NN      NN      II      DD      DD            SS  KK      KK      ....
      II      NN      NN      II      DD      DD            SS  KK      KK      ....
    IIIIII    NN      NN    IIIIII    DDDDDDDD    SSSSSSSS    KK      KK      ....
    IIIIII    NN      NN    IIIIII    DDDDDDDD    SSSSSSSS    KK      KK      ....

    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 INIDSK (
;     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:  INIT Utility Structure Level 1
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !
;    37          0037  1 !       This routine contains the main level logic to initialize a disk.
;    38          0038  1 !
;    39          0039  1 ! ENVIRONMENT:
;    40          0040  1 !
;    41          0041  1 !       STARLET operating system, including privileged system services
;    42          0042  1 !       and internal exec routines.
;    43          0043  1 !
;    44          0044  1 !--
;    45          0045  1 !
;    46          0046  1 !
;    47          0047  1 ! AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  9-Nov-1977  19:29
;    48          0048  1 !
;    49          0049  1 ! MODIFIED BY:
;    50          0050  1 !
;    51          0051  1 !       V03-004  MCN0140        Maria del C. Nasr       30-Nov-1983
;    52          0052  1 !               As part of the new CLI interface change, eliminate
;    53          0053  1 !               RECORD_PROT processing since that qualifier was not implemented.
;    54          0054  1 !
;    55          0055  1 !       V03-003 CWH3003         CW Hobbs                7-Oct-1983
;    56          0056  1 !               Perform an IO$_AVAILABLE function to undo the IO$_PACKACK.
;    57          0057  1 !
```

```
;    58          0058  1 !       V03-002 STJ3090         Steven T. Jeffreys,     25-Apr-1983
;    59          0059  1 !               Add support for /[NO]ERASE.
;    60          0060  1 !
;    61          0061  1 !       V03-001 ACG0283         Andrew C. Goldstein,    8-Apr-1982  14:09
;    62          0062  1 !               Disable bad block processing on MSCP disks;
;    63          0063  1 !               limit ODS-1 max files to 65500.
;    64          0064  1 !
;    65          0065  1 !       V02-006 LMP0006         L. Mark Pilant,         6-Jan-1982  10:40
;    66          0066  1 !               Correct a bug so that structure level 1 can only have a
;    67          0067  1 !               cluster size of 1.
;    68          0068  1 !
;    69          0069  1 !       V02-005 ACG0240         Andrew C. Goldstein,    11-Dec-1981  22:13
;    70          0070  1 !               Make default file protection more restrictive,
;    71          0071  1 !               make default index file position beginning for small disks
;    72          0072  1 !
;    73          0073  1 !       V02-004 LMP0001         L. Mark Pilant          4-Nov-1981  16:35
;    74          0074  1 !               Create a multi-header index file if the number of headers
;    75          0075  1 !               to be created cannot be contained in a single header
;    76          0076  1 !
;    77          0077  1 !       V02-003 RLRDENS         Robert L. Rappaport             6-Oct-1980
;    78          0078  1 !               Added /DENSITY=1 and /DENSITY=2 support for RX02's
;    79          0079  1 !
;    80          0080  1 !**
;    81          0081  1
;    82          0082  1
;    83          0083  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    84          0084  1 REQUIRE 'SRC$:INIDEF.B32';
;    85          0375  1 REQUIRE 'LIBD$:[VMSLIB.OBJ]INITMSG.B32';
```

```
;    87          0507  1 !+
;    88          0508  1 !
;    89          0509  1 ! Impure data specific to disk initialization.
;    90          0510  1 !
;    91          0511  1 !-
;    92          0512  1
;    93          0513  1 GLOBAL LITERAL
;    94          0514  1         BADBLOCK_MAX    = 128,          ! maximum size of bad block table
;    95          0515  1         ALLOC_MAX       = BADBLOCK_MAX + 8; ! total size of allocation table
;    96          0516  1
;    97          0517  1 GLOBAL
;    98          0518  1         BUFFER          : BBLOCK [512], ! all purpose I/O buffer
;    99          0519  1
;   100          0520  1 !
;   101          0521  1 ! Allocation table. Consists of 2 parallel tables, for size and LBN of
;   102          0522  1 ! allocated areas. Each contains 1 entry for each piece of the disk which
;   103          0523  1 ! is allocated to something.
;   104          0524  1 !
;   105          0525  1         ALLOC_TABLE_CNT : VECTOR [ALLOC_MAX],
;   106          0526  1         ALLOC_TABLE_LBN : VECTOR [ALLOC_MAX];
;   107          0527  1 !
;   108          0528  1 ! Various parts of the allocation table have dedicated meanings.
;   109          0529  1 !
;   110          0530  1
;   111          0531  1 GLOBAL LITERAL
;   112          0532  1         BOOTBLOCK_IDX   = 0,
;   113          0533  1         HOMEBLOCK1_IDX  = 1,
;   114          0534  1         HOMEBLOCK2_IDX  = 2,
;   115          0535  1         IDXHDR2_IDX     = 3,
;   116          0536  1         IDXFILE_IDX     = 4,
;   117          0537  1         BITMAP_IDX      = 5,
;   118          0538  1         MFD_IDX         = 6,
;   119          0539  1         VOLEND_IDX      = 7,
;   120          0540  1         BADBLOCK_IDX    = 8;
;   121          0541  1
;   122          0542  1 GLOBAL BIND
;   123          0543  1         BOOTBLOCK_CNT   = ALLOC_TABLE_CNT [BOOTBLOCK_IDX],
;   124          0544  1         BOOTBLOCK_LBN   = ALLOC_TABLE_LBN [BOOTBLOCK_IDX],
;   125          0545  1         HOMEBLOCK1_CNT  = ALLOC_TABLE_CNT [HOMEBLOCK1_IDX],
;   126          0546  1         HOMEBLOCK1_LBN  = ALLOC_TABLE_LBN [HOMEBLOCK1_IDX],
;   127          0547  1         HOMEBLOCK2_CNT  = ALLOC_TABLE_CNT [HOMEBLOCK2_IDX],
;   128          0548  1         HOMEBLOCK2_LBN  = ALLOC_TABLE_LBN [HOMEBLOCK2_IDX],
;   129          0549  1         IDXHDR2_CNT     = ALLOC_TABLE_CNT [IDXHDR2_IDX],
;   130          0550  1         IDXHDR2_LBN     = ALLOC_TABLE_LBN [IDXHDR2_IDX],
;   131          0551  1         IDXFILE_CNT     = ALLOC_TABLE_CNT [IDXFILE_IDX],
;   132          0552  1         IDXFILE_LBN     = ALLOC_TABLE_LBN [IDXFILE_IDX],
;   133          0553  1         BITMAP_CNT      = ALLOC_TABLE_CNT [BITMAP_IDX],
;   134          0554  1         BITMAP_LBN      = ALLOC_TABLE_LBN [BITMAP_IDX],
;   135          0555  1         MFD_CNT         = ALLOC_TABLE_CNT [MFD_IDX],
;   136          0556  1         MFD_LBN         = ALLOC_TABLE_LBN [MFD_IDX],
;   137          0557  1         VOLEND_CNT      = ALLOC_TABLE_CNT [VOLEND_IDX],
;   138          0558  1         VOLEND_LBN      = ALLOC_TABLE_LBN [VOLEND_IDX],                                    ;
;   139          0559  1         BADBLOCK_CNT    = ALLOC_TABLE_CNT [BADBLOCK_IDX]        : VECTOR,
;   140          0560  1         BADBLOCK_LBN    = ALLOC_TABLE_LBN [BADBLOCK_IDX]        : VECTOR;
;   141          0561  1
;   142          0562  1 !
;   143          0563  1 ! Other globals
```

```
;  144          0564  1 !
;  145          0565  1
;  146          0566  1 GLOBAL BIND
;  147          0567  1         HOME_BLOCK      = BUFFER : BBLOCK; ! buffer to read original home block
;  148          0568  1
;  149          0569  1 GLOBAL
;  150          0570  1         VOLUME_SIZE,                    ! volume size rounded up to next cluster
;  151          0571  1         REAL_HOMEBLOCK,                 ! LBN of actual secondary home block
;  152          0572  1         BADBLOCK_TOTAL,                 ! current count of bad areas
;  153          0573  1         SERIAL_NUMBER,                  ! serial number of disk pack
;  154          0574  1         HOMEBLOCK_LBN,                  ! LBN of home block read
;  155          0575  1         MOUNT_OPTIONS   : BITVECTOR [64]; ! used by some subroutines
;  156          0576  1
;  157          0577  1 !
;  158          0578  1 ! Default parameters. All have global names so they can be patched.
;  159          0579  1 !
;  160          0580  1
;  161          0581  1 GLOBAL BIND
;  162          0582  1         DEF_PRV_PROT    = UPLIT (%X'FF00'),     ! default private protection
;  163          0583  1         DEF_GRP_PROT    = UPLIT (%X'F000'),     ! default group protection
;  164          0584  1         DEF_SYS_PROT    = UPLIT (%X'0000'),     ! default system protection
;  165          0585  1         DEF_SHR_PROT    = UPLIT (%X'0000'),     ! default shared protection
;  166          0586  1         DEF_FIL_PROT    = UPLIT (%X'FA00'),     ! default file protection
;  167          0587  1         DEF_FIL_PROT1   = UPLIT (%X'FE00'),     ! default file protection, ODS1
;  168          0588  1         DEF_EXTEND      = UPLIT (5),            ! default file extend
;  169          0589  1         DEF_WINDOW      = UPLIT (7),            ! default window size
;  170          0590  1         DEF_ACCESSED    = UPLIT (3),            ! default directory LRU limit
;  171          0591  1         DEF_HEADERS     = UPLIT (16),           ! default initial # headers
;  172          0592  1         SMALL_DISK      = UPLIT (4096);         ! maximum size of a "small" disk
;  173          0593  1
;  174          0594  1 FORWARD ROUTINE
;  175          0595  1         INIT_DISK       : NOVALUE,      ! main routine
;  176          0596  1         ERASE_DISK      : NOVALUE,      ! conditionally erase disk
;  177          0597  1         VOL_AVAIL       : NOVALUE;      ! issue the IO$_AVAILABLE function at exit
;  178          0598  1
;  179          0599  1
```

```
;  181          0600  1 GLOBAL ROUTINE INIT_DISK : NOVALUE =
;  182          0601  1
;  183          0602  1 !++
;  184          0603  1 !
;  185          0604  1 ! FUNCTIONAL DESCRIPTION:
;  186          0605  1 !
;  187          0606  1 !       This routine contains the main line logic specific to initializing
;  188          0607  1 !       a disk. It sets up the bad block table, allocates the system files,
;  189          0608  1 !       initializes the storage map, and initializes the contents of the
;  190          0609  1 !       other files.
;  191          0610  1 !
;  192          0611  1 !
;  193          0612  1 ! CALLING SEQUENCE:
;  194          0613  1 !       INIT_DISK ()
;  195          0614  1 !
;  196          0615  1 ! INPUT PARAMETERS:
;  197          0616  1 !       NONE
;  198          0617  1 !
;  199          0618  1 ! IMPLICIT INPUTS:
;  200          0619  1 !       parser data base
;  201          0620  1 !       own storage of this module
;  202          0621  1 !
;  203          0622  1 ! OUTPUT PARAMETERS:
;  204          0623  1 !       NONE
;  205          0624  1 !
;  206          0625  1 ! IMPLICIT OUTPUTS:
;  207          0626  1 !       NONE
;  208          0627  1 !
;  209          0628  1 ! ROUTINE VALUE:
;  210          0629  1 !       NONE
;  211          0630  1 !
;  212          0631  1 ! SIDE EFFECTS:
;  213          0632  1 !       disk volume initialized
;  214          0633  1 !
;  215          0634  1 !--
;  216          0635  1
;  217          0636  2 BEGIN
;  218          0637  2
;  219          0638  2 LOCAL
;  220          0639  2         J,                              ! device table index
;  221          0640  2         C,                              ! available clusters on volume
;  222          0641  2         VOLUME_OWNER,                   ! previous owner UIC of volume
;  223          0642  2         STATUS,                         ! system service status
;  224          0643  2         IO_STATUS       : VECTOR [2],   ! I/O status block
;  225          0644  2         PRIVILEGE_MASK  : REF BBLOCK;   ! process privilege mask
;  226          0645  2
;  227          0646  2 OWN
;  228          0647  2         EXIT_BLOCK      : VECTOR [5]    ! exit handler block
;  229          0648  2                 PRESET ([0] = 0,                ! system link longword
;  230          0649  2                         [1] = VOL_AVAIL,        ! exit routine entry point
;  231          0650  2                         [2] = 1,                ! one parameter, the status
;  232          0651  2                         [3] = EXIT_BLOCK [4],   ! status at end of block
;  233          0652  2                         [4] = 0);               ! space for status, not used
;  234          0653  2
;  235          0654  2 EXTERNAL
;  236          0655  2         INIT_OPTIONS    : BITVECTOR,    ! command options
;  237          0656  2         DEVICE_CHAR     : BBLOCK,       ! disk device characteristics
```

```
;  238          0657  2          DEVCHAR_DESC    : VECTOR,          ! device characterictics descriptor
;  239          0658  2          CHANNEL,                           ! channel assigned to device
;  240          0659  2          PROCESS_UIC,                       ! UIC of this process
;  241          0660  2          PROTECTION,                        ! volume protection
;  242          0661  2          FILE_PROT,                         ! volume default file protection
;  243          0662  2          OWNER_UIC,                         ! volume owner UIC
;  244          0663  2          EXTENSION,                         ! volume default file extend
;  245          0664  2          WINDOW,                            ! volume default window size
;  246          0665  2          ACCESSED,                          ! volume default directory LRU limit
;  247          0666  2          HEADERS,                           ! initial file header allocation
;  248          0667  2          CLUSTER,                           ! volume cluster factor
;  249          0668  2          MAXIMUM,                           ! maximum number of files
;  250          0669  2          INDEX,                             ! index file start LBN
;  251          0670  2          CTL$GL_PHD      : REF BBLOCK ADDRESSING_MODE (ABSOLUTE);
;  252          0671  2                                             ! vector page pointer to process header
;  253          0672  2
;  254          0673  2 EXTERNAL ROUTINE
;  255          0674  2          SET_VALID,                         ! set software volume valid
;  256          0675  2          READ_HOMEBLOCK,                    ! read volume home block
;  257          0676  2          CLEAR_VALID,                       ! clear software volume valid
;  258          0677  2          INIT_BADBLOCKS,                    ! do bad block processing
;  259          0678  2          INIT_ALLOCATE,                     ! allocate file structures
;  260          0679  2          INIT_BITMAP,                       ! set up the storage map
;  261          0680  2          INIT_INDEX,                        ! initialize contents of index file
;  262          0681  2          INIT_INDEX1,                       ! initialize contents of index file, ODS1
;  263          0682  2          INIT_MFD;                          ! initialize contents of MFD
;  264          0683  2
;  265          0684  2
;  266          0685  2 ! Set the software volume valid and do a packack. Then read the characterictics
;  267          0686  2 ! again, since the packack may have caused the driver to discover new data
;  268          0687  2 ! about the device.  Declare an exit handler to give the IO$_AVAILABLE function
;  269          0688  2 ! to undo the IO$_PACKACK.
;  270          0689  2 !
;  271          0690  2
;  272          0691  2 KERNEL_CALL (SET_VALID);
;  273          0692  2
;  274          0693  2 STATUS = $DCLEXH (DESBLK=EXIT_BLOCK);
;  275          0694  2 IF NOT .STATUS THEN ERR_EXIT (.STATUS);
;  276          0695  2
;  277        P 0696  2 STATUS = $QIOW (CHAN = .CHANNEL,
;  278        P 0697  2                 FUNC = IO$_PACKACK,
;  279          0698  2                 IOSB = IO_STATUS);
;  280          0699  2 IF .STATUS THEN STATUS = .IO_STATUS<0,16>;
;  281          0700  2 IF NOT .STATUS AND .STATUS NEQ SS$_ILLIOFUNC
;  282          0701  2 THEN ERR_EXIT (.STATUS);
;  283          0702  2
;  284          0703  2 $GETCHN (CHAN = .CHANNEL, PRIBUF = DEVCHAR_DESC);
;  285          0704  2
;  286          0705  2 ! If the process does not have VOLPRO privilege, attempt to read the old
;  287          0706  2 ! home block. If there is a home block, check the volume owner UIC. It must be
;  288          0707  2 ! zero (unowned) or match the UIC of the user.
;  289          0708  2 !
;  290          0709  2
;  291          0710  2 PRIVILEGE_MASK = CTL$GL_PHD[PHD$Q_PRIVMSK];
;  292          0711  2 IF NOT .PRIVILEGE_MASK[PRV$V_VOLPRO]
;  293          0712  2 THEN
;  294          0713  3     BEGIN
```

```
;  295          0714  3     STATUS = READ_HOMEBLOCK (UPLIT (0, 0), 0);
;  296          0715  3
;  297          0716  3     IF .STATUS
;  298          0717  3     OR .STATUS EQL SS$_INCVOLLABEL
;  299          0718  3     THEN
;  300          0719  4         BEGIN
;  301          0720  4         IF .HOME_BLOCK[HM2$B_STRUCLEV] EQL 2
;  302          0721  4         THEN VOLUME_OWNER = .HOME_BLOCK[HM2$L_VOLOWNER]
;  303          0722  4         ELSE
;  304          0723  5             BEGIN
;  305          0724  5             VOLUME_OWNER = .(HOME_BLOCK[HM1$W_VOLOWNER])<0,8>;
;  306          0725  5             VOLUME_OWNER<16,8> = .(HOME_BLOCK[HM1$W_VOLOWNER])<8,8>;
;  307          0726  4             END;
;  308          0727  4
;  309          0728  4         IF .VOLUME_OWNER NEQ 0
;  310          0729  4         AND .VOLUME_OWNER NEQ .PROCESS_UIC
;  311          0730  4         THEN ERR_EXIT (SS$_NOPRIV);
;  312          0731  3         END;
;  313          0732  2     END;
;  314          0733  2
;  315          0734  2 ! Establish defaults for volume parameters not specified in the command.
;  316          0735  2 ! Also, if structure level 1 is specified, use level 1 defaults and disallow
;  317          0736  2 ! options that are not supported.
;  318          0737  2 !
;  319          0738  2
;  320          0739  2 IF .INIT_OPTIONS[OPT_DENSITY] THEN
;  321          0740  3 BEGIN
;  322          0741  3      IF .DEVICE_CHAR[DIB$B_DEVTYPE] NEQ DT$_RX02 THEN ERR_EXIT (INIT$_ILLOPT);
;  323          0742  3
;  324          0743  3      IF .INIT_OPTIONS[OPT_DENS_SING]
;  325          0744  4         THEN BEGIN
;  326        P 0745  4              STATUS = $QIOW (CHAN = .CHANNEL,
;  327        P 0746  4                              IOSB = IO_STATUS[0],
;  328        P 0747  4                              FUNC = IO$_FORMAT,
;  329          0748  4                              P1 = 1);
;  330          0749  4              IF .STATUS THEN STATUS = .IO_STATUS<0,16>;
;  331          0750  4              IF NOT .STATUS THEN ERR_EXIT (.STATUS);
;  332          0751  4             END
;  333          0752  3         ELSE IF .INIT_OPTIONS[OPT_DENS_DOUB]
;  334          0753  4              THEN BEGIN
;  335        P 0754  4                   STATUS = $QIOW (CHAN = .CHANNEL,
;  336        P 0755  4                                   IOSB = IO_STATUS[0],
;  337        P 0756  4                                   FUNC = IO$_FORMAT,
;  338          0757  4                                   P1 = 2);
;  339          0758  4                   IF .STATUS THEN STATUS = .IO_STATUS<0,16>;
;  340          0759  4                   IF NOT .STATUS THEN ERR_EXIT (.STATUS);
;  341          0760  4                  END
;  342          0761  3              ELSE ERR_EXIT (INIT$_BADDENS);
;  343          0762  3
;  344        P 0763  3      STATUS = $QIOW (CHAN = .CHANNEL,
;  345        P 0764  3                      FUNC = IO$_PACKACK,
;  346          0765  3                      IOSB = IO_STATUS);
;  347          0766  3      IF .STATUS THEN STATUS = .IO_STATUS<0,16>;
;  348          0767  3      IF NOT .STATUS AND .STATUS NEQ SS$_ILLIOFUNC
;  349          0768  3      THEN ERR_EXIT (.STATUS);
;  350          0769  3
;  351          0770  3      $GETCHN (CHAN = .CHANNEL, PRIBUF = DEVCHAR_DESC);
```

```
;  352          0771  2 END;
;  353          0772  2
;  354          0773  2 IF .INIT_OPTIONS[OPT_STRUCTURE1]
;  355          0774  2 THEN
;  356          0775  3     BEGIN
;  357          0776  3     MAP INIT_OPTIONS : VECTOR;
;  358          0777  3     IF (.INIT_OPTIONS[0] AND LEVEL2_OPTIONS) NEQ 0
;  359          0778  3     AND (.INIT_OPTIONS[1] AND LEVEL2_OPTIONS2) NEQ 0
;  360          0779  3     THEN ERR_EXIT (INIT$_NOTSTRUC1);
;  361          0780  3
;  362          0781  3     IF .DEVICE_CHAR[DIB$L_MAXBLOCK] GTRU 255^12
;  363          0782  3     THEN ERR_EXIT (INIT$_ARGECNT);
;  364          0783  2     END;
;  365          0784  2
;  366          0785  2 IF NOT .INIT_OPTIONS[OPT_PROTECTION]
;  367          0786  2 THEN
;  368          0787  3     BEGIN
;  369          0788  3     IF .INIT_OPTIONS[OPT_SYSTEM]
;  370          0789  3     THEN PROTECTION = .DEF_SYS_PROT
;  371          0790  3     ELSE IF .INIT_OPTIONS[OPT_GROUP]
;  372          0791  3     THEN PROTECTION = .DEF_GRP_PROT
;  373          0792  3     ELSE PROTECTION = .DEF_PRV_PROT;
;  374          0793  3     IF .INIT_OPTIONS[OPT_SHARE]
;  375          0794  3     THEN PROTECTION = .PROTECTION AND .DEF_SHR_PROT;
;  376          0795  2     END;
;  377          0796  2
;  378          0797  2 IF NOT .INIT_OPTIONS[OPT_FILE_PROT]
;  379          0798  2 THEN
;  380          0799  2     IF .INIT_OPTIONS[OPT_STRUCTURE1]
;  381          0800  2     THEN FILE_PROT = .DEF_FIL_PROT1
;  382          0801  2     ELSE FILE_PROT = .DEF_FIL_PROT;
;  383          0802  2
;  384          0803  2 IF NOT .INIT_OPTIONS[OPT_EXTENSION]
;  385          0804  2 THEN EXTENSION = .DEF_EXTEND;
;  386          0805  2
;  387          0806  2 IF NOT .INIT_OPTIONS[OPT_WINDOW]
;  388          0807  2 THEN WINDOW = .DEF_WINDOW;
;  389          0808  2
;  390          0809  2 IF NOT .INIT_OPTIONS[OPT_ACCESSED]
;  391          0810  2 THEN ACCESSED = .DEF_ACCESSED;
;  392          0811  2
;  393          0812  2 IF NOT .INIT_OPTIONS[OPT_HEADERS]
;  394          0813  2 THEN HEADERS = .DEF_HEADERS;
;  395          0814  2
;  396          0815  2 IF NOT .INIT_OPTIONS[OPT_OWNER_UIC]
;  397          0816  2 THEN
;  398          0817  3     BEGIN
;  399          0818  3     OWNER_UIC = .PROCESS_UIC;
;  400          0819  3     IF .INIT_OPTIONS[OPT_GROUP]
;  401          0820  3     THEN OWNER_UIC<0,16> = 0;
;  402          0821  3     IF .INIT_OPTIONS[OPT_SYSTEM]
;  403          0822  3     THEN OWNER_UIC = %X'T0001';
;  404          0823  2     END;
;  405          0824  2
;  406          0825  2 IF .INIT_OPTIONS[OPT_STRUCTURE1]
;  407          0826  2 THEN CLUSTER = 1
;  408          0827  2 ELSE
```

```
;  409          0828  3     BEGIN
;  410          0829  3     IF NOT .INIT_OPTIONS[OPT_CLUSTER]
;  411          0830  3     THEN
;  412          0831  3         IF .DEVICE_CHAR[DIB$L_MAXBLOCK] LEQU 50000
;  413          0832  3             THEN CLUSTER = 1
;  414          0833  3             ELSE CLUSTER = 3;
;  415          0834  2     END;
;  416          0835  2
;  417          0836  2 IF NOT .INIT_OPTIONS[OPT_MAXIMUM]
;  418          0837  2 THEN MAXIMUM = .DEVICE_CHAR[DIB$L_MAXBLOCK] / ((.CLUSTER+1)*2);
;  419          0838  2
;  420          0839  2 IF NOT .INIT_OPTIONS[OPT_EXP_VER]
;  421          0840  3 AND (.DEVICE_CHAR[DIB$L_MAXBLOCK] LEQU .SMALL_DISK
;  422          0841  3      OR .DEVICE_CHAR[DEV$V_RCT])
;  423          0842  2 THEN INIT_OPTIONS[OPT_VERIFIED] = 0;
;  424          0843  2
;  425          0844  2 ! Now verify the parameters against the volume size and characteristics.
;  426          0845  2 !
;  427          0846  2 ! Compute volume size, rounded up to next cluster boundary.
;  428          0847  2 !
;  429          0848  2
;  430          0849  2 VOLUME_SIZE = (.DEVICE_CHAR[DIB$L_MAXBLOCK] + .CLUSTER - 1) / .CLUSTER * .CLUSTER;
;  431          0850  2
;  432          0851  2 ! Check the cluster factor against its lower bound such that the storage map
;  433          0852  2 ! does not exceed 255 blocks. Also check for a reasonable minimum number of
;  434          0853  2 ! clusters.
;  435          0854  2 !
;  436          0855  2
;  437          0856  2 IF .VOLUME_SIZE / .CLUSTER GTRU 255^12
;  438          0857  2 OR .VOLUME_SIZE / .CLUSTER LSS 50
;  439          0858  2 THEN ERR_EXIT (INIT$_CLUSTER);
;  440          0859  2
;  441          0860  2 ! Check maximum number of files against number of clusters.
;  442          0861  2 !
;  443          0862  2
;  444          0863  2 C = .VOLUME_SIZE / (.CLUSTER+1);
;  445          0864  2 IF .MAXIMUM GTR .C
;  446          0865  2 THEN MAXIMUM = .C;
;  447          0866  2
;  448          0867  2 ! Check initial index file size against max files.
;  449          0868  2 !
;  450          0869  2
;  451          0870  2 IF .HEADERS LSSU 16 THEN HEADERS = 16;
;  452          0871  2 IF .INIT_OPTIONS[OPT_STRUCTURE1]
;  453          0872  2 THEN IF .MAXIMUM GTRU 65500 THEN MAXIMUM = 65500;
;  454          0873  2 IF .HEADERS GTRU .MAXIMUM THEN HEADERS = .MAXIMUM;
;  455          0874  2
;  456          0875  2 ! Check the position of ths initial index file.
;  457          0876  2 !
;  458          0877  2
;  459          0878  2 IF .INIT_OPTIONS[OPT_INDEX_BEG]
;  460          0879  2 THEN INDEX = 0
;  461          0880  2
;  462          0881  2 ELSE IF .INIT_OPTIONS[OPT_INDEX_END]
;  463          0882  2 THEN INDEX = .DEVICE_CHAR[DIB$L_MAXBLOCK] - 1
;  464          0883  2
;  465          0884  2 ELSE IF .INIT_OPTIONS[OPT_INDEX_MID] OR NOT .INIT_OPTIONS[OPT_INDEX_LBN]
```

```
;  466          0885  2 THEN
;  467          0886  3     BEGIN
;  468          0887  3     IF .DEVICE_CHAR[DIB$L_MAXBLOCK] LEQU .SMALL_DISK
;  469          0888  3     THEN INDEX = 0
;  470          0889  3     ELSE INDEX = .DEVICE_CHAR[DIB$L_MAXBLOCK] / 2;
;  471          0890  2     END;
;  472          0891  2
;  473          0892  2 IF .INDEX GTRU .DEVICE_CHAR[DIB$L_MAXBLOCK]
;  474          0893  2 THEN ERR_EXIT (INIT$_INDEX);
;  475          0894  2
;  476          0895  2 ! Now call the routines that do the work of initializing.
;  477          0896  2 !
;  478          0897  2
;  479          0898  2 INIT_BADBLOCKS ();
;  480          0899  2 ERASE_DISK (.DEVICE_CHAR[DIB$L_MAXBLOCK]-1);
;  481          0900  2 INIT_ALLOCATE ();
;  482          0901  2 INIT_BITMAP ();
;  483          0902  2 IF .INIT_OPTIONS[OPT_STRUCTURE1]
;  484          0903  2 THEN INIT_INDEX1 ()
;  485          0904  2 ELSE INIT_INDEX ();
;  486          0905  2 INIT_MFD ();
;  487          0906  2
;  488          0907  2 ! Issue the IO$_AVAILABLE call, cancel the exit handler and clear the valid bit
;  489          0908  2 !
;  490          0909  2
;  491          0910  2 VOL_AVAIL ();
;  492          0911  2 STATUS = $CANEXH (DESBLK=EXIT_BLOCK);
;  493          0912  2 IF NOT .STATUS THEN ERR_EXIT (.STATUS);
;  494          0913  2 KERNEL_CALL (CLEAR_VALID);
;  495          0914  2
;  496          0915  1 END;                                    ! end of routine INIT_DISK


                                                .TITLE  INIDSK
                                                .IDENT  \V04-000\

                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                                0000FF00  00000 P.AAA:  .LONG   65280                   ;
                                0000F000  00004 P.AAB:  .LONG   61440                   ;
                                00000000  00008 P.AAC:  .LONG   0                       ;
                                00000000  0000C P.AAD:  .LONG   0                       ;
                                0000FA00  00010 P.AAE:  .LONG   64000                   ;
                                0000FE00  00014 P.AAF:  .LONG   65024                   ;
                                00000005  00018 P.AAG:  .LONG   5                       ;
                                00000007  0001C P.AAH:  .LONG   7                       ;
                                00000003  00020 P.AAI:  .LONG   3                       ;
                                00000010  00024 P.AAJ:  .LONG   16                      ;
                                00001000  00028 P.AAK:  .LONG   4096                    ;
                    00000000    00000000  0002C P.AAL:  .LONG   0, 0                   ;

                                                .PSECT  $OWN$,NOEXE,2

                                00000000  00000 EXIT_BLOCK:
                                                .LONG   0                       ;
                                00000000V 00004         .ADDRESS VOL_AVAIL      ;
                                00000001  00008         .LONG   1               ;
```

```
00000000' 0000C          .ADDRESS EXIT_BLOCK+16
00000000  00010          .LONG   0

                         .PSECT  $GLOBAL$,NOEXE,2

                00000 BUFFER::.BLKB    512
                00200 ALLOC_TABLE_CNT::
                         .BLKB    544
                00420 ALLOC_TABLE_LBN::
                         .BLKB    544
                00640 VOLUME_SIZE::
                         .BLKB    4
                00644 REAL_HOMEBLOCK::
                         .BLKB    4
                00648 BADBLOCK_TOTAL::
                         .BLKB    4
                0064C SERIAL_NUMBER::
                         .BLKB    4
                00650 HOMEBLOCK_LBN::
                         .BLKB    4
                00654 MOUNT_OPTIONS::
                         .BLKB    8

                      BADBLOCK_MAX==        128
                      ALLOC_MAX==           136
                      BOOTBLOCK_IDX==       0
                      HOMEBLOCK1_IDX==      1
                      HOMEBLOCK2_IDX==      2
                      IDXHDR2_IDX==         3
                      IDXFILE_IDX==         4
                      BITMAP_IDX==          5
                      MFD_IDX==             6
                      VOLEND_IDX==          7
                      BADBLOCK_IDX==        8
                      BOOTBLOCK_CNT==       ALLOC_TABLE_CNT
                      BOOTBLOCK_LBN==       ALLOC_TABLE_LBN
                      HOMEBLOCK1_CNT==      ALLOC_TABLE_CNT+4
                      HOMEBLOCK1_LBN==      ALLOC_TABLE_LBN+4
                      HOMEBLOCK2_CNT==      ALLOC_TABLE_CNT+8
                      HOMEBLOCK2_LBN==      ALLOC_TABLE_LBN+8
                      IDXHDR2_CNT==         ALLOC_TABLE_CNT+12
                      IDXHDR2_LBN==         ALLOC_TABLE_LBN+12
                      IDXFILE_CNT==         ALLOC_TABLE_CNT+16
                      IDXFILE_LBN==         ALLOC_TABLE_LBN+16
                      BITMAP_CNT==          ALLOC_TABLE_CNT+20
                      BITMAP_LBN==          ALLOC_TABLE_LBN+20
                      MFD_CNT==             ALLOC_TABLE_CNT+24
                      MFD_LBN==             ALLOC_TABLE_LBN+24
                      VOLEND_CNT==          ALLOC_TABLE_CNT+28
                      VOLEND_LBN==          ALLOC_TABLE_LBN+28
                      BADBLOCK_CNT==        ALLOC_TABLE_CNT+32
                      BADBLOCK_LBN==        ALLOC_TABLE_LBN+32
                      HOME_BLOCK==          BUFFER
                      DEF_PRV_PROT==        P.AAA
                      DEF_GRP_PROT==        P.AAB
                      DEF_SYS_PROT==        P.AAC
                      DEF_SHR_PROT==        P.AAD
```

```
                        DEF_FIL_PROT==      P.AAE
                        DEF_FIL_PROT1==     P.AAF
                        DEF_EXTEND==        P.AAG
                        DEF_WINDOW==        P.AAH
                        DEF_ACCESSED==      P.AAI
                        DEF_HEADERS==       P.AAJ
                        SMALL_DISK==        P.AAK
                                .EXTRN  INIT_OPTIONS, DEVICE_CHAR
                                .EXTRN  DEVCHAR_DESC, CHANNEL
                                .EXTRN  PROCESS_UIC, PROTECTION
                                .EXTRN  FILE_PROT, OWNER_UIC
                                .EXTRN  EXTENSION, WINDOW
                                .EXTRN  ACCESSED, HEADERS
                                .EXTRN  CLUSTER, MAXIMUM
                                .EXTRN  INDEX, CTL$GL_PHD
                                .EXTRN  SET_VALID, READ_HOMEBLOCK
                                .EXTRN  CLEAR_VALID, INIT_BADBLOCKS
                                .EXTRN  INIT_ALLOCATE, INIT_BITMAP
                                .EXTRN  INIT_INDEX, INIT_INDEX1
                                .EXTRN  INIT_MFD, SYS$CMKRNL
                                .EXTRN  SYS$DCLEXH, SYS$QIOW
                                .EXTRN  SYS$GETCHN, SYS$CANEXH

                                .PSECT  $CODE$,NOWRT,2

                    0FFC 00000          .ENTRY  INIT_DISK, Save R2,R3,R4,R5,R6,R7,R8,R9,-    ; 0600
                                                R10,R11
5B      0000G   CF  9E 00002            MOVAB   HEADERS, R11
5A      0000G   CF  9E 00007            MOVAB   CHANNEL, R10
59      0000G   CF  9E 0000C            MOVAB   MAXIMUM, R9
58      0000'   CF  9E 00011            MOVAB   VOLUME_SIZE, R8
57  00000000G   00  9E 00016            MOVAB   SYS$QIOW, R7
56      0000G   CF  9E 0001D            MOVAB   DEVICE_CHAR+112, R6
55      0000'   CF  9E 00022            MOVAB   SMALL_DISK, R5
54  00000000G   00  9E 00027            MOVAB   LIB$STOP, R4
53      0000G   CF  9E 0002E            MOVAB   INIT_OPTIONS, R3
5E              08  C2 00033            SUBL2   #8, SP
                7E  D4 00036            CLRL    -(SP)                                        ; 0691
                5E  DD 00038            PUSHL   SP
        0000G   CF  9F 0003A            PUSHAB  SET_VALID
00000000G   9F      03  FB 0003E        CALLS   #3, @#SYS$CMKRNL
        0000'   CF  9F 00045            PUSHAB  EXIT_BLOCK                                   ; 0693
00000000G   00      01  FB 00049        CALLS   #1, SYS$DCLEXH
52              50  D0 00050            MOVL    R0, STATUS
05              52  E8 00053            BLBS    STATUS, 1$                                   ; 0694
                52  DD 00056            PUSHL   STATUS
64              01  FB 00058            CALLS   #1, LIB$STOP
                7E  7C 0005B 1$:        CLRQ    -(SP)                                        ; 0698
                7E  7C 0005D            CLRQ    -(SP)
                7E  7C 0005F            CLRQ    -(SP)
                7E  7C 00061            CLRQ    -(SP)
            20  AE  9F 00063            PUSHAB  IO_STATUS
                08  DD 00066            PUSHL   #8
                6A  DD 00068            PUSHL   CHANNEL
                7F  D4 0006A            CLRL    -(SP)
67              0C  FB 0006C            CALLS   #12, SYS$QIOW
52              50  D0 0006F            MOVL    R0, STATUS
```

```
                        06              52  E9 00072          BLBC    STATUS, 2$                                      ;0699
                        52              6E  3C 00075          MOVZWL  IO_STATUS, STATUS
                        0E              52  E8 00078          BLBS    STATUS, 3$                                      ;0700
                000000F4        8F      52  D1 0007B 2$:      CMPL    STATUS, #244
                                        05  13 00082          BEQL    3$
                                        52  DD 00084          PUSHL   STATUS                                          ;0701
                        64              01  FB 00086          CALLS   #1, LIB$STOP
                                        7E  7C 00089 3$:      CLRQ    -(SP)                                           ;0703
                              0000G     CF  9F 0008B          PUSHAB  DEVCHAR_DESC
                                        7E  D4 0008F          CLRL    -(SP)
                                        6A  DD 00091          PUSHL   CHANNEL
                00000000G       00      05  FB 00093          CALLS   #5, SYS$GETCHN
                        50 00000000G    9F  D0 0009A          MOVL    @#CTL$GL_PHD, PRIVILEGE_MASK                    ;0710
                43              60      15  E0 000A1          BBS     #21, (PRIVILEGE_MASK), 7$                       ;0711
                                        7E  D4 000A5          CLRL    -(SP)                                           ;0714
                                  04    A5  9F 000A7          PUSHAB  P.AAL
                          0000G CF      02  FB 000AA          CALLS   #2, READ_HOMEBLOCK
                                52      50  D0 000AF          MOVL    R0, STATUS
                                09      52  E8 000B2          BLBS    STATUS, 4$                                      ;0716
                00000010        8F      52  D1 000B5          CMPL    STATUS, #268                                    ;0717
                                        2A  12 000BC          BNEQ    7$
                        02      F9CD    C8  91 000BE 4$:      CMPB    HOME_BLOCK+13, #2                               ;0720
                                        07  12 000C3          BNEQ    5$
                        50      F9EC    C8  D0 000C5          MOVL    HOME_BLOCK+44, VOLUME_OWNER                     ;0721
                                        0C  11 000CA          BRB     6$
                        50      F9DE    C8  9A 000CC 5$:      MOVZBL  HOME_BLOCK+30, VOLUME_OWNER                     ;0724
        50              08      10      F9DF    C8  F0 000D1  INSV    HOME_BLOCK+31, #16, #8, VOLUME_OWNER            ;0725
                                        50  D5 000D8 6$:      TSTL    VOLUME_OWNER                                    ;0728
                                        0C  13 000DA          BEQL    7$
                          0000G CF      50  D1 000DC          CMPL    VOLUME_OWNER, PROCESS_UIC                       ;0729
                                        05  13 000E1          BEQL    7$
                                        24  DD 000E3          PUSHL   #36                                             ;0730
                        64              01  FB 000E5          CALLS   #1, LIB$STOP
                        03              63  E8 000E8 7$:      BLBS    INIT_OPTIONS, 8$                                ;0739
                                    00A3    31 000EB          BRW     18$
                        0B      95      A6  91 000EE 8$:      CMPB    DEVICE_CHAR+5, #11                              ;0741
                                        09  13 000F2          BEQL    9$
                            00758034    8F  DD 000F4          PUSHL   #7700532
                        64              01  FB 000FA          CALLS   #1, LIB$STOP
        1D      04      A3              04  E1 000FD 9$:      BBC     #4, INIT_OPTIONS+4, 10$                         ;0743
                                        7E  7C 00102          CLRQ    -(SP)                                           ;0748
                                        7E  7C 00104          CLRQ    -(SP)
                        7E              01  7D 00106          MOVQ    #1, -(SP)
                                        7E  7C 00109          CLRQ    -(SP)
                                  20    AE  9F 0010B          PUSHAB  IO_STATUS
                                        1E  DD 0010E          PUSHL   #30
                                        6A  DD 00110          PUSHL   CHANNEL
                                        7E  D4 00112          CLRL    -(SP)
                        67              0C  FB 00114          CALLS   #12, SYS$QIOW
                        52              50  D0 00117          MOVL    R0, STATUS
                        22              52  E8 0011A          BLBS    STATUS, 11$                                     ;0749
                                        26  11 0011D          BRB     12$                                             ;0750
        25      04      A3              05  E1 0011F 10$:      BBC     #5, INIT_OPTIONS+4, 13$                         ;0752
                                        7E  7C 00124          CLRQ    -(SP)                                           ;0757
                                        7E  7C 00126          CLRQ    -(SP)
                        7E              02  7D 00128          MOVQ    #2, -(SP)
                                        7E  7C 0012B          CLRQ    -(SP)
```

```
                    20  AE  9F 0012D          PUSHAB  IO_STATUS                                        ;
                        1E  DD 00130          PUSHL   #30                                             ;
                        6A  DD 00132          PUSHL   CHANNEL                                         ;
                        7E  D4 00134          CLRL    -(SP)                                           ;
                67      0C  FB 00136          CALLS   #12, SYS$QIOW                                   ;
                52      50  D0 00139          MOVL    R0, STATUS                                      ;
                06      52  E9 0013C          BLBC    STATUS, 12$                                     ; 0758
                52      6E  3C 0013F 11$:     MOVZWL  IO_STATUS, STATUS                               ;
                0D      52  E8 00142          BLBS    STATUS, 15$                                     ; 0759
                        52  DD 00145 12$:     PUSHL   STATUS                                          ;
                        06  11 00147          BRB     14$                                             ;
            00758014    8F  DD 00149 13$:     PUSHL   #7700500                                        ; 0761
                64      01  FB 0014F 14$:     CALLS   #1, LIB$STOP                                    ;
                        7E  7C 00152 15$:     CLRQ    -(SP)                                           ; 0765
                        7E  7C 00154          CLRQ    -(SP)                                           ;
                        7E  7C 00156          CLRQ    -(SP)                                           ;
                        7E  7C 00158          CLRQ    -(SP)                                           ;
                    20  AE  9F 0015A          PUSHAB  IO_STATUS                                       ;
                        08  DD 0015D          PUSHL   #8                                              ;
                        6A  DD 0015F          PUSHL   CHANNEL                                         ;
                        7E  D4 00161          CLRL    -(SP)                                           ;
                67      0C  FB 00163          CALLS   #12, SYS$QIOW                                   ;
                52      50  D0 00166          MOVL    R0, STATUS                                      ;
                06      52  E9 00169          BLBC    STATUS, 16$                                     ; 0766
                52      6E  3C 0016C          MOVZWL  IO_STATUS, STATUS                               ;
                0E      52  E8 0016F          BLBS    STATUS, 17$                                     ; 0767
    000000F4    8F      52  D1 00172 16$:     CMPL    STATUS, #244                                    ;
                        05  13 00179          BEQL    17$                                             ;
                        52  DD 0017B          PUSHL   STATUS                                          ; 0768
                64      01  FB 0017D          CALLS   #1, LIB$STOP                                    ;
                        7E  7C 00180 17$:     CLRQ    -(SP)                                           ; 0770
                  0000G CF  9F 00182          PUSHAB  DEVCHAR_DESC                                    ;
                        7E  D4 00186          CLRL    -(SP)                                           ;
                        6A  DD 00188          PUSHL   CHANNEL                                         ;
    00000000G   00      05  FB 0018A          CALLS   #5, SYS$GETCHN                                  ;
                    03  A3  95 00191 18$:     TSTB    INIT_OPTIONS+3                                  ; 0773
                        28  18 00194          BGEQ    20$                                             ;
        9180    8F          63  B3 00196      BITW    INIT_OPTIONS, #37248                            ; 0777
                        0F  13 0019B          BEQL    19$                                             ;
                0C      05  A3  93 0019D      BITB    INIT_OPTIONS+5, #12                             ; 0778
                        09  13 001A1          BEQL    19$                                             ;
            007580EC    8F  DD 001A3          PUSHL   #7700716                                        ; 0779
                64          01  FB 001A9      CALLS   #1, LIB$STOP                                    ;
    000FF000    8F          66  D1 001AC 19$:  CMPL    DEVICE_CHAR+112, #1044480                       ; 0781
                        09  1B 001B3          BLEQU   20$                                             ;
            007580DC    8F  DD 001B5          PUSHL   #7700700                                        ; 0782
                64          01  FB 001BB      CALLS   #1, LIB$STOP                                    ;
  2B        01  A3          02  E0 001BE 20$:  BBS     #2, INIT_OPTIONS+1, 24$                         ; 0785
  08            63          04  E1 001C3      BBC     #4, INIT_OPTIONS, 21$                           ; 0788
        0000G   CF      E0  A5  D0 001C7      MOVL    DEF_SYS_PROT, PROTECTION                        ; 0789
                        12  11 001CD          BRB     23$                                             ;
  08            63          03  E1 001CF 21$:  BBC     #3, INIT_OPTIONS, 22$                           ; 0790
        0000G   CF      DC  A5  D0 001D3      MOVL    DEF_GRP_PROT, PROTECTION                        ; 0791
                        06  11 001D9          BRB     23$                                             ;
        0000G   CF      D8  A5  D0 001DB 22$:  MOVL    DEF_PRV_PROT, PROTECTION                        ; 0792
  09            63          02  E1 001E1 23$:  BBC     #2, INIT_OPTIONS  24$                           ; 0793
                50      E4  A5  D2 001E5      MCOML   DEF_SHR_PROT, R0                                ; 0794
```

```
                0000G   CF              50  CA 001E9          BICL2   R0, PROTECTION                                  ; 0797
          13      01    A3              03  E0 001EE 24$:     BBS     #3, INIT_OPTIONS+1, 26$                         ; 0799
                              03        A3  95 001F3          TSTB    INIT_OPTIONS+3                                  ;
                                        08  18 001F6          BGEQ    25$                                             ; 0800
                0000G   CF        EC    A5  D0 001F8          MOVL    DEF_FIL_PROT1, FILE_PROT                        ;
                                        06  11 001FE          BRB     26$                                             ; 0801
                0000G   CF        E8    A5  D0 00200 25$:     MOVL    DEF_FIL_PROT, FILE_PROT                         ; 0803
          06      02    A3              01  E0 00206 26$:     BBS     #1, INIT_OPTIONS+2, 27$                         ; 0804
                0000G   CF        F0    A5  D0 0020B          MOVL    DEF_EXTEND, EXTENSION                           ; 0806
          06      02    A3              02  E0 00211 27$:     BBS     #2, INIT_OPTIONS+2, 28$                         ; 0807
                0000G   CF        F4    A5  D0 00216          MOVL    DEF_WINDOW, WINDOW                              ; 0809
          06      02    A3              03  E0 0021C 28$:     BBS     #3, INIT_OPTIONS+2, 29$                         ; 0810
                0000G   CF        F8    A5  D0 00221          MOVL    DEF_ACCESSED, ACCESSED                          ; 0812
                        04        02    A3  E8 00227 29$:     BLBS    INIT_OPTIONS+2, 30$                             ; 0813
                        6B        FC    A5  D0 0022B          MOVL    DEF_HEADERS, HEADERS                            ; 0815
          1C      01    A3              05  E0 0022F 30$:     BBS     #5, INIT_OPTIONS+1, 32$                         ; 0818
                0000G   CF      0000G   CF  D0 00234          MOVL    PROCESS_UIC, OWNER_UIC                          ; 0819
          04            63              03  E1 0023B          BBC     #3, INIT_OPTIONS, 31$                           ; 0820
                                0000G   CF  B4 0023F          CLRW    OWNER_UIC                                       ; 0821
          09            63              04  E1 00243 31$:     BBC     #4, INIT_OPTIONS, 32$                           ; 0822
                0000G   CF  00010001    8F  D0 00247          MOVL    #65537, OWNER_UIC                               ; 0825
                                  03    A3  95 00250 32$:     TSTB    INIT_OPTIONS+3                                  ;
                                        0E  19 00253          BLSS    33$                                             ; 0829
                                  01    A3  95 00255          TSTB    INIT_OPTIONS+1                                  ;
                                        15  19 00258          BLSS    35$                                             ; 0831
             0000C350   8F                 66  D1 0025A          CMPL    DEVICE_CHAR+112, #50000                         ;
                                        07  1A 00261          BGTRU   34$                                             ; 0832
                0000G   CF              01  D0 00263 33$:     MOVL    #1, CLUSTER                                     ;
                                        05  11 00268          BRB     35$                                             ; 0833
                0000G   CF              03  D0 0026A 34$:     MOVL    #3, CLUSTER                                     ; 0836
          0F      01    A3              06  E0 0026F 35$:     BBS     #6, INIT_OPTIONS+1, 36$                         ; 0837
                        50      0000G   CF  D0 00274          MOVL    CLUSTER, R0                                     ;
                        50              02  C4 00279          MULL2   #2, R0                                          ;
                        50              02  C0 0027C          ADDL2   #2, R0                                          ;
          69            66              50  C7 0027F          DIVL3   R0, DEVICE_CHAR+112, MAXIMUM                    ;
          0D      04    A3              02  E0 00283 36$:     BBS     #2, INIT_OPTIONS+4, 38$                         ; 0839
                        65              66  D1 00288          CMPL    DEVICE_CHAR+112, SMALL_DISK                     ; 0840
                                        04  1B 0028B          BLEQU   37$                                             ;
                        04      91      A6  E9 0028D          BLBC    DEVICE_CHAR+1, 38$                              ; 0841
                        63      40      8F  8A 00291 37$:     BICB2   #64, INIT_OPTIONS                               ; 0842
                        50      0000G   CF  D0 00295 38$:     MOVL    CLUSTER, R0                                     ; 0849
          51            66              50  C1 0029A          ADDL3   R0, DEVICE_CHAR+112, R1                         ;
                                        51  D7 0029E          DECL    R1                                              ;
                        51              50  C6 002A0          DIVL2   R0, R1                                          ;
          68            51              50  C5 002A3          MULL3   R0, R1, VOLUME_SIZE                             ;
          50            68              50  C7 002A7          DIVL3   R0, VOLUME_SIZE, R0                             ; 0856
             000FF000   8F                 50  D1 002AB          CMPL    R0, #1044480                                    ;
                                        05  1A 002B2          BGTRU   39$                                             ;
                        32              50  D1 002B4          CMPL    R0, #50                                         ; 0857
                                        09  18 002B7          BGEQ    40$                                             ;
                            0075809C    8F  DD 002B9 39$:     PUSHL   #7700636                                        ; 0858
                        64              01  FB 002BF          CALLS   #1, LIB$STOP                                    ;
          50      0000G   CF              01  C1 002C2 40$:     ADDL3   #1, CLUSTER, R0                                 ; 0863
          50            68              50  C7 002C8          DIVL3   R0, VOLUME_SIZE, C                              ;
                        50              69  D1 002CC          CMPL    MAXIMUM, C                                      ; 0864
                                        03  15 002CF          BLEQ    41$                                             ;
                        69              50  D0 002D1          MOVL    C, MAXIMUM                                      ; 0865
```

```
                                10              6B  D1 002D4 41$:    CMPL    HEADERS, #16                            ; 0870
                                                03  1E 002D7        BGEQU   42$
                                6B              10  D0 002D9        MOVL    #16, HEADERS
                                        03      A3  95 002DC 42$:    TSTB    INIT_OPTIONS+3                          ; 0871
                                                0E  18 002DF        BGEQ    43$
                            0000FFDC    8F      69  D1 002E1        CMPL    MAXIMUM, #65500                         ; 0872
                                                05  1B 002E8        BLEQU   43$
                                69      FFDC    8F  3C 002EA        MOVZWL  #65500, MAXIMUM
                                69              6B  D1 002EF 43$:    CMPL    HEADERS, MAXIMUM                        ; 0873
                                                03  1B 002F2        BLEQU   44$
                                6B              69  D0 002F4        MOVL    MAXIMUM, HEADERS
                        1C          02      A3      04  E0 002F7 44$:    BBS     #4, INIT_OPTIONS+2, 47$             ; 0878
                        08          02      A3      06  E1 002FC        BBC     #6, INIT_OPTIONS+2, 45$             ; 0881
                0000G   CF                  66      01  C3 00301        SUBL3   #1, DEVICE_CHAR+112, INDEX          ; 0882
                                                    1B  11 00307        BRB     49$
                        05          02      A3      05  E0 00309 45$:    BBS     #5, INIT_OPTIONS+2, 46$             ; 0884
                                        02      A3  95 0030E        TSTB    INIT_OPTIONS+2
                                                11  19 00311        BLSS    49$
                                65              66  D1 00313 46$:    CMPL    DEVICE_CHAR+112, SMALL_DISK             ; 0887
                                                06  1A 00316        BGTRU   48$
                                        0000G   CF  D4 00318 47$:    CLRL    INDEX                                   ; 0888
                                                06  11 0031C        BRB     49$
                0000G   CF                  66      02  C7 0031E 48$:    DIVL3   #2, DEVICE_CHAR+112, INDEX          ; 0889
                                66      0000G   CF  D1 00324 49$:    CMPL    INDEX, DEVICE_CHAR+112                  ; 0892
                                                09  1B 00329        BLEQU   50$
                                    007580B4    8F  DD 0032B        PUSHL   #7700660                                ; 0893
                                64              01  FB 00331        CALLS   #1, LIB$STOP
                            0000G   CF              00  FB 00334 50$:    CALLS   #0, INIT_BADBLOCKS                  ; 0898
                    7E              66              01  C3 00339        SUBL3   #1, DEVICE_CHAR+112, -(SP)          ; 0899
                            0000V   CF              01  FB 0033D        CALLS   #1, ERASE_DISK
                            0000G   CF              00  FB 00342        CALLS   #0, INIT_ALLOCATE                   ; 0900
                            0000G   CF              00  FB 00347        CALLS   #0, INIT_BITMAP                     ; 0901
                                            03      A3  95 0034C        TSTB    INIT_OPTIONS+3                      ; 0902
                                                    07  18 0034F        BGEQ    51$
                            0000G   CF              00  FB 00351        CALLS   #0, INIT_INDEX1                     ; 0903
                                                    05  11 00356        BRB     52$
                            0000G   CF              00  FB 00358 51$:    CALLS   #0, INIT_INDEX                      ; 0904
                            0000G   CF              00  FB 0035D 52$:    CALLS   #0, INIT_MFD                        ; 0905
                            0000V   CF              00  FB 00362        CALLS   #0, VOL_AVAIL                       ; 0910
                                            0000'   CF  9F 00367        PUSHAB  EXIT_BLOCK                          ; 0911
                        00000000G   00              01  FB 0036B        CALLS   #1, SYS$CANEXH
                                    52              50  D0 00372        MOVL    R0, STATUS
                                    05              52  E8 00375        BLBS    STATUS, 53$                         ; 0912
                                                    52  DD 00378        PUSHL   STATUS
                                    64              01  FB 0037A        CALLS   #1, LIB$STOP
                                                    7E  D4 0037D 53$:    CLRL    -(SP)                               ; 0913
                                                    5E  DD 0037F        PUSHL   SP
                                            0000G   CF  9F 00381        PUSHAB  CLEAR_VALID
                        00000000G   9F              03  FB 00385        CALLS   #3, @#SYS$CMKRNL
                                                    04  0038C        RET                                             ; 0915

; Routine Size:  909 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  498          0916  1 ROUTINE ERASE_DISK (LAST_LBN) : NOVALUE =
;  499          0917  1
;  500          0918  1 !++
;  501          0919  1 !
;  502          0920  1 ! FUNCTIONAL DESCRIPTION:
;  503          0921  1 !
;  504          0922  1 !       This routine contains the logic specific to performing a Data
;  505          0923  1 !       Security Erase (DSE) to an ODS-2 disk.  The erase is conditional,
;  506          0924  1 !       and must be performed after the volume's bad blocks have been
;  507          0925  1 !       processed.
;  508          0926  1 !
;  509          0927  1 ! CALLING SEQUENCE:
;  510          0928  1 !       ERASE_DISK (ARG1)
;  511          0929  1 !
;  512          0930  1 ! INPUT PARAMETERS:
;  513          0931  1 !       ARG1:   Last LBN on this disk.
;  514          0932  1 !
;  515          0933  1 ! IMPLICIT INPUTS:
;  516          0934  1 !       parser data base
;  517          0935  1 !       own storage of this module
;  518          0936  1 !
;  519          0937  1 ! OUTPUT PARAMETERS:
;  520          0938  1 !       NONE
;  521          0939  1 !
;  522          0940  1 ! IMPLICIT OUTPUTS:
;  523          0941  1 !       NONE
;  524          0942  1 !
;  525          0943  1 ! ROUTINE VALUE:
;  526          0944  1 !       NONE
;  527          0945  1 !
;  528          0946  1 ! SIDE EFFECTS:
;  529          0947  1 !       disk volume is erased.
;  530          0948  1 !
;  531          0949  1 !--
;  532          0950  1
;  533          0951  2 BEGIN
;  534          0952  2
;  535          0953  2 EXTERNAL
;  536          0954  2         CLUSTER,                                ! Volume cluster factor
;  537          0955  2         CHANNEL,                                ! I/O channel to device
;  538          0956  2         INIT_OPTIONS    : BITVECTOR;            ! Command options
;  539          0957  2
;  540          0958  2 EXTERNAL ROUTINE
;  541          0959  2         ERASE_BLOCKS;                           ! Common disk DSE routine
;  542          0960  2
;  543          0961  2 LOCAL
;  544          0962  2         COUNT,                                  ! # of blocks to erase
;  545          0963  2         ERASE_STATUS,                           ! Status of the DSE
;  546          0964  2         I,                                      ! Table index
;  547          0965  2         LBN,                                    ! Current LogicaL Block Number
;  548          0966  2         NEXT_LBN,                               ! Next Logical Block Number
;  549          0967  2         STATUS;                                 ! Temporary storage
;  550          0968  2
;  551          0969  2 !
;  552          0970  2 ! If the user did not explicitly request a DSE on an ODS-2 disk, return.
;  553          0971  2 !
;  554          0972  2 IF  NOT .INIT_OPTIONS[OPT_ERASE]
```

```
;  555     0973  2 OR  .INIT_OPTIONS[OPT_STRUCTURE1]
;  556     0974  2 THEN
;  557     0975  2     RETURN;
;  558     0976  2
;  559     0977  2 !
;  560     0978  2 ! Perform a DSE on the disk.  Use a common routine that does the DSE
;  561     0979  2 ! via logical I/O.  The bad block data has been processed, so take care
;  562     0980  2 ! not to overwrite any of the data, especially the factory bad block file.
;  563     0981  2 ! The bad block allocation information is stored in descending order on
;  564     0982  2 ! the LBN, and the count is in clusters.  The volume blocking factor has
;  565     0983  2 ! been accounted for.
;  566     0984  2 !
;  567     0985  2
;  568     0986  2 ERASE_STATUS = 1;
;  569     0987  2 LBN = 0;
;  570     0988  2 I = .BADBLOCK_TOTAL - 1;
;  571     0989  2
;  572     0990  2 WHILE .LBN LSS .LAST_LBN DO
;  573     0991  3     BEGIN
;  574     0992  3     !
;  575     0993  3     ! Determine the area to be erased.  Do not erase those portions of
;  576     0994  3     ! the volume that are allocated to the bad block file.
;  577     0995  3     !
;  578     0996  4     IF (.I GEQ 0) AND (.BADBLOCK_CNT[.I] NEQ 0)
;  579     0997  3     THEN
;  580     0998  4         BEGIN
;  581     0999  4         !
;  582     1000  4         ! Erase from the current position to the start of the allocated extent.
;  583     1001  4         !
;  584     1002  4         COUNT = .BADBLOCK_LBN[.I] - .LBN;
;  585     1003  4         NEXT_LBN = .BADBLOCK_LBN[.I] + .BADBLOCK_CNT[.I];
;  586     1004  4         I = .I - 1;
;  587     1005  4         END
;  588     1006  3     ELSE
;  589     1007  4         BEGIN
;  590     1008  4         !
;  591     1009  4         ! Erase from the current position to the end of the volume.
;  592     1010  4         !
;  593     1011  4         COUNT = .LAST_LBN - .LBN;
;  594     1012  4         NEXT_LBN = .LAST_LBN;
;  595     1013  3         END;
;  596     1014  3
;  597     1015  3     !
;  598     1016  3     ! Do the DSE.  If errors are encountered, only the first error is reported.
;  599     1017  3     !
;  600     1018  4     IF NOT (STATUS = EXEC_CALL (ERASE_BLOCKS, .LBN, .COUNT, .CHANNEL))
;  601     1019  3     THEN
;  602     1020  3         IF .ERASE_STATUS
;  603     1021  3         THEN
;  604     1022  3             ERASE_STATUS = .STATUS;
;  605     1023  3
;  606     1024  3     !
;  607     1025  3     ! Advance to the next range of logical blocks.
;  608     1026  3     !
;  609     1027  3     LBN = .NEXT_LBN;
;  610     1028  2     END;                                    ! End of WHILE loop
;  611     1029  2
```

```
;  612          1030  2 !
;  613          1031  2 ! If an error was encountered, let the user know about it.
;  614          1032  2 !
;  615          1033  2 IF NOT .ERASE_STATUS
;  616          1034  2 THEN
;  617          1035  2     ERR_MESSAGE (INIT$_ERASEFAIL, 0, .ERASE_STATUS);
;  618          1036  2
;  619          1037  1 END;


                                                .EXTRN  ERASE_BLOCKS, SYS$CMEXEC

                                007C 00000 ERASE_DISK:
                                                .WORD   Save R2,R3,R4,R5,R6                                     ; 0916
                79      0000G   CF              02  E1 00002            BBC     #2, INIT_OPTIONS+5, 6$          ; 0972
                                        0000G   CF  95 00008            TSTB    INIT_OPTIONS+3                  ; 0973
                                                73  19 0000C            BLSS    6$                              ;
                                    54          01  D0 0000E            MOVL    #1, ERASE_STATUS                ; 0986
                                                53  D4 00011            CLRL    LBN                             ; 0987
                52      0000'   CF              01  C3 00013            SUBL3   #1, BADBLOCK_TOTAL, I           ; 0988
                          04    AC              53  D1 00019 1$:        CMPL    LBN, LAST_LBN                   ; 0990
                                                4E  18 0001D            BGEQ    5$                              ;
                                                52  D5 0001F            TSTL    I                               ; 0996
                                                1C  19 00021            BLSS    2$                              ;
                                        0000'CF42   D5 00023            TSTL    BADBLOCK_CNT[I]                 ;
                                                15  13 00028            BEQL    2$                              ;
                56      0000'CF42               53  C3 0002A            SUBL3   LBN, BADBLOCK_LBN[I], COUNT     ; 1002
                55      0000'CF42       0000'CF42   C1 00031            ADDL3   BADBLOCK_CNT[I], BADBLOCK_LBN[I], NEXT_LBN ; 1003
                                                52  D7 0003B            DECL    I                               ; 1004
                                                09  11 0003D            BRB     3$                              ; 0996
                56          04  AC              53  C3 0003F 2$:        SUBL3   LBN, LAST_LBN, COUNT            ; 1011
                                    55      04  AC  D0 00044            MOVL    LAST_LBN, NEXT_LBN              ; 1012
                                        0000G   CF  DD 00048 3$:        PUSHL   CHANNEL                         ; 1018
                                        0048    8F  BB 0004C            PUSHR   #^M<R3,R6>                      ;
                                                03  DD 00050            PUSHL   #3                              ;
                                                5E  DD 00052            PUSHL   SP                              ;
                                        0000G   CF  9F 00054            PUSHAB  ERASE_BLOCKS                    ;
                        00000000G       9F          06  FB 00058        CALLS   #6, @#SYS$CMEXEC                ;
                                    06              50  E8 0005F        BLBS    STATUS, 4$                      ;
                                    03              54  E9 00062        BLBC    ERASE_STATUS, 4$                ; 1020
                                    54              50  D0 00065        MOVL    STATUS, ERASE_STATUS            ; 1022
                                    53              55  D0 00068 4$:    MOVL    NEXT_LBN, LBN                   ; 1027
                                                    AC  11 0006B        BRB     1$                              ; 0990
                                    11              54  E8 0006D 5$:    BLBS    ERASE_STATUS, 6$                ; 1033
                                                    54  DD 00070        PUSHL   ERASE_STATUS                    ; 1035
                                                    7E  D4 00072        CLRL    -(SP)                           ;
                                        00759010    8F  DD 00074        PUSHL   #7704592                        ;
                        00000000G       00          03  FB 0007A        CALLS   #3, LIB$SIGNAL                  ;
                                                        04 00081 6$:    RET                                     ; 1037

; Routine Size:  130 bytes,    Routine Base:  $CODE$ + 038D
```

```
;  621          1038  1 ROUTINE VOL_AVAIL : NOVALUE =
;  622          1039  1
;  623          1040  1 !++
;  624          1041  1 !
;  625          1042  1 ! FUNCTIONAL DESCRIPTION:
;  626          1043  1 !
;  627          1044  1 !       This routine issues an IO$_AVAILABLE function to undo the effects of
;  628          1045  1 !       the IO$_PACKACK function.
;  629          1046  1 !
;  630          1047  1 !
;  631          1048  1 ! CALLING SEQUENCE:
;  632          1049  1 !       Implicitly called during image rundown, or called via VOL_AVAIL ().
;  633          1050  1 !
;  634          1051  1 ! INPUT PARAMETERS:
;  635          1052  1 !       NONE
;  636          1053  1 !
;  637          1054  1 ! IMPLICIT INPUTS:
;  638          1055  1 !       CHANNEL - channel for the disk
;  639          1056  1 !
;  640          1057  1 ! OUTPUT PARAMETERS:
;  641          1058  1 !       NONE
;  642          1059  1 !
;  643          1060  1 ! IMPLICIT OUTPUTS:
;  644          1061  1 !       NONE
;  645          1062  1 !
;  646          1063  1 ! ROUTINE VALUE:
;  647          1064  1 !       NONE
;  648          1065  1 !
;  649          1066  1 ! SIDE EFFECTS:
;  650          1067  1 !       disk volume released
;  651          1068  1 !
;  652          1069  1 !--
;  653          1070  1
;  654          1071  2 BEGIN
;  655          1072  2
;  656          1073  2 EXTERNAL
;  657          1074  2         CHANNEL;                      ! channel assigned to device
;  658          1075  2
;  659        P 1076  2 $QIOW (CHAN = .CHANNEL,               ! issue the function, ignore status
;  660          1077  2        FUNC = IO$_AVAILABLE);
;  661          1078  2
;  662          1079  1 END;                                  ! end of routine VOL_AVAIL


                                0000 00000 VOL_AVAIL:
                                           .WORD   Save nothing                    ; 1038
                                7E  7C 00002       CLRQ    -(SP)                ; 1077
                                7E  7C 00004       CLRQ    -(SP)                ;
                                7E  7C 00006       CLRQ    -(SP)                ;
                                7E  7C 00008       CLRQ    -(SP)                ;
                          7E    11  7D 0000A       MOVQ    #17, -(SP)           ;
                      0000G     CF  DD 0000D       PUSHL   CHANNEL              ;
                                7E  D4 00011       CLRL    -(SP)                ;
                00000000G   00    0C  FB 00013       CALLS   #12, SYS$QIOW        ;
```

```
                                04 0001A        RET                                     ; 1079

; Routine Size:  27 bytes,    Routine Base:  $CODE$ + 040F
```

```
;  664          1080  1 END
;  665          1081  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL, LIB$STOP

;                               PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $GLOBAL$                       1628  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                           52  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $OWN$                            20  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                         1066  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  .  ABS  .                         0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)


;                             Library Statistics
;
;                                    -------- Symbols --------      Pages       Processing
;       File                         Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1      18619        25         0      1000          00:01.9


;                                       COMMAND QUALIFIERS

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

; Size:          1066 code + 1700 data bytes
; Run Time:         00:27.3
; Elapsed Time:     00:55.4
; Lines/CPU Min:     2374
; Lexemes/CPU-Min: 29732
; Memory Used:  299 pages
; Compilation Complete
```