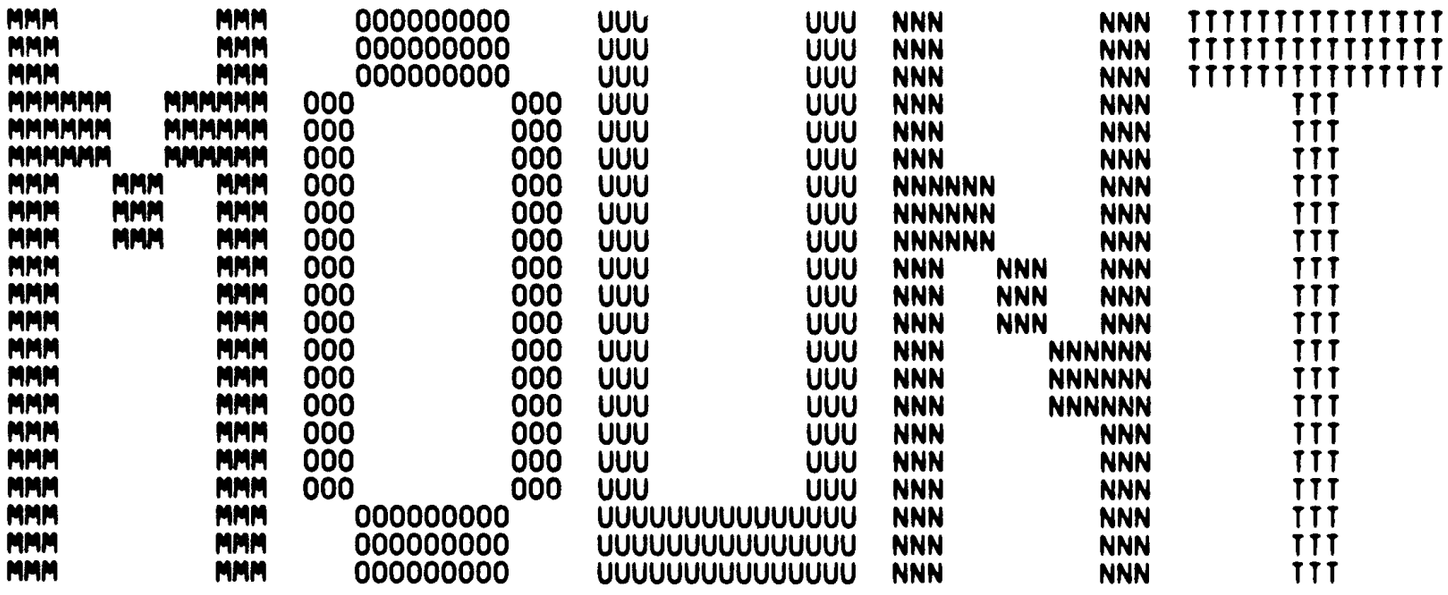
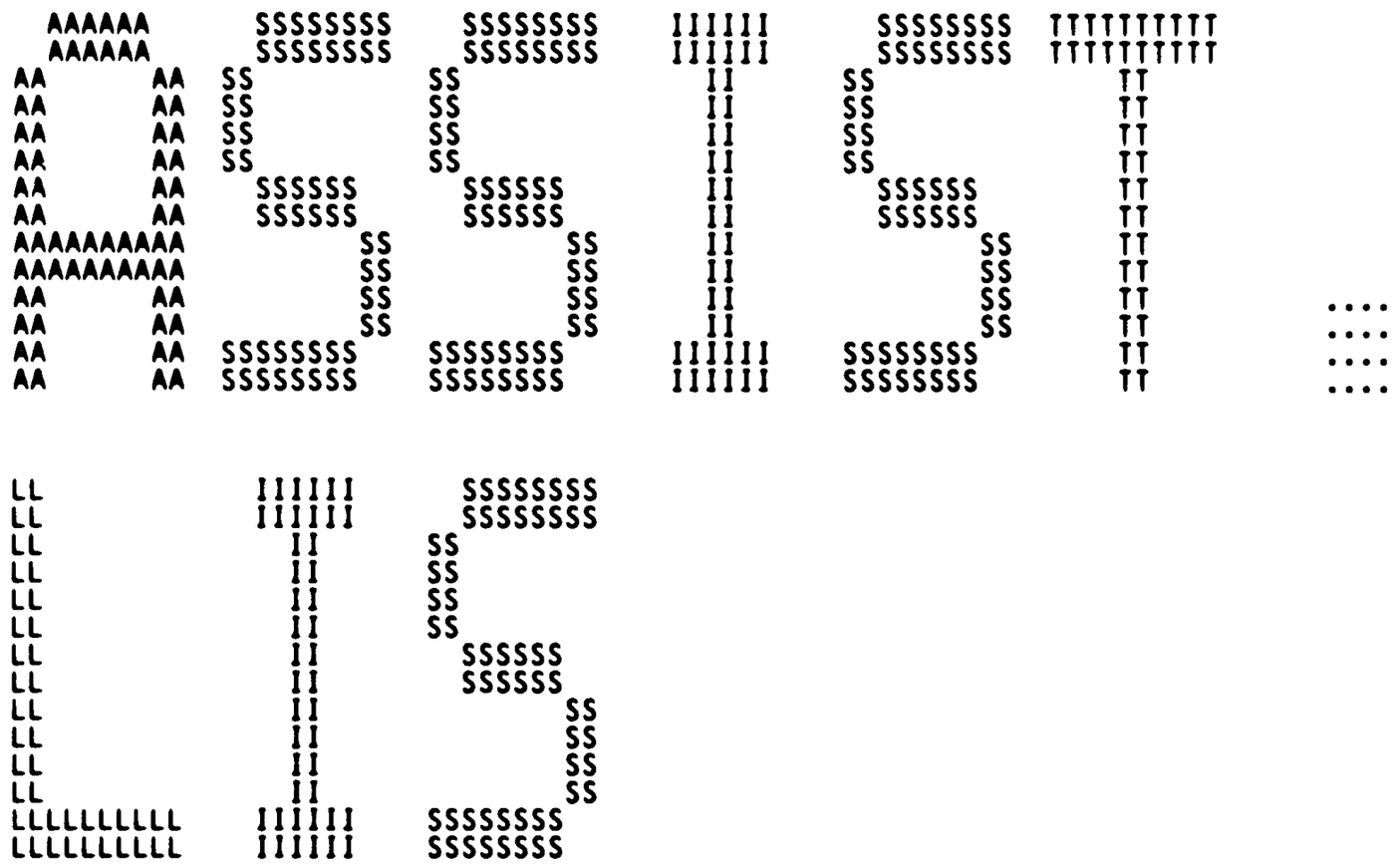

MMM MMM 000000000 UUU UUU NNN NNN TTTTTTTTTTTTTTT
MMM MMM 000000000 UUU UUU NNN NNN TTTTTTTTTTTTTTT
MMM MMM 000000000 UUU UUU NNN NNN TTTTTTTTTTTTTTT
MMMMMMM MMMMMMM 000 000 UUU UUU NNN NNN TTT
MMMMMMM MMMMMMM 000 000 UUU UUU NNN NNN TTT
MMMMMMM MMMMMMM 000 000 UUU UUU NNN NNN TTT
MMM MMM MMM 000 000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

FILEID**ASSIST



```
;     1          0001  0 MODULE ASSIST   (
;     2          0002  0                  LANGUAGE (BLISS32),
;     3          0003  0                  IDENT = 'V04-001'
;     4          0004  0                 ) =
;     5          0005  0
;     6          0006  1 BEGIN
;     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.
;    34          0034  1 !
;    35          0035  1 !       MOUNT
;    36          0036  1 !
;    37          0037  1 ! ABSTRACT:
;    38          0038  1 !
;    39          0039  1 !       This module contains the routines to
;    40          0040  1 !       implement operator assisted mount.
;    41          0041  1 !
;    42          0042  1 ! ENVIRONMENT:
;    43          0043  1 !
;    44          0044  1 !       VAX/VMS operating system.
;    45          0045  1 !
;    46          0046  1 ! AUTHOR:
;    47          0047  1 !
;    48          0048  1 !       Steven T. Jeffreys
;    49          0049  1 !
;                0050  1 ! CREATION DATE:
;    51          0051  1 !
;    52          0052  1 !       October 9, 1980
;    53          0053  1 !
;    54          0054  1 ! MODIFIED BY:
;    55          0055  1 !
;    56          0056  1 !       V04-001 HH0056          Hai Huang               11-Sep-1984
;    57          0057  1 !               Do limited number of retries on VOLINV error.
```

```
;    58          0058  1 !
;    59          0059  1 !       V03-007 HH0041          Hai Huang                       24-Jul-1984
;    60          0060  1 !               Remove REQUIRE  'LIBD$:[VMSLIB.OBJ]MOUNTMSG.B32'.
;    61          0061  1 !
;    62          0062  1 !       V03-006 CWH3001         CW Hobbs                        30-Jul-1983
;    63          0063  1 !               Various and sundry things to make OPCOM distributed
;    64          0064  1 !               across the cluster.
;    65          0065  1 !
;    66          0066  1 !       V03-005 TCM0002         Trudy C. Matthews       28-Jul-1983
;    67          0067  1 !               Add DEV_ACQUIRED flag that indicates whether mount interlock
;    68          0068  1 !               has been taken out for this device.  Remove DEALLOCATE_DEVICE
;    69          0069  1 !               routine, since devices mounted /SHARE, /SYSTEM or /GROUP are
;    70          0070  1 !               no longer allocated.  Remove temporary change introduced in
;    71          0071  1 !               TCM0001.
;    72          0072  1 !
;    73          0073  1 !       V03-004 TCM0001         Trudy C. Matthews       18-Jul-1983
;    74          0074  1 !               Make SS$_NOTQUEUED status (received from the $ENQ system
;    75          0075  1 !               service when we cannot take out a cluster-wide allocation
;    76          0076  1 !               lock on this device) one of the status codes acted on by
;    77          0077  1 !               operator-assisted mount.
;    78          0078  1 !
;    79          0079  1 !       V03-003 STJ50311        Steven T. Jeffreys      10-Feb-1983
;    80          0080  1 !               - Make all uses of PHYS_NAME indexed by DEVICE_INDEX.
;    81          0081  1 !               - Reset PREVIOUS_STATUS after an operator reply arrives.
;    82          0082  1 !               - If the mount failed with an operator request outstanding,
;    83          0083  1 !                 signal MOUN$_OPRQSTCAN instead of MOUN$_RQSTDON.
;    84          0084  1 !               - Define and use routine $DALLOC_DEVS.
;    85          0085  1 !
;    86          0086  1 !       V03-002 STJ0244         Steven T. Jeffreys      04-Apr-1982
;    87          0087  1 !               - Use common I/O routines, and make the code more
;    88          0088  1 !                 tolerant to random event flag setting and clearing.
;    89          0089  1 !               - Issue the MOUN$_RQSTDON status if the mount completes
;    90          0090  1 !                 successfully while we have an operator request outstanding.
;    91          0091  1 !
;    92          0092  1 !       V03-001 BLS0160         Benn Schreiber          18-Mar-1982
;    93          0093  1 !               Get OPCDEFTMP from SHRLIB$.
;    94          0094  1 !
;    95          0095  1 !       V02-011 STJ0229         Steven T. Jeffreys      01-Mar-1982
;    96          0096  1 !               - Set the inhibit message bit in the exit status
;    97          0097  1 !                 code if the message output via $PUTMSG.
;    98          0098  1 !
;    99          0099  1 !       V02-010 STJ0218         Steven T. Jeffreys      16-Feb-1982
;   100          0100  1 !               - Cancel exit handler before declaring it.
;   101          0101  1 !               - Clear system service failure exception mode and
;   102          0102  1 !                 restore it on exit.
;   103          0103  1 !
;   104          0104  1 !       V02-009 STJ0214         Steven T. Jeffreys      11-Feb-1982
;   105          0105  1 !               Add support for the /COMMENT switch.
;   106          0106  1 !
;   107          0107  1 !       V02-008 STJ0206         Steven T. Jeffreys      08-Feb-1982
;   108          0108  1 !               Set mailbox access rights to allow SYSTEM and OWNER
;   109          0109  1 !               read and write privileges.
;   110          0110  1 !
;   111          0111  1 !       V02-007 STJ0189         Steven T. Jeffreys      02-Feb-1982
;   112          0112  1 !               Initalize GLOBAL storage at run time, and fix various bugs.
;   113          0113  1 !
;   114          0114  1 !       V02-006 STJ174          Steven T. Jeffreys      19-Jan-1982
```

```
;   115          0115  1 !                 Made most of the GLOBAL routines in to local routines.
;   116          0116  1 !
;   117          0117  1 !       V02-005 STJ162          Steven T. Jeffreys      04-Jan-1982
;   118          0118  1 !               Removed copy of INTERCEPT_SIGNAL.
;   119          0119  1 !
;   120          0120  1 !       V02-004 STJ0150         Steven T. Jeffreys
;   121          0121  1 !               Extensive rewrite to support the $MOUNT system service.
;   122          0122  1 !
;   123          0123  1 !       V02-003 STJ0112         Steven T. Jeffreys
;   124          0124  1 !               - Use general addressing mode for library routines.
;   125          0125  1 !               - Fixed SET_TARGET_MASK.
;   126          0126  1 !               - Fixed SUBMIT_REQUEST to calculate actual message size.
;   127          0127  1 !               - Added support for alternate cancellation message.
;   128          0128  1 !               - Handle REPLY/BLANK_TAPE and REPLY/INITIALIZE_TAPE operator replies.
;   129          0129  1 !
;   130          0130  1 !       V02-002 STJ0083         Steven T. Jeffreys
;   131          0131  1 !               - Changed $DELMBX call in CANCEL_REQUEST to $DASSGN to properly
;   132          0132  1 !                 delete the mailbox and free up the channel.
;   133          0133  1 !               - Changed error recovery handlers to use the physical device
;   134          0134  1 !                  name string when referring to the device.
;   135          0135  1 !               - Added logic to recover from an SS$_INCVOLLABEL error, which
;   136          0136  1 !                 occurs when the label of the volume present in the drive does
;   137          0137  1 !                 not match the volume label specified by the user.
;   138          0138  1 !
;   139          0139  1 !--
;   140          0140  1
;   141          0141  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;   142          0142  1 LIBRARY 'SYS$LIBRARY:TPAMAC';
;   143          0143  1 REQUIRE 'LIBD$:[VMSLIB.OBJ]INITMSG.REQ';
;   144          0275  1 REQUIRE 'SHRLIB$:OPCDEFTMP';    ! *** TEMPORARY
;   145          0516  1 REQUIRE 'SRC$:MOUDEF.B32';
;   146          1048  1
;   147          1049  1 FORWARD ROUTINE
;   148          1050  1
;   149          1051  1         SYS$MOUNT,                      ! Main entry point of $MOUNT
;   150          1052  1         INTERCEPT_SIGNAL,               ! Main condition handler
;   151          1053  1         SUBMIT_REQUEST  : NOVALUE,      ! Send request to operator
;   152          1054  1         SET_TARGET_MASK : NOVALUE,      ! Sets operator target mask
;   153          1055  1         POST_READ_TO_MBX: NOVALUE,      ! Post read to reply mailbox
;   154          1056  1         INTERACTIVE_JOB,                ! Determines if we're a batch job
;   155          1057  1         PRINT_REPLY     : NOVALUE,      ! Print the operator reply
;   156          1058  1         PARSE_REPLY     : NOVALUE,      ! Parse the oprator's reply
;   157          1059  1         CANCEL_REQUEST  : NOVALUE,      ! Cancel the operator request
;   158          1060  1         CHECK_FOR_REPLY : NOVALUE,      ! Check for operator response
;   159          1061  1         ALLOCFAIL_HNDLR : NOVALUE,      ! Handle device allocation failures
;   160          1062  1         MEDOFL_HNDLR    : NOVALUE,      ! Handle SS$_MEDOFL condition
;   161          1063  1         WRONGVOL_HNDLR  : NOVALUE,      ! Handle SS$_INCVOLLABEL condition
;   162          1064  1         INVALID_COMMAND,                ! Notify user/operator of invalid reply
;   163          1065  1         EXIT_HANDLER    : NOVALUE;      ! Exit handler
;   164          1066  1
;   165          1067  1 FORWARD
;   166          1068  1
;   167          1069  1         STATE_TABLE     : VECTOR [0],           ! TPARSE state table
;   168          1070  1         KEY_TABLE       : VECTOR [0];           ! TPARSE key table
;   169          1071  1
;   170          1072  1 STRUCTURE
;   171          1073  1
```

```
;  172          1074  1          EXTERNAL_CTRL_BLK [I ; N] =                     ! exit handler descriptor
;  173          1075  1              [(4+N)*4]                                   ! N = # of arguments ( N <= 1)
;  174          1076  1              (EXIT_CTRL_BLK+I*4)<0,32,0>;                ! the block is a longword array
;  175          1077  1
;  176          1078  1 MACRO
;  177          1079  1          !
;  178          1080  1          ! Abort the mount operation.
;  179          1081  1          !
;  180        M 1082  1          ABORT_MOUNT (CODE) =
;  181        M 1083  1                  SIGNAL_STOP     (%IF NOT %NULL (CODE) %THEN CODE %ELSE 0 %FI
;  182        M 1084  1                                   %IF NOT %NULL (%REMAINING) %THEN , %REMAINING %FI
;  183          1085  1                                  )%;
;  184          1086  1
;  185          1087  1 MACRO
;  186          1088  1          !
;  187          1089  1          ! Generate a static string descriptor
;  188          1090  1          !
;  189        M 1091  1          DESCRIP (STRING) =
;  190        M 1092  1                  BBLOCK  [DSC$K_S_BLN]
;  191        M 1093  1                  INITIAL (WORD (%CHARCOUNT (STRING)),
;  192        M 1094  1                           BYTE (DSC$K_DTYPE_T),
;  193        M 1095  1                           BYTE (DSC$K_CLASS_S),
;  194        M 1096  1                           LONG (UPLIT_BYTE (STRING))
;  195          1097  1                          )%;
;  196          1098  1
;  197          1099  1 MACRO
;  198          1100  1          !
;  199          1101  1          ! 3 byte operator mask field definition.
;  200          1102  1          !
;  201          1103  1          TARGET_FIELD = $BYTEOFFSET(OPC$B_MS_TARGET), 0, 24, 0%;
;  202          1104  1
;  203          1105  1 MACRO
;  204          1106  1          !
;  205          1107  1          ! For documentation purposes, define a boolean variable
;  206          1108  1          ! that can only take on the values TRUE or FALSE.
;  207          1109  1          !
;  208          1110  1          BOOLEAN = LONG%;
;  209          1111  1
;  210          1112  1 LITERAL
;  211          1113  1          FAO_BUFFER_SIZE = 512,          ! Max length of FAO result string
;  212          1114  1          MAX_DEV_LENGTH  = 63,           ! Max length of device name
;  213          1115  1          !
;  214          1116  1          ! Create the reply mailbox protection mask.  Allow only
;  215          1117  1          ! OWNER(read) and SYSTEM(read,write) access.  See documentation
;  216          1118  1          ! of the $CREMBX system service for more info.
;  217          1119  1          !
;  218          1120  1          MAILBOX_PROTECTION = %X'FF00',
;  219          1121  1          !
;  220          1122  1          ! The following are boolean values that are used to make the
;  221          1123  1          ! code more readable.  They are used as input to CANCEL_REQUEST.
;  222          1124  1          !
;  223          1125  1          REQUEST_SATISFIED       = 1,    ! The request completed w/o opertor intervention
;  224          1126  1          REQUEST_NOT_SATISFIED   = 0,    ! The request is being cancled for some reason
;  225          1127  1          !
;  226          1128  1          ! The following are mask definitions used for retrieving specified
;  227          1129  1          ! portions of a message via the $GETMSG system service.
;  228          1130  1          !
```

```
;  229          1131  1          MSG_TEXT        = 1,            ! Include message text
;  230          1132  1          MSG_ID          = 2,            ! Include message identifier
;  231          1133  1          MSG_SEVERITY    = 4,            ! Include severity indicator
;  232          1134  1          MSG_FACILITY    = 8,            ! Include message facility name
;  233          1135  1          !
;  234          1136  1          ! The following are indexes into the Exit Handler Control Block
;  235          1137  1          !
;  236          1138  1          XHNDLR_ADDRESS  = 1,            ! exit handler address
;  237          1139  1          XHNDLR_ARGCNT   = 2,            ! exit handler argument count
;  238          1140  1          XHNDLR_STSADDR  = 3,            ! system exit status address
;  239          1141  1          !
;  240          1142  1          TRUE            = 1,            ! Boolean value
;  241          1143  1          FALSE           = 0,            ! Boolean value
;  242          1144  1          !
;  243          1145  1          WAIT            = 1,            ! Enable wait for reply
;  244          1146  1          NO_WAIT         = 0,            ! Disable wait for reply
;  245          1147  1          !
;  246          1148  1          REPLY_FLAG      = MOUNT_EFN,    ! A local event flag #
;  247          1149  1          TIMER_FLAG      = TIMER_EFN,    ! A local event flag #
;  248          1150  1          TIMER_ID        = 999,          ! Timer identification #
;  249          1151  1          !
;  250          1152  1          EXPECT_REPLY    = 1,            ! Indicates that we expect a reply
;  251          1153  1          NO_REPLY        = 0;            ! Indecates that we don't desire a reply
;  252          1154  1
;  253          1155  1  GLOBAL LITERAL
;  254          1156  1          VOLINV_LIMIT    = 20;           ! VOLINV retry limit
;  255          1157  1
;  256          1158  1  !
;  257          1159  1  ! Define the static storage used by this module.  Note that the
;  258          1160  1  ! virtual pages on which this data resides must be USER writable.
;  259          1161  1  ! It is important that this data start on a page boundary, so that
;  260          1162  1  ! the $SETPRT call does not make pages writable that were not meant
;  261          1163  1  ! to be.
;  262          1164  1  !
;  263          1165  1
;  264          1166  1  PSECT GLOBAL = $USER_DATA$ (WRITE, NOEXECUTE, NOSHARE, ALIGN (9));
;  265          1167  1
;  266          1168  1  GLOBAL
;  267          1169  1          VA_START        : VECTOR [0] ALIGN (9),             ! Start of 'user data'
;  268          1170  1          VOLINV_COUNT    : LONG,             ! VOLINV retry counter
;  269          1171  1          !
;  270          1172  1          ! Declare boolean variables.
;  271          1173  1          !
;  272          1174  1          REPLY_PENDING   : BOOLEAN VOLATILE,     ! Determines if response outstanding
;  273          1175  1          MOUNT_FAILED    : BOOLEAN VOLATILE,     ! Used in conjunction with MOUNT_STATUS
;  274          1176  1          OPERATOR_PRESENT: BOOLEAN VOLATILE,     ! Determines operator presence
;  275          1177  1          RETRY_COUNTER   : LONG VOLATILE,        ! Number of retries
;  276          1178  1          SS_FAIL_MODE    : BOOLEAN,              ! System service failure mode
;  277          1179  1          !
;  278          1180  1          ! Declare condition context variables.
;  279          1181  1          !
;  280          1182  1          MOUNT_STATUS    : BBLOCK[4] VOLATILE,   ! Primary condition
;  281          1183  1          PREVIOUS_STATUS : BBLOCK[4] VOLATILE,   ! Previous primary condition
;  282          1184  1          PREVIOUS_DEV_IDX: LONG VOLATILE,        ! Previous device index #
;  283          1185  1          OPERATOR_MASK   : LONG VOLATILE,        ! Mask of operators to receive requests
;  284          1186  1          REQUEST_ID      : LONG VOLATILE,        ! Operator request #
;  285          1187  1          !
```

```
;  286          1188  1          ! Declare exit handler control block.
;  287          1189  1          !
;  288          1190  1          EXIT_HNDLR_DSC  : EXIT_CTRL_BLK [0],    ! Define exit handler descriptor
;  289          1191  1          !
;  290          1192  1          ! Declare storge related to the operator reply message.
;  291          1193  1          !
;  292          1194  1          REPLY_CHANNEL   : LONG VOLATILE,        ! Channel of reply mailbox
;  293          1195  1          REPLY_IOSB      : BBLOCK [8] VOLATILE,  ! IOSB  for operator reply read
;  294          1196  1          REPLY_BUFFER    : BBLOCK [OPC$S_MS_OTEXT+8] VOLATILE,
;  295          1197  1          REPLY_DESC      : BBLOCK [DSC$K_S_BLN] VOLATILE
;  296          1198  1                            INITIAL (WORD (OPC$S_MS_OTEXT+8),
;  297          1199  1                                     BYTE (DSC$K_DTYPE_T),
;  298          1200  1                                     BYTE (DSC$K_CLASS_S),
;  299          1201  1                                     LONG (REPLY_BUFFER)
;  300          1202  1                                    ),
;  301          1203  1
;  302          1204  1          !
;  303          1205  1          ! Define the TPARSE control block.
;  304          1206  1          !
;  305          1207  1          TPARSE_BLOCK    : BBLOCK [TPA$K_LENGTH0]
;  306          1208  1                            INITIAL (TPA$R_COUNT0,TPA$M_ABBREV),
;  307          1209  1
;  308          1210  1          !
;  309          1211  1          ! Define the device name descriptor that is used as an implicit
;  310          1212  1          ! output to a TPARSE action routine.
;  311          1213  1          !
;  312          1214  1          DEVICE_DESC     : BBLOCK [DSC$K_S_BLN]  ! Descriptor for device name
;  313          1215  1                            INITIAL (WORD (MAX_DEV_LENGTH),
;  314          1216  1                                     BYTE (DSC$K_DTYPE_T),
;  315          1217  1                                     BYTE (DSC$K_CLASS_S),
;  316          1218  1                                     LONG (0)
;  317          1219  1                                    ),
;  318          1220  1          !
;  319          1221  1          ! Declare storage for operator message and its descriptor.
;  320          1222  1          !
;  321          1223  1          OP_MSG_BUF      : BBLOCK [OPC$S_MS_OTEXT] ! Buffer for op. request ms
;  322          1224  1                            INITIAL (BYTE (OPC$_RQ_RQST)),
;  323          1225  1
;  324          1226  1          OP_MSG_DESC     : BBLOCK [DSC$K_S_BLN]  ! Descriptor for op. request
;  325          1227  1                            INITIAL (WORD (OPC$S_MS_OTEXT),
;  326          1228  1                                     BYTE (DSC$K_DTYPE_T),
;  327          1229  1                                     BYTE (DSC$K_CLASS_S),
;  328          1230  1                                     LONG (OP_MSG_BUF)
;  329          1231  1                                    ),
;  330          1232  1
;  331          1233  1          CANCEL_MSG_BUF  : BBLOCK [OPC$K_HDR_SIZE]               ! Cancel message
;  332          1234  1                            INITIAL (BYTE (OPC$_X_CANCEL),       ! Set cancellation code
;  333          1235  1                                     BYTE (OPC$R_UNSPEC)         ! Set SCOPE unspecified
;  334          1236  1                                    ),
;  335          1237  1
;  336          1238  1          CANCEL_MSG_DESC : BBLOCK [DSC$K_S_BLN]  ! Cancel message descriptor
;  337          1239  1                            INITIAL (WORD (OPC$K_HDR_SIZE),
;  338          1240  1                                     BYTE (DSC$K_DTYPE_T),
;  339          1241  1                                     BYTE (DSC$K_CLASS_S),
;  340          1242  1                                     LONG (CANCEL_MSG_BUF)
;  341          1243  1                                    ),
;  342          1244  1          !
```

```
;  343          1245  1          ! Declare storage for FAO resultant string buffer and descriptor.
;  344          1246  1          !
;  345          1247  1          FAO_BUFFER      : BBLOCK [FAO_BUFFER_SIZE],
;  346          1248  1          FAO_RESULT_DESC : BBLOCK [DSC$K_S_BLN]
;  347          1249  1                            INITIAL (WORD (LOG$C_NAMLENGTH),
;  348          1250  1                                     BYTE (DSC$K_DTYPE_T),
;  349          1251  1                                     BYTE (DSC$K_CLASS_S),
;  350          1252  1                                     LONG (FAO_BUFFER)
;  351          1253  1                                    ),
;  352          1254  1          !
;  353          1255  1          ! Define the INADR vector used in the $SETPRT call.
;  354          1256  1          ! Note that VA_RANGE is on the next virtual page after VA_END.
;  355          1257  1          !
;  356          1258  1          VA_END          : VECTOR [0],           ! End of 'user data'
;  357          1259  1          VA_RANGE        : VECTOR [2] INITIAL (VA_START, VA_END) ALIGN (9);
;  358          1260  1
;  359          1261  1
;  360          1262  1 BIND
;  361          1263  1          !
;  362          1264  1          ! This is the delta-time value for all timers used.
;  363          1265  1          ! The time is a quadword value, is currently set for 5 seconds.
;  364          1266  1          !
;  365          1267  1          DELTA_TIME      = UPLIT (-5 * 10000000, -1);
```

```
;  367          1268  1 GLOBAL ROUTINE SYS$MOUNT  (ITEM_LIST) =
;  368          1269  1 !++
;  369          1270  1 ! Functional description:
;  370          1271  1 !
;  371          1272  1 !       This routine is the main entry point of the $MOUNT system service,
;  372          1273  1 !       and executes in the access mode of the caller.  Usually this will
;  373          1274  1 !       be USER mode.  This routine others defined in this module implement
;  374          1275  1 !       the logic for "operator assisted mount".  This code must execute
;  375          1276  1 !       in  USER mode, to allow users to CTRL\Y out of a mount request.
;  376          1277  1 !
;  377          1278  1 ! Input:
;  378          1279  1 !
;  379          1280  1 !       ITEM_LIST       : Address of a $GETJPI-like item list
;  380          1281  1 !
;  381          1282  1 ! Output:
;  382          1283  1 !
;  383          1284  1 !       None.
;  384          1285  1 !
;  385          1286  1 ! Implicit Inputs:
;  386          1287  1 !
;  387          1288  1 !       The MOUNT data base.
;  388          1289  1 !
;  389          1290  1 ! Implicit Outputs:                                                    ; F
;  390          1291  1 !
;  391          1292  1 !       The MOUNT data base may be altered as
;  392          1293  1 !       the result of operator intervention.                          ;
;  393          1294  1 !
;  394          1295  1 !--
;  395          1296  1
;  396          1297  2 BEGIN                                           ! Start of OPERATOR_ASSIST
;  397          1298  2
;  398          1299  2 BUILTIN
;  399          1300  2         FP,
;  400          1301  2         AP,
;  401          1302  2         CALLG;
;  402          1303  2
;  403          1304  2 LOCAL
;  404          1305  2         STATUS;
;  405          1306  2
;  406          1307  2 EXTERNAL
;  407          1308  2         MOUNT_OPTIONS   : BITVECTOR VOLATILE;   ! Mount options bit vector
;  408          1309  2
;  409          1310  2 EXTERNAL ROUTINE
;  410          1311  2         $DALLOC_DEVS$U  : ADDRESSING_MODE (GENERAL),    ! Address of transfer vector
;  411          1312  2         $CHANGE_PROT$U  : ADDRESSING_MODE (GENERAL),    ! Address of the transfer vector
;  412          1313  2         SYS$VMOUNT$U    : ADDRESSING_MODE (GENERAL);    ! Address of the transfer vector
;  413          1314  2
;  414          1315  2 !
;  415          1316  2 ! Enable a condition handler that will force the primary
;  416          1317  2 ! condition code facility-code to the MOUNT facility.
;  417          1318  2 !
;  418          1319  2 ENABLE INTERCEPT_SIGNAL;
;  419          1320  2
;  420          1321  2 !
;  421          1322  2 ! Set the page protection of this module's data to allow user
;  422          1323  2 ! mode read/write access.  This must be done here, since this
;  423          1324  2 ! image is INSTALLed as a protected shareable image, which has
```

```
;  424          1325  2 ! the effect of setting the protection to be USER read, EXEC write.
;  425          1326  2 ! Note that the data sits in a specail PSECT, to avoid changing
;  426          1327  2 ! the page protection on adjacent pages.
;  427          1328  2 !
;  428          1329  3 IF NOT (MOUNT_STATUS = $CHANGE_PROT$U ())
;  429          1330  2 THEN
;  430          1331  2     RETURN (.MOUNT_STATUS);
;  431          1332  2 !
;  432          1333  2 ! Initialize the necessary variables.  Most of the
;  433          1334  2 ! descriptors are not significantly changed, and do
;  434          1335  2 ! not have to be initialized at run time.
;  435          1336  2 !
;  436          1337  2 REPLY_PENDING = FALSE;
;  437          1338  2 MOUNT_FAILED = TRUE;
;  438          1339  2 OPERATOR_PRESENT = TRUE;
;  439          1340  2 PREVIOUS_STATUS = -1;
;  440          1341  2 PREVIOUS_DEV_IDX = -1;
;  441          1342  2 RETRY_COUNTER = 0;
;  442          1343  2 SS_FAIL_MODE = 0;
;  443          1344  2
;  444          1345  2 !
;  445          1346  2 ! Clear the system service failure exception flag, but save it's state.
;  446          1347  2 !
;  447          1348  2 STATUS = $SETSFM (ENBFLG=0);
;  448          1349  3 IF (.STATUS EQL SS$_WASSET)
;  449          1350  2 THEN
;  450          1351  2     SS_FAIL_MODE = 1;
;  451          1352  2
;  452          1353  2 !
;  453          1354  2 ! Set up the exit handler descriptor and declare the handler.
;  454          1355  2 !
;  455          1356  2 EXIT_HNDLR_DSC[XHNDLR_ADDRESS] = EXIT_HANDLER;
;  456          1357  2 EXIT_HNDLR_DSC[XHNDLR_ARGCNT]  = 1;
;  457          1358  2 EXIT_HNDLR_DSC[XHNDLR_STSADDR] = MOUNT_STATUS;
;  458          1359  2 $CANEXH (DESBLK = EXIT_HNDLR_DSC);
;  459          1360  2 $DCLEXH (DESBLK=EXIT_HNDLR_DSC);
;  460          1361  2
;  461          1362  2 !
;  462          1363  2 ! Perform the mount request.  If it fails, attempt to recover
;  463          1364  2 ! via some operator assistance.  If that is not possible, or the
;  464          1365  2 ! operator or user aborts the mount, die gracefully and return the
;  465          1366  2 ! status to the user.
;  466          1367  2 !
;  467          1368  2 MOUNT_STATUS = 0;
;  468          1369  2 VOLINV_COUNT = 0;
;  469          1370  2 WHILE NOT .MOUNT_STATUS DO
;  470          1371  3     BEGIN
;  471          1372  4     IF NOT (MOUNT_STATUS = CALLG (.AP, SYS$VMOUNT$U))
;  472          1373  3     THEN
;  473          1374  3         IF NOT .MOUNT_OPTIONS [OPT_ASSIST]
;  474          1375  3         THEN
;  475          1376  3
;  476          1377  4             BEGIN
;  477          1378  4             !
;  478          1379  4             ! If the mount operation failed for some reason other than VOLINV,
;  479          1380  4             ! exit loop with the error status. Else, do a limited number of
;  480          1381  4             ! retries. This automatic retry is implemented due to a race
```

```
;  481          1382  4              ! between mount and mount-verification. If mount is in progress
;  482          1383  4              ! and some event (e.g. cluster state transition) triggers mount-
;  483          1384  4              ! verification, mount-verification will clear the volume-valid
;  484          1385  4              ! bit in the UCB, causing mount to fail with a VOLINV error.
;  485          1386  4              !
;  486          1387  4              ! Not that the VOLINV error message will be suppressed (in module
;  487          1388  4              ! VMOUNT) unless the last retry fails with a VOLINV error.
;  488          1389  4              !
;  489          1390  5              IF (.MOUNT_STATUS AND STS$M_MSG_NO) NEQ (SS$_VOLINV AND STS$M_MSG_NO)
;  490          1391  4              THEN
;  491          1392  4                  EXITLOOP;
;  492          1393  4              VOLINV_COUNT = .VOLINV_COUNT + 1;
;  493          1394  4              IF .VOLINV_COUNT GEQ VOLINV_LIMIT
;  494          1395  4              THEN
;  495          1396  4                  EXITLOOP;
;  496          1397  4              END
;  497          1398  4
;  498          1399  3          ELSE
;  499          1400  3
;  500          1401  4              BEGIN
;  501          1402  4              !
;  502          1403  4              ! SELECT an error recovery handler based on the mount status value.
;  503          1404  4              ! Use only the message number and the facility code in the comparisons.
;  504          1405  4              !
;  505          1406  4              SELECTONEU (.MOUNT_STATUS AND STS$M_MSG_NO) OF
;  506          1407  4                  SET
;  507          1408  4                  [SS$_DEVALLOC    AND STS$M_MSG_NO]      : ALLOCFAIL_HNDLR ();
;  508          1409  4                  [SS$_MEDOFL      AND STS$M_MSG_NO]      : MEDOFL_HNDLR ();
;  509          1410  4                  [SS$_VOLINV      AND STS$M_MSG_NO]      : MEDOFL_HNDLR ();
;  510          1411  4                  [SS$_NODEVAVL    AND STS$M_MSG_NO]      : ALLOCFAIL_HNDLR ();
;  511          1412  4                  [SS$_NOSUCHDEV   AND STS$M_MSG_NO]      : ALLOCFAIL_HNDLR ();
;  512          1413  4                  [SS$_INCVOLLABEL AND STS$M_MSG_NO]      : WRONGVOL_HNDLR ();
;  513          1414  4                  [OTHERWISE]                             : EXITLOOP;
;  514          1415  4                  TES;
;  515          1416  4
;  516          1417  4                  !
;  517          1418  4                  ! Check for a reply to the operator request.  If it has
;  518          1419  4                  ! arrived, it will be processed.  If it hasn't, wait for
;  519          1420  4                  ! a few seconds and try again.
;  520          1421  4                  !
;  521          1422  4                  CHECK_FOR_REPLY ();
;  522          1423  3                  END;
;  523          1424  2          END;
;  524          1425  2
;  525          1426  2          !
;  526          1427  2          ! Attempt to deallocate devices that are not mounted and
;  527          1428  2          ! were not previously allocated.
;  528          1429  2          !
;  529          1430  2          ! If the mount interlock on this device is still in effect, dequeue it now.
;  530          1431  2          !
;  531          1432  2          ! Cancel the any outstanding requests and the exit handler.
;  532          1433  2          ! Also restore the system service failure exception flag to its
;  533          1434  2          ! original state, and disable the condition handler.
;  534          1435  2          !
;  535          1436  2          $DALLOC_DEVS$U (0);                             ! Attempt to deallocate devices
;  536          1437  2          CANCEL_REQUEST (REQUEST_SATISFIED);
;  537          1438  2          $SETSFM (ENBFLG = .SS_FAIL_MODE);
```

```
;  538          1439  2 .FP = 0;
;  539          1440  2 $CANEXH (DESBLK = EXIT_HNDLR_DSC);
;  540          1441  2
;  541          1442  3 RETURN (.MOUNT_STATUS)                    ! Return the status code
;  542          1443  3
;  543          1444  1 END;                                      ! End of SYS$MOUNT


                                        .TITLE  ASSIST
                                        .IDENT  \V04-001\

                                        .PSECT  $USER_DATA$,NOEXE,9

                                00000 VA_START::
                                                .BLKB   0
                                00000 VOLINV_COUNT::
                                                .BLKB   4
                                00004 REPLY_PENDING::
                                                .BLKB   4
                                00008 MOUNT_FAILED::
                                                .BLKB   4
                                0000C OPERATOR_PRESENT::
                                                .BLKB   4
                                00010 RETRY_COUNTER::
                                                .BLKB   4
                                00014 SS_FAIL_MODE::
                                                .BLKB   4
                                00018 MOUNT_STATUS::
                                                .BLKB   4
                                0001C PREVIOUS_STATUS::
                                                .BLKB   4
                                00020 PREVIOUS_DEV_IDX::
                                                .BLKB   4
                                00024 OPERATOR_MASK::
                                                .BLKB   4
                                00028 REQUEST_ID::
                                                .BLKB   4
                                0002C EXIT_HNDLR_DSC::
                                                .BLKB   16
                                0003C REPLY_CHANNEL::
                                                .BLKB   4
                                00040 REPLY_IOSB::
                                                .BLKB   8
                                00048 REPLY_BUFFER::
                                                .BLKB   136
                          0088  000D0 REPLY_DESC::
                                                .WORD   136                     ;
                            0E  000D2           .BYTE   14                      ;
                            01  000D3           .BYTE   1                       ;
                     00000000'  000D4           .ADDRESS REPLY_BUFFER           ;
                    00000002    00000008  000D8 TPARSE_BLOCK::
                                                .LONG   8, 2                    ;
                                000E0           .BLKB   28
                          003F  000FC DEVICE_DESC::
                                                .WORD   63                      ;
                            0E  000FE           .BYTE   14                      ;
                            01  000FF           .BYTE   1                       ;
```

```
                00000000  0C100          .LONG   0                                                       ;
                      03  00104 OP_MSG_BUF::
                                         .BYTE   3                                                       ;
                          00105          .BLKB   127
                    0080  00184 OP_MSG_DESC::
                                         .WORD   .28                                                     ;
                      0E  00186          .BYTE   14                                                      ;
                      01  00187          .BYTE   1                                                       ;
                00000000' 00188          .ADDRESS OP_MSG_BUF                                             ;
                      0E  0018C CANCEL_MSG_BUF::
                                         .BYTE   14                                                      ;
                      04  0018D          .BYTE   4                                                       ;
                          0018E          .BLKB   24
                          001A6          .BLKB   2
                    001A  001A8 CANCEL_MSG_DESC::
                                         .WORD   26                                                      ;
                      0E  001AA          .BYTE   14                                                      ;
                      01  001AB          .BYTE   1                                                       ;
                00000000' 001AC          .ADDRESS CANCEL_MSG_BUF                                         ;
                          001B0 FAO_BUFFER::
                                         .BLKB   512
                    0040  003B0 FAO_RESULT_DESC::
                                         .WORD   64                                                      ;
                      0E  003B2          .BYTE   14                                                      ;
                      01  003B3          .BYTE   1                                                       ;
                00000000' 003B4          .ADDRESS FAO_BUFFER                                             ;
                          003B8 VA_END::.BLKB   0
                          003B8          .BLKB   72
    00000000' 00000000' 00400 VA_RANGE::
                                         .ADDRESS VA_START, VA_END                                       ;

                                         .PSECT  $PLIT$,NOWRT,NOEXE,2

    FFFFFFFF  FD050F80  00000 P.AAA:     .LONG   -50000000, -1                                           ;

                                VOLINV_LIMIT==      20
                                DELTA_TIME=         P.AAA
                                         .EXTRN  MOUNT_OPTIONS, $DALLOC_DEVS$U
                                         .EXTRN  $CHANGE_PROT$U, SYS$VMOUNT$U
                                         .EXTRN  SYS$SETSFM, SYS$CANEXH
                                         .EXTRN  SYS$DCLEXH

                                         .PSECT  $CODE$,NOWRT,2

                          003C 00000             .ENTRY  SYS$MOUNT, Save R2,R3,R4,R5                     ; 1268
                55 00000000G   00  9E 00002      MOVAB   SYS$CANEXH, R5                                  ;
                54 00000000G   00  9E 00009      MOVAB   SYS$SETSFM, R4                                  ;
                53      0000'  CF  9E 00010      MOVAB   MOUNT_STATUS, R3                                ;
                6D      0108   CF  DE 00015      MOVAL   13$, (FP)                                       ; 1297
  00000000G     00             00  FB 0001A      CALLS   #0, $CHANGE_PROT$U                              ; 1329
                63             50  D0 00021      MOVL    R0, MOUNT_STATUS                                ;
                03             50  E8 00024      BLBS    R0, 1$                                          ;
                          00F3 31  00027          BRW     12$                                             ;
                      EC   A3  D4 0002A 1$:      CLRL    REPLY_PENDING                                   ; 1337
              F0      A3     01  D0 0002D        MOVL    #1, MOUNT_FAILED                                ; 1338
              F4      A3     01  D0 00031        MOVL    #1, OPERATOR_PRESENT                            ; 1339
              04      A3     01  CE 00035        MNEGL   #1, PREVIOUS_STATUS                             ; 1340
```

```
                08   A3                 01 CE 00039          MNEGL   #1, PREVIOUS_DEV_IDX                    ; 1341
                                   F8   A3 D4 0003D          CLRL    RETRY_COUNTER                           ; 1342
                                   FC   A3 D4 00040          CLRL    SS_FAIL_MODE                            ; 1343
                                        7E D4 00043          CLRL    -(SP)                                   ; 1348
                     64                 01 FB 00045          CALLS   #1, SYS$SETSFM
                     09                 50 D1 00048          CMPL    STATUS, #9                              ; 1349
                                        04 12 0004B          BNEQ    2$
                   FC   A3              01 D0 0004D          MOVL    #1, SS_FAIL_MODE                        ; 1351
                   18   A3       0000V  CF 9E 00051 2$:      MOVAB   EXIT_HANDLER, EXIT_HNDLR_DSC+4          ; 1356
                   1C   A3              01 D0 00057          MOVL    #1, EXIT_HNDLR_DSC+8                    ; 1357
                   20   A3              63 9E 0005B          MOVAB   MOUNT_STATUS, EXIT_HNDLR_DSC+12         ; 1358
                                   14   A3 9F 0005F          PUSHAB  EXIT_HNDLR_DSC                          ; 1359
                     65                 01 FB 00062          CALLS   #1, SYS$CANEXH
                                   14   A3 9F 00065          PUSHAB  EXIT_HNDLR_DSC                          ; 1360
       00000000G   00                   01 FB 00068          CALLS   #1, SYS$DCLEXH
                                        63 D4 0006F          CLRL    MOUNT_STATUS                            ; 1368
                                   E8   A3 D4 00071          CLRL    VOLINV_COUNT                            ; 1369
                     2D                 63 E8 00074 3$:      BLBS    MOUNT_STATUS, 4$                        ; 1370
       00000000G   00                   6C FA 00077          CALLG   (AP), SYS$VMOUNT$U                      ; 1372
                     63                 50 D0 0007E          MOVL    R0, MOUNT_STATUS
                     F0                 50 E8 00081          BLBS    R0, 3$
             1C      0000G    CF        02 E0 00084          BBS     #2, MOUNT_OPTIONS+6, 5$                 ; 1374
             50              63 FFFF0007 8F CB 0008A          BICL3   #-65529, MOUNT_STATUS, R0               ; 1390
       00000250   8F                    50 D1 00092          CMPL    R0, #592
                                        64 12 00099          BNEQ    11$
                                   E8   A3 D6 0009B          INCL    VOLINV_COUNT                            ; 1393
                     14            E8   A3 D1 0009E          CMPL    VOLINV_COUNT, #20                       ; 1394
                                        D0 19 000A2          BLSS    3$
                                        59 11 000A4 4$:      BRB     11$                                     ; 1396
             52              63 FFFF0007 8F CB 000A6 5$:      BICL3   #-65529, MOUNT_STATUS, R2               ; 1406
       00000840   8F                    52 D1 000AE          CMPL    R2, #2112                               ; 1408
                                        2B 13 000B5          BEQL    8$
       000001A0   8F                    52 D1 000B7          CMPL    R2, #416                                ; 1409
                                        09 13 000BE          BEQL    6$
       00000250   8F                    52 D1 000C0          CMPL    R2, #592                                ; 1410
                                        07 12 000C7          BNEQ    7$
            0000V   CF                  00 FB 000C9 6$:      CALLS   #0, MEDOFL_HNDLR
                                        27 11 000CE          BRB     10$
       000009B0   8F                    52 D1 000D0 7$:      CMPL    R2, #2480                               ; 1411
                                        09 13 000D7          BEQL    8$
       00000908   8F                    52 D1 000D9          CMPL    R2, #2312                               ; 1412
                                        07 12 000E0          BNEQ    9$
            0000V   CF                  00 FB 000E2 8$:      CALLS   #0, ALLOCFAIL_HNDLR
                                        0E 11 000E7          BRB     10$
       00000108   8F                    52 D1 000E9 9$:      CMPL    R2, #264                                ; 1413
                                        0D 12 000F0          BNEQ    11$
            0000V   CF                  00 FB 000F2          CALLS   #0, WRONGVOL_HNDLR
            0000V   CF                  00 FB 000F7 10$:     CALLS   #0, CHECK_FOR_REPLY                     ; 1422
                                  FF75  31 000FC             BRW     3$                                      ; 1374
                                        7E D4 000FF 11$:     CLRL    -(SP)                                   ; 1436
       00000000G   00                   01 FB 00101          CALLS   #1, $DALLOC_DEVS$U
                                        01 DD 00108          PUSHL   #1                                      ; 1437
            0000V   CF                  01 FB 0010A          CALLS   #1, CANCEL_REQUEST
                                   FC   A3 DD 0010F          PUSHL   SS_FAIL_MODE                            ; 1438
                     64                 01 FB 00112          CALLS   #1, SYS$SETSFM
                                        6D D4 00115          CLRL    (FP)                                    ; 1439
                                   14   A3 9F 00117          PUSHAB  EXIT_HNDLR_DSC                          ; 1440
```

```
                        65              01  FB 0011A           CALLS   #1, SYS$CANEXH                          ;
                        50              63  D0 0011D 12$:      MOVL    MOUNT_STATUS, R0                        ; 1442
                                            04 00120           RET                                             ; 1444
                                          0000 00121 13$:      .WORD   Save nothing                            ; 1297
                                        7E  D4 00123           CLRL    -(SP)                                   ;
                                        5E  DD 00125           PUSHL   SP                                      ;
                        7E      04      AC  7D 00127           MOVQ    4(AP), -(SP)                            ;
                  0000V CF              03  FB 0012B           CALLS   #3, INTERCEPT_SIGNAL                    ;
                                            04 00130           RET                                             ;

; Routine Size:  305 bytes,    Routine Base:  $CODE$ + 0000
```

```
;   545          1445  1 ROUTINE INTERCEPT_SIGNAL (SIGNAL, MECHANISM) =
;   546          1446  1
;   547          1447  1 !++
;   548          1448  1 ! Functional Description:
;   549          1449  1 !
;   550          1450  1 !       This routine is a conditon handler whose sole
;   551          1451  1 !       reason for existence is to force the primary
;   552          1452  1 !       conditon code's facility-code to that of the
;   553          1453  1 !       MOUNT facility.
;   554          1454  1 !
;   555          1455  1 ! Input:
;   556          1456  1 !
;   557          1457  1 !       SIGNAL    = Address of the signal array
;   558          1458  1 !       MECHANISM = Address of the mechanism array
;   559          1459  1 !
;   560          1460  1 ! Output:
;   561          1461  1 !
;   562          1462  1 !       The condition facility code is equal to MOUN$_FACILITY
;   563          1463  1 !--
;   564          1464  1
;   565          1465  2 BEGIN                                           ! Start of INTERCEPT_SIGNAL
;   566          1466  2
;   567          1467  2 MAP
;   568          1468  2
;   569          1469  2         SIGNAL          : REF BBLOCK,           ! Signal array
;   570          1470  2         MECHANISM       : REF BBLOCK;           ! Mechanism array
;   571          1471  2
;   572          1472  2 EXTERNAL
;   573          1473  2
;   574          1474  2         MOUNT_OPTIONS   : BITVECTOR VOLATILE,   ! parser option flags
;   575          1475  2         USER_STATUS     : VECTOR;               ! Status return of some routines
;   576          1476  2
;   577          1477  2
;   578          1478  2 IF .SIGNAL[CHF$L_SIG_NAME] NEQ SS$_UNWIND
;   579          1479  2 THEN
;   580          1480  3     BEGIN
;   581          1481  3     !
;   582          1482  3     ! Make the facility code MOUN$_FCILITY.
;   583          1483  3     !
;   584          1484  3     IF .BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_FAC_NO] EQL 0
;   585          1485  3     OR .BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_FAC_NO] EQL INIT$_FACILITY
;   586          1486  3     THEN
;   587          1487  3         BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_FAC_NO] = MOUN$_FACILITY;
;   588          1488  3
;   589          1489  3     IF .BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_MSG_NO] EQL 0
;   590          1490  3     THEN
;   591          1491  3         BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_MSG_NO] = .USER_STATUS [0] ^ (-$BITPOSITION (STS$V_MSG_NO));
;   592          1492  3
;   593          1493  3     !
;   594          1494  3     ! If the caller requested it, print the message text associated with the message code.
;   595          1495  3     !
;   596          1496  3     IF .MOUNT_OPTIONS [OPT_MESSAGE]
;   597          1497  3     THEN
;   598          1498  4         BEGIN
;   599          1499  4         SIGNAL [CHF$L_SIG_ARGS] = .SIGNAL [CHF$L_SIG_ARGS] - 2;
;   600          1500  4         $PUTMSG (MSGVEC = SIGNAL [CHF$L_SIG_ARGS], ACTRTN=0, FACNAM=0);
;   601          1501  4         SIGNAL [CHF$L_SIG_ARGS] = .SIGNAL [CHF$L_SIG_ARGS] + 2;
```

```
;   602          1502  4          BBLOCK [SIGNAL [CHF$L_SIG_NAME], STS$V_INHIB_MSG] = 1;
;   603          1503  3          END;
;   604          1504  3
;   605          1505  3      !
;   606          1506  3      ! If the condition severity code is SEVERE or ERROR, then unwind the
;   607          1507  3      ! stack back to the caller of the frame that established this handler.
;   608          1508  3      ! Return the condition code in R0.
;   609          1509  3      !
;   610          1510  3      IF .BBLOCK [SIGNAL [CHF$L_SIG_NAME], STS$V_SEVERITY] EQL STS$K_SEVERE
;   611          1511  3      OR .BBLOCK [SIGNAL [CHF$L_SIG_NAME], STS$V_SEVERITY] EQL STS$K_ERROR
;   612          1512  3      THEN
;   613          1513  4          BEGIN
;   614          1514  4          MECHANISM [CHF$L_MCH_SAVR0] = .SIGNAL [CHF$L_SIG_NAME];
;   615          1515  4          $UNWIND ();
;   616          1516  3          END;
;   617          1517  2      END;
;   618          1518  2
;   619          1519  2 !
;   620          1520  2 ! Attempt to continue the operation.
;   621          1521  2 !
;   622          1522  2 RETURN (SS$_CONTINUE);
;   623          1523  2
;   624          1524  1 END;                                    ! End of INTERCEPT_SIGNAL


                                                                .EXTRN  USER_STATUS, SYS$PUTMSG
                                                                .EXTRN  SYS$UNWIND

                                                    000C 00000 INTERCEPT_SIGNAL:
                                                                .WORD   Save R2,R3                      ; 1445
                                                    52      04 AC  D0 00002         MOVL    SIGNAL, R2  ; 1478
                                                    53      04 A2  9E 00006         MOVAB   4(R2), R3   ;
                                          00000920  8F         63  D1 0000A         CMPL    (R3), #2336 ;
                                                               6D  13 00011         BEQL    6$          ;
                                              0FFF  8F      02 A3  B3 00013         BITW    2(R3), #4095; 1484
                                                               0C  13 00019         BEQL    1$          ;
00000075  8F      02  A3                      0C               00  ED 0001B         CMPZV   #0, #12, 2(R3), #117; 1485
                                                               0A  12 00025         BNEQ    2$          ;
      02  A3                                0C               00  8F  F0 00027 1$: INSV    #114, #0, #12, 2(R3); 1487
                                              FFF8  8F         63  B3 00031 2$: BITW    (R3), #65528; 1489
                                                               0C  12 00036         BNEQ    3$          ;
                                    50        0000G CF      FD 8F  78 00038         ASHL    #-3, USER_STATUS, R0; 1491
          63                        0D              03         50  F0 0003F         INSV    R0, #3, #13, (R3)
                                    17        0000G CF         03  E1 00044 3$: BBC     #3, MOUNT_OPTIONS+6, 4$; 1496
                                              62               02  C2 0004A         SUBL2   #2, (R2)    ; 1499
                                                               7E  7C 0004D         CLRQ    -(SP)       ; 1500
                                                               7E  D4 0004F         CLRL    -(SP)       ;
                                                               52  DD 00051         PUSHL   R2          ;
                                    00000000G 00               04  FB 00053         CALLS   #4, SYS$PUTMSG;
                                              62               02  C0 0005A         ADDL2   #2, (R2)    ; 1501
                                        03    A3               10  88 0005D         BISB2   #16, 3(R3)  ; 1502
          04                        63        03               00  ED 00061 4$: CMPZV   #0, #3, (R3), #4; 1510
                                                               07  13 00066         BEQL    5$          ;
          02                        63        03               00  ED 00068         CMPZV   #0, #3, (R3), #2; 1511
                                                               11  12 0006D         BNEQ    6$          ;
                                              50        08 AC  D0 0006F 5$: MOVL    MECHANISM, R0       ; 1514
```

```
                        0C   A0          63  D0 00073          MOVL    (R3), 12(R0)                                ;
                                         7E  7C 00077          CLRQ    -(SP)                                       ; 1515
                00000000G    00          02  FB 00079          CALLS   #2, SYS$UNWIND                              ;
                             50          01  D0 00080 6$:      MOVL    #1, R0                                      ; 1522
                                             04 00083          RET                                                 ; 1524

; Routine Size:  132 bytes,    Routine Base:  $CODE$ + 0131
```

```
;  626          1525  1 ROUTINE POST_READ_TO_MBX (MBX_CHANNEL) : NOVALUE =
;  627          1526  1
;  628          1527  1 !++
;  629          1528  1 ! Functional description:
;  630          1529  1 !
;  631          1530  1 !       This routine will post a read to the reply mailbox.
;  632          1531  1 !       Instead of waiting for the I/O to complete, request
;  633          1532  1 !       that an event flag be set when the I/O is finally done.
;  634          1533  1 !
;  635          1534  1 ! Input:
;  636          1535  1 !
;  637          1536  1 !       None.
;  638          1537  1 !
;  639          1538  1 ! Implicit Input:
;  640          1539  1 !
;  641          1540  1 !       REPLY_CHANNEL   : Channel # of channel to the reply mailbox.
;  642          1541  1 !
;  643          1542  1 ! Output:
;  644          1543  1 !
;  645          1544  1 !       None.
;  646          1545  1 !
;  647          1546  1 ! Implict output:
;  648          1547  1 !
;  649          1548  1 !       REPLY_IOSB      : Address of an I/O status block to receive the status of the I/O.
;  650          1549  1 !       REPLY_BUFFER    : Address of buffer to receive the operator's reply.
;  651          1550  1 !
;  652          1551  1 ! Side effects:
;  653          1552  1 !
;  654          1553  1 !       If the $QIO fails, the user will be notified
;  655          1554  1 !       of the failure and the mount will be aborted.
;  656          1555  1 !
;  657          1556  1 ! Routine value:
;  658          1557  1 !
;  659          1558  1 !       None.
;  660          1559  1 !--
;  661          1560  1
;  662          1561  2 BEGIN                                           ! Start of POST_READ_TO_MBX
;  663          1562  2
;  664          1563  2 LOCAL
;  665          1564  2         STATUS          : LONG;                ! Hold status of $QIO call
;  666          1565  2
;  667        P 1566  3 IF NOT (STATUS = $QIO   (FUNC = IO$_READVBLK,
;  668        P 1567  3                          EFN  = REPLY_FLAG,
;  669        P 1568  3                          CHAN = .REPLY_CHANNEL,
;  670        P 1569  3                          IOSB = REPLY_IOSB,
;  671        P 1570  3                          P1   = REPLY_BUFFER,
;  672        P 1571  3                          P2   = ($BYTEOFFSET (OPC$S_MS_OTEXT) + $BYTEOFFSET (OPC$L_MS_TEXT))
;  673          1572  3                         ))
;  674          1573  2 THEN
;  675          1574  2     ABORT_MOUNT (MOUN$_MBXRDERR, 0, .STATUS);
;  676          1575  2
;  677          1576  1 END;                                           ! End of POST_READ_TO_MBX


                                                  .EXTRN  SYS$QIO
```

```
                                        0000 00000 POST_READ_TO_MBX:
                                                           .WORD   Save nothing                                    ; 1525
                                             7E  7C 00002          CLRQ    -(SP)                                  ; 1572
                                             7E  7C 00004          CLRQ    -(SP)                                  ;
                                    7E        88  8F  9A 00006          MOVZBL  #136, -(SP)                            ;
                                          0000' CF  9F 0000A          PUSHAB  REPLY_BUFFER                           ;
                                             7E  7C 0000E          CLRQ    -(SP)                                  ;
                                          0000' CF  9F 00010          PUSHAB  REPLY_IOSB                             ;
                                             31  DD 00014          PUSHL   #49                                    ;
                                          0000' CF  DD 00016          PUSHL   REPLY_CHANNEL                          ;
                                             1A  DD 0001A          PUSHL   #26                                    ;
                        00000000G         00     0C  FB 0001C          CALLS   #12, SYS$QIO                           ;
                                          11     50  E8 00023          BLBS    STATUS, 1$                             ;
                                                 50  DD 00026          PUSHL   STATUS                                 ; 1574
                                                 7E  D4 00028          CLRL    -(SP)                                  ;
                                       007281DC  8F  DD 0002A          PUSHL   #7504348                               ;
                        00000000G         00     03  FB 00030          CALLS   #3, LIB$STOP                           ;
                                                     04 00037 1$:      RET                                            ; 1576

; Routine Size:  56 bytes,    Routine Base:  $CODE$ + 01B5
```

```
;  679          1577  1 ROUTINE INTERACTIVE_JOB =
;  680          1578  1
;  681          1579  1 !++
;  682          1580  1 ! Functional Description:
;  683          1581  1 !
;  684          1582  1 !       This routine will determine if the current process is an
;  685          1583  1 !       interactive process, and return that information to the
;  686          1584  1 !       caller.  By definition, a process is interactive if it
;  687          1585  1 !       has a terminal associated with it.
;  688          1586  1 !
;  689          1587  1 ! Input:
;  690          1588  1 !
;  691          1589  1 !       None.
;  692          1590  1 !
;  693          1591  1 ! Output:
;  694          1592  1 !
;  695          1593  1 !       None.
;  696          1594  1 !
;  697          1595  1 ! Routine Value:
;  698          1596  1 !
;  699          1597  1 !       1 if current process is an interactive process
;  700          1598  1 !       0 if current process is not an interactive process
;  701          1599  1 !--
;  702          1600  1
;  703          1601  2 BEGIN                                           ! Start of INTERACTIVE_JOB
;  704          1602  2
;  705          1603  2 LOCAL
;  706          1604  2         ITEM_LIST       : BBLOCK [16],          ! Item list for $GETJPI
;  707          1605  2         DEVICE_NAME     : BBLOCK [16],          ! Device name buffer
;  708          1606  2         NAME_LENGTH     : LONG;                 ! Cell for device name length
;  709          1607  2
;  710          1608  2 !
;  711          1609  2 ! Build the $GETJPI item list and get the terminal name.
;  712          1610  2 !
;  713          1611  2 NAME_LENGTH = 0;                                ! Zero the output cell
;  714          1612  2 ITEM_LIST [0, 0, 16, 0] = 16;                  ! Set buffer length
;  715          1613  2 ITEM_LIST [2, 0, 16, 0] = JPI$_TERMINAL;       ! Set item code
;  716          1614  2 ITEM_LIST [4, 0, 32, 0] = DEVICE_NAME;        ! Set buffer address
;  717          1615  2 ITEM_LIST [8, 0, 32, 0] = NAME_LENGTH;        ! Set result length address
;  718          1616  2 ITEM_LIST [12,0, 32, 0] = 0;                  ! Set list terminator
;  719          1617  2 $GETJPI (ITMLST = ITEM_LIST);
;  720          1618  2
;  721          1619  2 !
;  722          1620  2 ! If a terminal is associated with the process, the terminal name
;  723          1621  2 ! length should be nonzero.
;  724          1622  2 !
;  725          1623  2 IF .NAME_LENGTH NEQ 0
;  726          1624  2 THEN
;  727          1625  2     1                                          ! Return TRUE
;  728          1626  2 ELSE
;  729          1627  2     0                                          ! Return FALSE
;  730          1628  2
;  731          1629  1 END;                                           ! End of INTERACTIVE_JOB


                                                .EXTRN  SYS$GETJPI
```

```
                                        0000 00000 INTERACTIVE_JOB:
                                                   .WORD   Save nothing                                ; 1577
                         5E                 20  C2 00002          SUBL2   #32, SP                     ;
                                            7E  D4 00005          CLRL    NAME_LENGTH                 ; 1611
                   14    AE 031D0010        8F  D0 00007          MOVL    #52232208, ITEM_LIST        ; 1612
                   18    AE         04      AE  9E 0000F          MOVAB   DEVICE_NAME, ITEM_LIST+4    ; 1614
                   1C    AE                 6E  9E 00014          MOVAB   NAME_LENGTH, ITEM_LIST+8    ; 1615
                                       20   AE  D4 00018          CLRL    ITEM_LIST+12                ; 1616
                                            7E  7C 0001B          CLRQ    -(SP)                       ; 1617
                                            7E  D4 0001D          CLRL    -(SP)                       ;
                                       20   AE  9F 0001F          PUSHAB  ITEM_LIST                   ;
                                            7E  7C 00022          CLRQ    -(SP)                       ;
                                            7E  D4 00024          CLRL    -(SP)                       ;
             00000000G                  00   07  FB 00026          CALLS   #7, SYS$GETJPI              ;
                                            6E  D5 0002D          TSTL    NAME_LENGTH                 ; 1623
                                            04  13 0002F          BEQL    1$                          ;
                         50                 01  D0 00031          MOVL    #1, R0                      ;
                                            04  00034          RET                                 ;
                                            50  D4 00035 1$:      CLRL    R0                          ;
                                                04 00037          RET                                 ; 1629

; Routine Size:  56 bytes,    Routine Base:  $CODE$ + 01ED

;  732          1630  1
```

```
;  734     1631  1 ROUTINE SUBMIT_REQUEST (MSG_DESC,REPLY_EXPECTED) : NOVALUE =
;  735     1632  1
;  736     1633  1 !++
;  737     1634  1 ! Functional Description:
;  738     1635  1 !
;  739     1636  1 !       This routine will send a request to all operators enabled
;  740     1637  1 !       to receive disk and tape messages.  All requests that are
;  741     1638  1 !       issued to the operator are echoed to the user.  Also, the
;  742     1639  1 !       request context is saved so that when the operator replies
;  743     1640  1 !       we can parse the reply in the context of the request.
;  744     1641  1 !
;  745     1642  1 ! Input:
;  746     1643  1 !
;  747     1644  1 !       MSG_DESC        = Address of a quadword string descriptor.
;  748     1645  1 !                         The string is the operator request.
;  749     1646  1 !
;  750     1647  1 !       REPLY_EXPECTED  = Boolean value.  If true then an operator
;  751     1648  1 !                         response is expected.
;  752     1649  1 !
;  753     1650  1 ! Output:
;  754     1651  1 !
;  755     1652  1 !       None.
;  756     1653  1 !
;  757     1654  1 ! Implicit Inputs:
;  758     1655  1 !
;  759     1656  1 !       MOUNT_STATUS    = status from current mount attempt
;  760     1657  1 !
;  761     1658  1 ! Implicit Outputs:
;  762     1659  1 !
;  763     1660  1 !       The request context is saved, the request is made.
;  764     1661  1 !--
;  765     1662  1
;  766     1663  2 BEGIN                                           ! Start of SUBMIT_REQUEST
;  767     1664  2
;  768     1665  2 MAP
;  769     1666  2
;  770     1667  2         MSG_DESC        : REF BBLOCK;           ! Address of request descriptor
;  771     1668  2
;  772     1669  2 EXTERNAL
;  773     1670  2
;  774     1671  2         DEVICE_INDEX    : LONG VOLATILE;        ! Index into device list
;  775     1672  2
;  776     1673  2 LITERAL
;  777     1674  2
;  778     1675  2         BLANK           = %ASCII ' ',           ! Fill character
;  779     1676  2         ZERO            = 0;                    ! Handy literal
;  780     1677  2
;  781     1678  2 LOCAL
;  782     1679  2
;  783     1680  2         STATUS          : LONG,                 ! Return status
;  784     1681  2         MBX_CHAN        : LONG;                 ! Operator reply mailbox channel
;  785     1682  2
;  786     1683  2
;  787     1684  2 !
;  788     1685  2 ! If no mailbox exists, create one.
;  789     1686  2 !
;  790     1687  2 IF .REPLY_CHANNEL EQL ZERO
```

```
;  791          1688  2 THEN
;  792          1689  3     IF NOT (STATUS = $CREMBX (CHAN = REPLY_CHANNEL, PROMSK = MAILBOX_PROTECTION))
;  793          1690  2     THEN
;  794          1691  2         ABORT_MOUNT (MOUN$_MBXCRERR, 0, .STATUS);
;  795          1692  2
;  796          1693  2 !
;  797          1694  2 ! Fill in the necessary fields in the request string.
;  798          1695  2 ! Copy the message string to the operator message buffer.
;  799          1696  2 !
;  800          1697  2 REQUEST_ID = .REQUEST_ID + 1;                    ! Inc reqeust #
;  801          1698  2 OP_MSG_BUF[OPC$L_MS_RQSTID]= .REQUEST_ID;        ! Set request #
;  802          1699  2 !
;  803          1700  2 CH$COPY (.MSG_DESC[DSC$W_LENGTH],                ! Source length
;  804          1701  2           .MSG_DESC[DSC$A_POINTER],              ! Source pointer
;  805          1702  2           BLANK,                                 ! Fill character
;  806          1703  2           OPC$S_MS_OTEXT-$BYTEOFFSET(OPC$L_MS_TEXT),! Destination length
;  807          1704  2           OP_MSG_BUF+$BYTEOFFSET(OPC$L_MS_TEXT)  ! Destination pointer
;  808          1705  2          );
;  809          1706  2 OP_MSG_DESC[DSC$W_LENGTH] = .MSG_DESC[DSC$W_LENGTH]+$BYTEOFFSET(OPC$L_MS_TEXT);
;  810          1707  2
;  811          1708  2 IF .REPLY_EXPECTED
;  812          1709  2 THEN
;  813          1710  3     BEGIN
;  814          1711  3     !
;  815          1712  3     ! An operator reply is expected.  Save the condition
;  816          1713  3     ! context and set up the reply mailbox channel.
;  817          1714  3     !
;  818          1715  3     PREVIOUS_STATUS  = .MOUNT_STATUS;
;  819          1716  3     PREVIOUS_DEV_IDX = .DEVICE_INDEX;
;  820          1717  3     REPLY_PENDING    =  TRUE;
;  821          1718  3     MBX_CHAN         = .REPLY_CHANNEL;
;  822          1719  3     END
;  823          1720  2 ELSE
;  824          1721  2     !
;  825          1722  2     ! An operator reply is not expected.
;  826          1723  2     ! Indicate this to OPCOM by specifying a mailbox channel of zero.
;  827          1724  2     !
;  828          1725  2     MBX_CHAN = ZERO;
;  829          1726  2
;  830          1727  2 !
;  831          1728  2 ! Set the operator target mask.
;  832          1729  2 !
;  833          1730  2 SET_TARGET_MASK ();
;  834          1731  2 OP_MSG_BUF[TARGET_FIELD] = .OPERATOR_MASK;
;  835          1732  2 !
;  836          1733  2 ! Send the request to the operator.
;  837          1734  2 !
;  838          1735  3 IF NOT (STATUS = $SNDOPR (MSGBUF=OP_MSG_DESC, CHAN=.MBX_CHAN))
;  839          1736  2 THEN
;  840          1737  2     ABORT_MOUNT (MOUN$_OPRSNDERR, 0, .STATUS);
;  841          1738  2 !
;  842          1739  2 ! Echo the operator request to the user.  If no operator is
;  843          1740  2 ! present, do not echo the request.  This interlock is necessary
;  844          1741  2 ! to prevent repeatedly issuing the request if no OPCOM process
;  845          1742  2 ! is present.
;  846          1743  2 !
;  847          1744  2 IF .OPERATOR_PRESENT
```

```
;  848     1745  2 THEN
;  849     1746  2     SIGNAL (MOUN$_OPRQST, 1, .MSG_DESC);
;  850     1747  2 !
;  851     1748  2 ! An alternate request status returned by $SNDOPR is SS$_NOPERATOR,
;  852     1749  2 ! which indicates that there is no operator present to service the
;  853     1750  2 ! request.  Taken in this context, it means that there is no OPCOM
;  854     1751  2 ! process present on the system.
;  855     1752  2 !
;  856     1753  2 IF .STATUS EQL OPC$_NOPERATOR
;  857     1754  2 THEN
;  858     1755  3     BEGIN
;  859     1756  3     REPLY_PENDING = FALSE;
;  860     1757  3     IF NOT INTERACTIVE_JOB ()
;  861     1758  3     THEN
;  862     1759  3         !
;  863     1760  3         ! Abort the mount, as no one can service the request.
;  864     1761  3         !
;  865     1762  3         ABORT_MOUNT (MOUN$_BATCHNOOPR)
;  866     1763  3     ELSE
;  867     1764  4         BEGIN
;  868     1765  4         !
;  869     1766  4         ! Inform the user that no operator is available to service
;  870     1767  4         ! the request.  The user then has three courses of action:
;  871     1768  4         !   - Abort the mount via CTRL-C
;  872     1769  4         !   - Wait for an operator to enable himself to service the request
;  873     1770  4         !   - Service the request himself.  (Hands-on environment)
;  874     1771  4         !
;  875     1772  4         ! Since the problem may go away in time, wait a short while after
;  876     1773  4         ! informing the user before continuing the MOUNT operation.
;  877     1774  4         !
;  878     1775  4         IF .OPERATOR_PRESENT
;  879     1776  4         THEN
;  880     1777  4             SIGNAL (MOUN$_NOOPR);
;  881     1778  4         OPERATOR_PRESENT = FALSE;
;  882     1779  5         IF NOT (STATUS = $SETIMR (EFN=TIMER_FLAG, REQIDT=TIMER_ID, DAYTIM=DELTA_TIME))
;  883     1780  4         THEN
;  884     1781  4             ABORT_MOUNT (.STATUS);
;  885     1782  4         $WAITFR (EFN = TIMER_FLAG);
;  886     1783  4         $CANTIM (REQIDT = TIMER_ID);
;  887     1784  4         $SETEF  (EFN = TIMER_FLAG);
;  888     1785  3         END;
;  889     1786  2     END;
;  890     1787  2 !
;  891     1788  2 ! If an operator reply is expected, then issue a read to the reply mailbox.
;  892     1789  2 !
;  893     1790  2 REPLY_IOSB = 0;
;  894     1791  2 IF .REPLY_PENDING
;  895     1792  2 THEN
;  896     1793  2     POST_READ_TO_MBX ();
;  897     1794  2
;  898     1795  1 END;                                   ! End of SUBMIT_REQUEST


                                                        .EXTRN  DEVICE_INDEX, SYS$CREMBX
                                                        .EXTRN  SYS$SNDOPR, SYS$SETIMR
                                                        .EXTRN  SYS$WAITFR, SYS$CANTIM
```

```
                                                        .EXTRN  SYS$SETEF

                                                07FC 00000 SUBMIT_REQUEST:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10                ; 1631
                                        5A 00000000G   00   9E 00002           MOVAB   LIB$SIGNAL, R10
                                        59 00000000G   00   9E 00009           MOVAB   LIB$STOP, R9
                                        58      0000'  CF   9E 00010           MOVAB   REPLY_CHANNEL, R8
                                                       68   D5 00015           TSTL    REPLY_CHANNEL               ; 1687
                                                       27   12 00017           BNEQ    1$
                                                       7E   7C 00019           CLRQ    -(SP)                       ; 1689
                                        7E      FF00   8F   3C 0001B           MOVZWL  #65280, -(SP)
                                                       7E   7C 00020           CLRQ    -(SP)
                                                       58   DD 00022           PUSHL   R8
                                                       7E   D4 00024           CLRL    -(SP)
                                00000000G   00         07   FB 00026           CALLS   #7, SYS$CREMBX
                                        57             50   D0 0002D           MOVL    R0, STATUS
                                        0D             57   E8 00030           BLBS    STATUS, 1$
                                                       57   DD 00033           PUSHL   STATUS                      ; 1691
                                                       7E   D4 00035           CLRL    -(SP)
                                    007281D4           8F   DD 00037           PUSHL   #7504340
                                        69             03   FB 0003D           CALLS   #3, LIB$STOP
                                          EC           A8   D6 00040 1$:       INCL    REQUEST_ID                  ; 1697
                                00CC      C8           EC   A8 D0 00043        MOVL    REQUEST_ID, OP_MSG_BUF+4    ; 1698
                                          56           04   AC D0 00049        MOVL    MSG_DESC, R6                ; 1700
    0078    8F          20        04      B6                 66   2C 0004D    MOVC5   (R6), @4(R6), #32, #120, OP_MSG_BUF+8  ; 1704
                                    00D0   C8                 00055
                0148    C8                 66                 08   A1 00058    ADDW3   #8, (R6), OP_MSG_DESC       ; 1706
                                          14           08   AC E9 0005E        BLBC    REPLY_EXPECTED, 2$          ; 1708
                                    E0    A8           DC   A8 D0 00062        MOVL    MOUNT_STATUS, PREVIOUS_STATUS  ; 1715
                                    E4    A8      0000G  CF   D0 00067         MOVL    DEVICE_INDEX, PREVIOUS_DEV_IDX  ; 1716
                                    C8    A8                 01   D0 0006D    MOVL    #1, REPLY_PENDING           ; 1717
                                          52                 68   D0 00071    MOVL    REPLY_CHANNEL, MBX_CHAN     ; 1718
                                                             02   11 00074    BRB     3$                          ; 1708
                                                             52   D4 00076 2$: CLRL    MBX_CHAN                    ; 1725
                                    0000V  CF                00   FB 00078 3$: CALLS   #0, SET_TARGET_MASK       ; 1730
    00C9    C8          18                 00        E8      A8   F0 0007D    INSV    OPERATOR_MASK, #0, #24, OP_MSG_BUF+1  ; 1731
                                                             52   DD 00085    PUSHL   MBX_CHAN                    ; 1735
                                              0148         C8   9F 00087    PUSHAB  OP_MSG_DESC
                                00000000G   00             02   FB 0008B    CALLS   #2, SYS$SNDOPR
                                        57                 50   D0 00092    MOVL    R0, STATUS
                                        0D                 57   E8 00095    BLBS    STATUS, 4$
                                                           57   DD 00098    PUSHL   STATUS                      ; 1737
                                                           7E   D4 0009A    CLRL    -(SP)
                                    007281EC               8F   DD 0009C    PUSHL   #7504364
                                        69                 03   FB 000A2    CALLS   #3, LIB$STOP
                                        0D        D0       A8   E9 000A5 4$: BLBC    OPERATOR_PRESENT, 5$        ; 1744
                                                           56   DD 000A9    PUSHL   R6                          ; 1746
                                                           01   DD 000AB    PUSHL   #1
                                    0072A023               8F   DD 000AD    PUSHL   #7512099
                                        6A                 03   FB 000B3    CALLS   #3, LIB$SIGNAL
                                00058061   8F              57   D1 000B6 5$: CMPL    STATUS, #360545             ; 1753
                                                           65   12 000BD    BNEQ    9$
                                              C8           A8   D4 000BF    CLRL    REPLY_PENDING               ; 1756
                                    FF01   CF              00   FB 000C2    CALLS   #0, INTERACTIVE_JOB         ; 1757
                                        0B                 50   E8 000C7    BLBS    R0, 6$
                                    007281FC               8F   DD 000CA    PUSHL   #7504380                    ; 1762
                                        69                 01   FB 000D0    CALLS   #1, LIB$STOP
```

```
                                 4F  11 000D3          BRB     9$
                           09     D0  A8  E9 000D5 6$:  BLBC    OPERATOR_PRESENT, 7$                    ;1775
                       0072A03B       8F  DD 000D9          PUSHL   #7512123                            ;1777
                           6A         01  FB 000DF          CALLS   #1, LIB$SIGNAL
                           D0         A8  D4 000E2 7$:  CLRL    OPERATOR_PRESENT                        ;1778
                       7E  03E7       8F  3C 000E5          MOVZWL  #999, -(SP)                         ;1779
                                      7E  D4 000EA          CLRL    -(SP)
                           0000'      CF  9F 000EC          PUSHAB  DELTA_TIME
                                      19  DD 000F0          PUSHL   #25
                 00000000G 00         04  FB 000F2          CALLS   #4, SYS$SETIMR
                           57         50  D0 000F9          MOVL    R0, STATUS
                           05         57  E8 000FC          BLBS    STATUS, 8$
                                      57  DD 000FF          PUSHL   STATUS                              ;1781
                           69         01  FB 00101          CALLS   #1, LIB$STOP
                                      19  DD 00104 8$:      PUSHL   #25                                 ;1782
                 00000000G 00         01  FB 00106          CALLS   #1, SYS$WAITFR
                                      7E  D4 0010D          CLRL    -(SP)                               ;1783
                       7E  03E7       8F  3C 0010F          MOVZWL  #999, -(SP)
                 00000000G 00         02  FB 00114          CALLS   #2, SYS$CANTIM
                                      19  DD 0011B          PUSHL   #25                                 ;1784
                 00000000G 00         01  FB 0011D          CALLS   #1, SYS$SETEF
                           04         A8  D4 00124 9$:      CLRL    REPLY_IOSB                          ;1790
                           05     C8  A8  E9 00127          BLBC    REPLY_PENDING, 10$                  ;1791
                       FE60   CF      00  FB 0012B          CALLS   #0, POST_READ_TO_MBX                ;1793
                                          04 00130 10$:     RET                                         ;1795

; Routine Size:  305 bytes,    Routine Base:  $CODE$ + 0225

;  899          1796  1
```

```
;  901          1797  1 ROUTINE SET_TARGET_MASK : NOVALUE =
;  902          1798  1
;  903          1799  1 !++
;  904          1800  1 ! Functional description:
;  905          1801  1 !
;  906          1802  1 !       Get the device characteristics and figure out which class
;  907          1803  1 !       of operator is to receive the request.  If the device is a
;  908          1804  1 !       tape, send the request to tape class operators.  If the
;  909          1805  1 !       device is a disk, send the request to disk class operators.
;  910          1806  1 !       If the device is neither tape or disk (ie. the user screwed
;  911          1807  1 !       up the device name on the command line) then send the
;  912          1808  1 !       request to both disk and tape class operators.  We remember
;  913          1809  1 !       the operator class mask in case we later have to cancel
;  914          1810  1 !       the request.
;  915          1811  1 !
;  916          1812  1 ! Input:
;  917          1813  1 !
;  918          1814  1 !       None.
;  919          1815  1 !
;  920          1816  1 ! Output:
;  921          1817  1 !
;  922          1818  1 !       None.
;  923          1819  1 !
;  924          1820  1 ! Implicit Input:
;  925          1821  1 !
;  926          1822  1 !       The MOUNT data base.  Note that:
;  927          1823  1 !       DEVICE_STRING[.DEVICE_INDEX*2] = the address of string descriptor
;  928          1824  1 !                                        of the device currently being mounted.
;  929          1825  1 !
;  930          1826  1 ! Implicit Output:
;  931          1827  1 !
;  932          1828  1 !       OPERATOR_MASK = mask of target operators.  Only
;  933          1829  1 !                       the low 3 bytes are significant.
;  934          1830  1 !
;  935          1831  1 !--
;  936          1832  1
;  937          1833  2 BEGIN                                           ! Start of SET_TARGET_MASK
;  938          1834  2
;  939          1835  2 EXTERNAL
;  940          1836  2         DEVICE_INDEX    : LONG VOLATILE,       ! Index into aforementioned vector
;  941          1837  2         PHYS_NAME       : VECTOR VOLATILE;     ! Vector of device descriptors
;  942          1838  2
;  943          1839  2 LOCAL
;  944          1840  2         DEVICE_CHAR     : BBLOCK [DIB$K_LENGTH],! Primary characteristics buffer
;  945          1841  2         DEVICE_CHAR2    : BBLOCK [DIB$K_LENGTH],! Secondary characteristics buffer
;  946          1842  2         DEVCHAR_DESC    : BBLOCK [DSC$K_S_BLN], ! Descriptor of primary char. buffer
;  947          1843  2         DEVCHAR_DESC2   : BBLOCK [DSC$K_S_BLN], ! Descriptor of secondary char. buffer
;  948          1844  2         STATUS          : LONG;
;  949          1845  2
;  950          1846  2 !
;  951          1847  2 ! Set up the device characteristic buffer descriptors.
;  952          1848  2 !
;  953          1849  2 DEVCHAR_DESC [DSC$W_LENGTH]  = DIB$K_LENGTH;
;  954          1850  2 DEVCHAR_DESC [DSC$B_DTYPE]   = DSC$K_DTYPE_T;
;  955          1851  2 DEVCHAR_DESC [DSC$B_CLASS]   = DSC$K_CLASS_S;
;  956          1852  2 DEVCHAR_DESC [DSC$A_POINTER] = DEVICE_CHAR;
;  957          1853  2 DEVCHAR_DESC2 [DSC$W_LENGTH]  = DIB$K_LENGTH;
```

```
;  958          1854  2 DEVCHAR_DESC2 [DSC$B_DTYPE]   = DSC$K_DTYPE_T;
;  959          1855  2 DEVCHAR_DESC2 [DSC$B_CLASS]   = DSC$K_CLASS_S;
;  960          1856  2 DEVCHAR_DESC2 [DSC$A_POINTER] = DEVICE_CHAR2;
;  961          1857  2 OPERATOR_MASK = 0;                              ! Zero the operator targt mask.
;  962          1858  2 !
;  963          1859  2 ! Get the device characteristics and perform some sanity checking.
;  964          1860  2 ! If this device is not mountable, don't worry, the operator will be
;  965          1861  2 ! notified and he'll think of something.
;  966          1862  2 !
;  967        P 1863  2 STATUS = $GETDEV (DEVNAM = PHYS_NAME [.DEVICE_INDEX *2],
;  968        P 1864  2                   PRIBUF=DEVCHAR_DESC,
;  969        P 1865  2                   SCDBUF = DEVCHAR_DESC2
;  970          1866  2                  );
;  971          1867  3 IF (NOT .DEVICE_CHAR[DEV$V_FOD]) OR  (.STATUS EQL SS$_NOSUCHDEV)
;  972          1868  2 THEN
;  973          1869  3     OPERATOR_MASK = (OPC$M_NM_DISKS OR OPC$M_NM_TAPES)  ! Send to tape and disk operators
;  974          1870  2 ELSE
;  975          1871  2     !
;  976          1872  2     ! Set the operator mask according to device class.  That is, tape
;  977          1873  2     ! requests go to TAPE operators, disk requests go to DISK operators.
;  978          1874  2     !
;  979          1875  3     OPERATOR_MASK = (IF .DEVICE_CHAR[DEV$V_SQD]
;  980          1876  3                      THEN
;  981          1877  3                          OPC$M_NM_TAPES
;  982          1878  3                      ELSE
;  983          1879  2                          OPC$M_NM_DISKS);
;  984          1880  1 END;                                             ! End of SET_TARGET_MASK


                                                .EXTRN  PHYS_NAME, SYS$GETDEV

                                0004 00000 SET_TARGET_MASK:
                                           .WORD   Save R2                                  ; 1797
                         52      0000'  CF  9E 00002       MOVAB   OPERATOR_MASK, R2        ;
                         5E      FF0C   CE  9E 00007       MOVAB   -244(SP), SP             ;
                      04 AE  010E0074   8F  D0 0000C       MOVL    #17694836, DEVCHAR_DESC  ; 1849
                      08 AE        8C   AD  9E 00014       MOVAB   DEVICE_CHAR, DEVCHAR_DESC+4 ; 1852
                             010E0074   8F  DD 00019       PUSHL   #17694836                ; 1853
                      04 AE        10   AE  9E 0001F       MOVAB   DEVICE_CHAR2, DEVCHAR_DESC2+4 ; 1856
                                        62  D4 00024       CLRL    OPERATOR_MASK            ; 1857
                                        5E  DD 00026       PUSHL   SP                       ; 1866
                                        7E  D4 00028       CLRL    -(SP)                    ;
                                     10 AE  9F 0002A       PUSHAB  DEVCHAR_DESC             ;
                                        7E  D4 0002D       CLRL    -(SP)                    ;
                50      0000G   CF          01  78 0002F       ASHL    #1, DEVICE_INDEX, R0     ;
                                     0000GCF40  DF 00035       PUSHAL  PHYS_NAME[R0]            ;
                   00000000G    00          05  FB 0003A       CALLS   #5, SYS$GETDEV           ;
                09       8D   AD          06  E1 00041       BBC     #6, DEVICE_CHAR+1, 1$    ; 1867
                   00000908   8F          50  D1 00046       CMPL    STATUS, #2312            ;
                                           04  12 0004D       BNEQ    2$                       ;
                              62           0C  D0 0004F 1$:    MOVL    #12, OPERATOR_MASK       ; 1869
                                           04  00052         RET                              ;
                05       8C   AD          05  E1 00053 2$:    BBC     #5, DEVICE_CHAR, 3$      ; 1875
                              50           04  D0 00058       MOVL    #4, R0                   ;
                                           03  11 0005B       BRB     4$                       ;
                              50           08  D0 0005D 3$:    MOVL    #8, R0                   ;
```

```
                          62          50  D0 00060 4$:    MOVL    R0, OPERATOR_MASK                       ;
                                          04 00063        RET                                             ; 1880

; Routine Size:  100 bytes,    Routine Base:  $CODE$ + 0356
```

```
;  986          1881  1 ROUTINE CANCEL_REQUEST (REQUEST_STATUS) : NOVALUE =
;  987          1882  1
;  988          1883  1 !++
;  989          1884  1 ! Functional Description:
;  990          1885  1 !
;  991          1886  1 !       This routine will cancel an outstanding operator request.
;  992          1887  1 !       The reply mailbox is deleted after the cancelation message
;  993          1888  1 !       is sent so there will be no stale messages lying around to
;  994          1889  1 !       confuse things later on.  The user is notified of the cancelation.
;  995          1890  1 !
;  996          1891  1 ! Input:
;  997          1892  1 !
;  998          1893  1 !       REQUEST_STATUS  : A boolean value that describes the status of the
;  999          1894  1 !                         operator request.  A value of 1 indicates the request
; 1000          1895  1 !                         has been successfully completed without  operator
; 1001          1896  1 !                         intervention, and the reason for the request no
; 1002          1897  1 !                         longer exists.  A value of 0 indicates that the
; 1003          1898  1 !                         request has not been satisfied, but is being canceled
; 1004          1899  1 !                         for some reason.
; 1005          1900  1 !
; 1006          1901  1 ! Output:
; 1007          1902  1 !
; 1008          1903  1 !       None.
; 1009          1904  1 !
; 1010          1905  1 ! Implicit Input:
; 1011          1906  1 !
; 1012          1907  1 !       REPLY_PENDING = TRUE if there is an outstanding operater request.
; 1013          1908  1 !
; 1014          1909  1 ! Implicit Outputs:
; 1015          1910  1 !
; 1016          1911  1 !       REPLY_PENDING = FALSE
; 1017          1912  1 !--
; 1018          1913  1
; 1019          1914  2 BEGIN                                           ! Start of CANCEL_REQUEST
; 1020          1915  2
; 1021          1916  2
; 1022          1917  2 IF .REPLY_PENDING
; 1023          1918  2 THEN
; 1024          1919  3     BEGIN
; 1025          1920  3     !
; 1026          1921  3     ! Send cancelation notice to operator
; 1027          1922  3     !
; 1028          1923  3     BBLOCK [CANCEL_MSG_BUF [OPC$L_RQ_OPTIONS], OPC$V_RQSTDONE] = .REQUEST_STATUS;
; 1029          1924  3     CANCEL_MSG_BUF[OPC$L_RQSTID] = .REQUEST_ID;
; 1030          1925  3     CANCEL_MSG_BUF[OPC$L_ATTNMASK1] = .OPERATOR_MASK;
; 1031          1926  3     $SNDOPR (MSGBUF=CANCEL_MSG_DESC, CHAN=.REPLY_CHANNEL);
; 1032          1927  3     !
; 1033          1928  3     ! Deassign the channel to the reply mailbox.  Since it
; 1034          1929  3     ! is a temporary mailbox, it will be deleted.
; 1035          1930  3     !
; 1036          1931  3     $DASSGN (CHAN = .REPLY_CHANNEL);
; 1037          1932  3     REPLY_CHANNEL = 0;
; 1038          1933  3     REPLY_PENDING = FALSE;
; 1039          1934  3     !
; 1040          1935  3     ! Clear the reply event flag.
; 1041          1936  3     !
; 1042          1937  3     $CLREF (EFN=REPLY_FLAG);
```

```
;  1043          1938  3      !
;  1044          1939  3      ! Notify the user of the cancelation.
;  1045          1940  3      !
;  1046          1941  4      IF .REQUEST_STATUS AND (NOT .MOUNT_FAILED)
;  1047          1942  4      THEN
;  1048          1943  5          SIGNAL (MOUN$_RQSTDON)
;  1049          1944  4      ELSE
;  1050          1945  5          SIGNAL (MOUN$_OPRQSTCAN);
;  1051          1946  2      END;
;  1052          1947  2
;  1053          1948  1 END;                                    ! End of CANCEL_REQUEST


                                                               .EXTRN  SYS$DASSGN, SYS$CLREF

                                                0004 00000 CANCEL_REQUEST:
                                                               .WORD   Save R2                                         ; 1881
                                      52     0000'  CF  9E 00002      MOVAB   REPLY_CHANNEL, R2
                                      55       C8   A2  E9 00007      BLBC    REPLY_PENDING, 3$                       ; 1917
    0156   C2              01                00       04   AC  F0 0000B      INSV    REQUEST_STATUS, #0, #1, CANCEL_MSG_BUF+6  ; 1923
                                0162   C2       EC   A2  D0 00013      MOVL    REQUEST_ID, CANCEL_MSG_BUF+18           ; 1924
                                015A   C2       E8   A2  D0 00019      MOVL    OPERATOR_MASK, CANCEL_MSG_BUF+10        ; 1925
                                               62   DD  0001F      PUSHL   REPLY_CHANNEL                           ; 1926
                                       016C    C2   9F  00021      PUSHAB  CANCEL_MSG_DESC
                          00000000G    00             02   FB  00025      CALLS   #2, SYS$SNDOPR
                                               62   DD  0002C      PUSHL   REPLY_CHANNEL                           ; 1931
                          00000000G    00             01   FB  0002E      CALLS   #1, SYS$DASSGN
                                               62   D4  00035      CLRL    REPLY_CHANNEL                           ; 1932
                                         C8    A2   D4  00037      CLRL    REPLY_PENDING                           ; 1933
                                               1A   DD  0003A      PUSHL   #26                                     ; 1937
                          00000000G    00             01   FB  0003C      CALLS   #1, SYS$CLREF
                                       0C        04   AC  E9 00043      BLBC    REQUEST_STATUS, 1$                      ; 1941
                                       08        CC   A2  E8 00047      BLBS    MOUNT_FAILED, 1$
                                          0072A073   8F  DD  0004B      PUSHL   #7512T79                                ; 1943
                                               06   11  00051      BRB     2$
                                          0072A033   8F  DD  00053 1$:  PUSHL   #7512115                               ; 1945
                          00000000G    00             01   FB  00059 2$:  CALLS   #1, LIB$SIGNAL
                                                    04   00060 3$:  RET                                             ; 1948

; Routine Size:  97 bytes,    Routine Base:  $CODE$ + 03BA
```

```
;  1055     1949  1 ROUTINE CHECK_FOR_REPLY : NOVALUE =
;  1056     1950  1
;  1057     1951  1 !++
;  1058     1952  1 ! Functional Description:
;  1059     1953  1 !
;  1060     1954  1 !       This routine will check to see if the operator
;  1061     1955  1 !       replied to a request after DELTA_TIME expired.
;  1062     1956  1 !       If so, the response must be parsed and acted upon.
;  1063     1957  1 !       Note that this might require undoing a successful mount.
;  1064     1958  1 !       If the request is still outstanding and the mount
;  1065     1959  1 !       completed successfully, then cancel the request.
;  1066     1960  1 !
;  1067     1961  1 ! Input:
;  1068     1962  1 !
;  1069     1963  1 !       WAIT_ENABLED = TRUE if we are to wait, FALSE if not.
;  1070     1964  1 !
;  1071     1965  1 ! Output:
;  1072     1966  1 !
;  1073     1967  1 !       None.
;  1074     1968  1 !
;  1075     1969  1 ! Implicit Inputs:
;  1076     1970  1 !
;  1077     1971  1 !       REPLY_PENDING = 1 if there is an outstanding request.
;  1078     1972  1 !       REPLY_DESC = string descriptor of the operator's reply.
;  1079     1973  1 !       REPLY_BUFFER = buffer holding the operator's reply.
;  1080     1974  1 !       MOUNT data base.
;  1081     1975  1 !
;  1082     1976  1 ! Implicit Outputs:
;  1083     1977  1 !
;  1084     1978  1 !       The MOUNT data base may be updated as a result of the operator's reply.
;  1085     1979  1 !--
;  1086     1980  1
;  1087     1981  2 BEGIN                                           ! Start of CHECK_FOR_REPLY
;  1088     1982  2
;  1089     1983  2 LOCAL
;  1090     1984  2
;  1091     1985  2         EF_STATE        : LONG,                 ! State of Event flags
;  1092     1986  2         STATUS          : LONG;
;  1093     1987  2
;  1094     1988  2 IF NOT .MOUNT_FAILED
;  1095     1989  2 THEN
;  1096     1990  2     !
;  1097     1991  2     ! The mount succeeded.  Operator intervention is
;  1098     1992  2     ! no longer necessary, so cancel the request.
;  1099     1993  2     !
;  1100     1994  2     CANCEL_REQUEST (REQUEST_SATISFIED)
;  1101     1995  2 ELSE
;  1102     1996  3     BEGIN
;  1103     1997  3     !
;  1104     1998  3     ! The mount failed (again).
;  1105     1999  3     !
;  1106     2000  3     ! If a reply was pending, wait for either the timer to go off or
;  1107     2001  3     ! for the reply to arrive, whichever comes first.  If no reply is
;  1108     2002  3     ! pending, then simply wait for the timer to go off.  Cancel the
;  1109     2003  3     ! timer on the way out, just to be thourough.
;  1110     2004  3     !
;  1111     2005  3     ! If no operator is present, only attempt to read the reply mailbox
```

```
;  1112          2006  3     ! every tenth time through this routine.  This is necessaray to prevent
;  1113          2007  3     ! prevent mount from looping rapidly through this code.
;  1114          2008  3     !
;  1115          2009  4     IF NOT (STATUS = $SETIMR (EFN=TIMER_FLAG, REQIDT=TIMER_ID, DAYTIM=DELTA_TIME))
;  1116          2010  3     THEN
;  1117          2011  3         ABORT_MOUNT (.STATUS, 0, .MOUNT_STATUS);
;  1118          2012  3
;  1119          2013  4     IF (.REPLY_PENDING AND .OPERATOR_PRESENT)
;  1120          2014  4     OR ((NOT .OPERATOR_PRESENT) AND (.RETRY_COUNTER/10) GEQ 1)
;  1121          2015  3     THEN
;  1122          2016  4         BEGIN
;  1123          2017  4         RETRY_COUNTER = 0;
;  1124          2018  5         IF (.REPLY_IOSB [0,0,16,0] NEQ 0)
;  1125          2019  4         THEN
;  1126          2020  4             PARSE_REPLY ()
;  1127          2021  4         ELSE
;  1128          2022  4             $WAITFR (EFN = TIMER_FLAG);
;  1129          2023  4         END
;  1130          2024  3     ELSE
;  1131          2025  3         $WAITFR (EFN = TIMER_FLAG);
;  1132          2026  3
;  1133          2027  3     $CANTIM (REQIDT = TIMER_ID);                ! Cancel the timer
;  1134          2028  3     $SETEF (EFN = TIMER_FLAG);                  ! Set timer flag
;  1135          2029  2     END;
;  1136          2030  2 RETRY_COUNTER = .RETRY_COUNTER + 1;
;  1137          2031  1 END;                                            ! End of CHECK_FOR_REPLY



                                0004 00000 CHECK_FOR_REPLY:
                                           .WORD   Save R2                                          ; 1949
                    52    0000'  CF  9E 00002    MOVAB   RETRY_COUNTER, R2
                    08      F8   A2  E8 00007    BLBS    MOUNT_FAILED, 1$                         ; 1988
                                 01  DD 0000B    PUSHL   #1                                       ; 1994
                 8E AF           01  FB 0000D    CALLS   #1, CANCEL_REQUEST
                                 65  11 00011    BRB     7$
                    7E    03E7   8F  3C 00013 1$: MOVZWL  #999, -(SP)                              ; 2009
                                 7E  D4 00018    CLRL    -(SP)
                          0000'  CF  9F 0001A    PUSHAB  DELTA_TIME
                                 19  DD 0001E    PUSHL   #25
             00000000G    00          04  FB 00020    CALLS   #4, SYS$SETIMR
                          0E          50  E8 00027    BLBS    STATUS, 2$
                            08        A2  DD 0002A    PUSHL   MOUNT_STATUS                         ; 2011
                                      7E  D4 0002D    CLRL    -(SP)
                                      50  DD 0002F    PUSHL   STATUS
             00000000G    00          03  FB 00031    CALLS   #3, LIB$STOP
                          04      F4  A2  E9 00038 2$: BLBC    REPLY_PENDING, 3$                   ; 2013
                          0A      FC  A2  E8 0003C    BLBS    OPERATOR_PRESENT, 4$
                          14      FC  A2  E8 00040 3$: BLBS    OPERATOR_PRESENT, 5$                ; 2014
              50          62          0A  C7 00044    DIVL3   #10, RETRY_COUNTER, R0
                                      0E  15 00048    BLEQ    5$
                                      62  D4 0004A 4$: CLRL    RETRY_COUNTER                       ; 2017
                            30        A2  B5 0004C    TSTW    REPLY_IOSB                          ; 2018
                                      07  13 0004F    BEQL    5$
                 0000V    CF              00  FB 00051    CALLS   #0, PARSE_REPLY                 ; 2020
```

```
                                        09  11 00056           BRB     6$                                      ;
                                        19  DD 00058 5$:       PUSHL   #25                                     ; 2025
                    00000000G   00      01  FB 0005A           CALLS   #1, SYS$WAITFR                          ;
                                        7E  D4 00061 6$:       CLRL    -(SP)                                   ; 2027
                                7E  03E7    8F  3C 00063       MOVZWL  #999, -(SP)                             ;
                    00000000G   00      02  FB 00068           CALLS   #2, SYS$CANTIM                          ;
                                        19  DD 0006F           PUSHL   #25                                     ; 2028
                    00000000G   00      01  FB 00071           CALLS   #1, SYS$SETEF                           ;
                                        62  D6 00078 7$:       INCL    RETRY_COUNTER                           ; 2030
                                            04 0007A           RET                                             ; 2031

; Routine Size:  123 bytes,    Routine Base:  $CODE$ + 041B
```

```
;  1139     2032  1 ROUTINE ALLOCFAIL_HNDLR : NOVALUE =
;  1140     2033  1
;  1141     2034  1 !++
;  1142     2035  1 ! Functional Description:
;  1143     2036  1 !
;  1144     2037  1 !       This routine will attempt to recover from a device
;  1145     2038  1 !       allocation failure.  This means that the device
;  1146     2039  1 !       specified by the user (or operator) cannot be
;  1147     2040  1 !       successfully allocated.  Notify the operator and
;  1148     2041  1 !       try again.  Current allocation failures handled are:
;  1149     2042  1 !
;  1150     2043  1 !               SS$_DEVALLOC    - device allocated to another user
;  1151     2044  1 !               SS$_NODEVAVL    - no devices of generic type are available
;  1152     2045  1 !               SS$_NOSUCHDEV   - incorrect device specifier
;  1153     2046  1 !
;  1154     2047  1 ! Input:
;  1155     2048  1 !
;  1156     2049  1 !       None.
;  1157     2050  1 !
;  1158     2051  1 ! Output:
;  1159     2052  1 !
;  1160     2053  1 !       None.
;  1161     2054  1 !
;  1162     2055  1 ! Implicit Input:
;  1163     2056  1 !
;  1164     2057  1 !       MOUNT_STATUS  = status of current mount attempt
;  1165     2058  1 !       REPLY_PENDING = TRUE if an operator request is outstanding
;  1166     2059  1 !       The MOUNT data base.
;  1167     2060  1 !
;  1168     2061  1 ! Implict Output:
;  1169     2062  1 !
;  1170     2063  1 !       The MOUNT data base may be changed as
;  1171     2064  1 !       the result of operator intervention.
;  1172     2065  1 !--
;  1173     2066  1
;  1174     2067  2 BEGIN                                           ! Start of ALLOCFAIL_HNDLR
;  1175     2068  2
;  1176     2069  2 EXTERNAL
;  1177     2070  2
;  1178     2071  2         COMMENT_STRING  : BBLOCK,               ! User comment string
;  1179     2072  2         DEVICE_INDEX    : LONG VOLATILE,        ! Index into device name vector
;  1180     2073  2         PHYS_NAME       : VECTOR VOLATILE;      ! Physical device name descriptor
;  1181     2074  2 LITERAL
;  1182     2075  2
;  1183     2076  2         FAO_CTRL_SIZ    = FAO_BUFFER_SIZE/2;    ! Maximum size for FAO control string
;  1184     2077  2
;  1185     2078  2 LOCAL
;  1186     2079  2
;  1187     2080  2         ALLOCFAIL_FAO   : BBLOCK [DSC$K_S_BLN], ! FAO control string descriptor
;  1188     2081  2         FAO_CTRL_BUF    : BBLOCK [FAO_CTRL_SIZ],! Buffer for FAO control string
;  1189     2082  2         STATUS          : LONG;
;  1190     2083  2
;  1191     2084  2
;  1192     2085  2 !
;  1193     2086  2 ! If this condition is different from the one signaled previously,
;  1194     2087  2 ! cancel any outstanding requests before handling this condition.
;  1195     2088  2 ! Otherwise do nothing.
```

```
;  1196          2089  2 !
;  1197          2090  3 IF (.MOUNT_STATUS NEQ .PREVIOUS_STATUS)
;  1198          2091  3 OR (.DEVICE_INDEX NEQ .PREVIOUS_DEV_IDX)
;  1199          2092  2 THEN
;  1200          2093  3     BEGIN
;  1201          2094  3     CANCEL_REQUEST (REQUEST_NOT_SATISFIED);
;  1202          2095  3     OPERATOR_PRESENT = TRUE;                    ! Assume operator present
;  1203          2096  3     !
;  1204          2097  3     ! Set up the output descriptor and get the FAO control string.
;  1205          2098  3     !
;  1206          2099  3     ALLOCFAIL_FAO [DSC$W_LENGTH]  = FAO_CTRL_SIZ;
;  1207          2100  3     ALLOCFAIL_FAO [DSC$B_DTYPE]   = DSC$K_DTYPE_T;
;  1208          2101  3     ALLOCFAIL_FAO [DSC$B_CLASS]   = DSC$K_CLASS_S;
;  1209          2102  3     ALLOCFAIL_FAO [DSC$A_POINTER] = FAO_CTRL_BUF;
;  1210        P 2103  4     IF NOT (STATUS = $GETMSG    (MSGID  = MOUN$_NODEVAVL,
;  1211        P 2104  4                                 MSGLEN = ALLOCFAIL_FAO [DSC$W_LENGTH],
;  1212        P 2105  4                                 BUFADR = ALLOCFAIL_FAO,
;  1213        P 2106  4                                 FLAGS  = MSG_TEXT
;  1214          2107  4                                ))
;  1215          2108  3     THEN
;  1216          2109  3         ABORT_MOUNT (.STATUS, 0, .MOUNT_STATUS);
;  1217          2110  3     !
;  1218          2111  3     ! Set up the output descriptor and format the operator request.
;  1219          2112  3     !
;  1220          2113  3     FAO_RESULT_DESC[DSC$A_POINTER] = FAO_BUFFER;
;  1221          2114  3     FAO_RESULT_DESC[DSC$W_LENGTH]  = FAO_BUFFER_SIZE;
;  1222        P 2115  3     $FAO (ALLOCFAIL_FAO,
;  1223        P 2116  3           FAO_RESULT_DESC [DSC$W_LENGTH],
;  1224        P 2117  3           FAO_RESULT_DESC,
;  1225        P 2118  3           PHYS_NAME [.DEVICE_INDEX*2],
;  1226        P 2119  3           COMMENT_STRING
;  1227          2120  3          );
;  1228          2121  3     !
;  1229          2122  3     ! Send the request to the operator.
;  1230          2123  3     !
;  1231          2124  3     SUBMIT_REQUEST (FAO_RESULT_DESC,EXPECT_REPLY);
;  1232          2125  2     END;
;  1233          2126  2
;  1234          2127  1 END;                                          ! End of ALLOCFAIL_HNDLR


                                                    .EXTRN  COMMENT_STRING, SYS$GETMSG
                                                    .EXTRN  SYS$FAO

                                        0004 00000 ALLOCFAIL_HNDLR:
                                                    .WORD   Save R2                                 ; 2032
                              52      0000'  CF  9E 00002           MOVAB   FAO_RESULT_DESC, R2
                              5E      FEF8   CE  9E 00007           MOVAB   -264(SP), SP
                        FC6C  C2      FC68   C2  D1 0000C           CMPL    MOUNT_STATUS, PREVIOUS_STATUS   ; 2090
                                             09  12 00013           BNEQ    1$
                        FC70  C2      0000G  CF  D1 00015           CMPL    DEVICE_INDEX, PREVIOUS_DEV_IDX  ; 2091
                                             71  13 0001C           BEQL    3$
                                             7E  D4 0001E 1$:      CLRL    -(SP)                           ; 2094
                        FEFF  CF              01  FB 00020           CALLS   #1, CANCEL_REQUEST
                        FC5C  C2              01  D0 00025           MOVL    #1, OPERATOR_PRESENT            ; 2095
                          F8  AD 010E0100     8F  D0 0002A           MOVL    #17694976, ALLOCFAIL_FAO        ; 2099
```

```
                          FC    AD              6E  9E 00032          MOVAB   FAO_CTRL_BUF, ALLOCFAIL_FAO+4          ; 2102
                                7E              01  7D 00036          MOVQ    #1, -(SP)                             ; 2107
                                          F8    AD  9F 00039          PUSHAB  ALLOCFAIL_FAO                         ;
                                          F8    AD  9F 0003C          PUSHAB  ALLOCFAIL_FAO                         ;
                                   0072A05B     8F  DD 0003F          PUSHL   #7512155                              ;
                    00000000G     00                05  FB 00045          CALLS   #5, SYS$GETMSG                        ;
                                  0F                50  E8 0004C          BLBS    STATUS, 2$                            ;
                                          FC68  C2  DD 0004F          PUSHL   MOUNT_STATUS                          ; 2109
                                                7E  D4 00053          CLRL    -(SP)                                 ;
                                                50  DD 00055          PUSHL   STATUS                                ;
                    00000000G     00                03  FB 00057          CALLS   #3, LIB$STOP                          ;
                          04      A2        FE00    C2  9E 0005E  2$:     MOVAB   FAO_BUFFER, FAO_RESULT_DESC+4         ; 2113
                                  62        0200    8F  B0 00064          MOVW    #512, FAO_RESULT_DESC                 ; 2114
                                            0000G   CF  9F 00069          PUSHAB  COMMENT_STRING                        ; 2120
                    50      0000G     CF              01  78 0006D          ASHL    #1, DEVICE_INDEX, R0                  ;
                                            0000GCF40   DF 00073          PUSHAL  PHYS_NAME[R0]                         ;
                                                52  DD 00078          PUSHL   R2                                    ;
                                                52  DD 0007A          PUSHL   R2                                    ;
                                          F8    AD  9F 0007C          PUSHAB  ALLOCFAIL_FAO                         ;
                    00000000G     00                05  FB 0007F          CALLS   #5, SYS$FAO                           ;
                                                01  DD 00086          PUSHL   #1                                    ; 2124
                                                52  DD 00088          PUSHL   R2                                    ;
                          FD00    CF              02  FB 0008A          CALLS   #2, SUBMIT_REQUEST                    ;
                                                    04  0008F  3$:     RET                                           ; 2127

; Routine Size:  144 bytes,    Routine Base:  $CODE$ + 0496
```

```
;  1236          2128  1 ROUTINE MEDOFL_HNDLR : NOVALUE =
;  1237          2129  1
;  1238          2130  1 !++
;  1239          2131  1 ! Functional Description:
;  1240          2132  1 !
;  1241          2133  1 !       This routine will attempt to recover from a medium
;  1242          2134  1 !       offline condition.  This usually means that the disk is
;  1243          2135  1 !       not spun up.  Notify the operator that the device
;  1244          2136  1 !       needs to be put online.
;  1245          2137  1 !
;  1246          2138  1 ! Input:
;  1247          2139  1 !
;  1248          2140  1 !       None.
;  1249          2141  1 !
;  1250          2142  1 ! Output:
;  1251          2143  1 !
;  1252          2144  1 !       None.
;  1253          2145  1 !
;  1254          2146  1 ! Implicit Input:
;  1255          2147  1 !
;  1256          2148  1 !       MOUNT_STATUS  = status of the current mount attempt
;  1257          2149  1 !       REPLY_PENDING = TRUE if an operator request is outstanding
;  1258          2150  1 !       The MOUNT data base.
;  1259          2151  1 !
;  1260          2152  1 ! Implict Output:
;  1261          2153  1 !
;  1262          2154  1 !       The MOUNT data base may be changed as
;  1263          2155  1 !       the result of operator intervention.
;  1264          2156  1 !--
;  1265          2157  1
;  1266          2158  2 BEGIN                                           ! Start of MEDOFL_HNDLR
;  1267          2159  2
;  1268          2160  2 EXTERNAL
;  1269          2161  2
;  1270          2162  2         COMMENT_STRING  : BBLOCK,              ! User comment string
;  1271          2163  2         LABEL_STRING    : VECTOR VOLATILE,     ! Vector of label descriptors
;  1272          2164  2         PHYS_NAME       : VECTOR VOLATILE,     ! Physical device name descriptor
;  1273          2165  2         DEVICE_INDEX    : LONG VOLATILE;       ! Index into DEVICE_STRING vector
;  1274          2166  2
;  1275          2167  2 LITERAL
;  1276          2168  2
;  1277          2169  2         FAO_CTRL_SIZ    = FAO_BUFFER_SIZE/2;   ! FAO control string size
;  1278          2170  2
;  1279          2171  2 LOCAL
;  1280          2172  2
;  1281          2173  2         MEDOFL_FAO      : BBLOCK [DSC$K_S_BLN],
;  1282          2174  2         MEDOFL_BUF      : BBLOCK [FAO_CTR[_SIZ],
;  1283          2175  2         VOLUME_FAO      : BBLOCK [DSC$K_S_BLN],
;  1284          2176  2         VOLUME_BUF      : BBLOCK [FAO_CTR[_SIZ],
;  1285          2177  2         VOLUME_DESC     : BBLOCK [DSC$K_S_BLN],
;  1286          2178  2         VOLUME_BUFFER   : BBLOCK [FAO_CTR[_SIZ],
;  1287          2179  2         STATUS          : LONG;
;  1288          2180  2
;  1289          2181  2
;  1290          2182  2
;  1291          2183  2 !
;  1292          2184  2 ! If this condition is different from the one signaled previously,
```

```
;  1293          2185  2 ! cancel any outstanding requests before handling this condition.
;  1294          2186  2 ! Note that if the previous condition was SS$_INCVOLLABEL, we do
;  1295          2187  2 ! not cancel the request and issue another one.  This is to give
;  1296          2188  2 ! the operator a chance to remove the incorrect volume from the drive
;  1297          2189  2 ! and to (hopefully) insert the correct volume.
;  1298          2190  2 !
;  1299          2191  3 IF  ((.MOUNT_STATUS AND STS$M_COND_ID) NEQ (SS$_INCVOLLABEL AND STS$M_COND_ID))
;  1300          2192  3 AND ((.PREVIOUS_STATUS AND STS$M_COND_ID) EQL (SS$_INCVOLLABEL AND STS$M_COND_ID))
;  1301          2193  3 AND (.DEVICE_INDEX  EQL .PREVIOUS_DEV_IDX)
;  1302          2194  2 THEN
;  1303          2195  3     BEGIN
;  1304          2196  3     PREVIOUS_STATUS = .MOUNT_STATUS;
;  1305          2197  2     END;
;  1306          2198  2
;  1307          2199  3 IF (.DEVICE_INDEX NEQ .PREVIOUS_DEV_IDX)
;  1308          2200  3 OR (.MOUNT_STATUS NEQ .PREVIOUS_STATUS)
;  1309          2201  2 THEN
;  1310          2202  3     BEGIN
;  1311          2203  3     CANCEL_REQUEST (REQUEST_NOT_SATISFIED);
;  1312          2204  3     OPERATOR_PRESENT = TRUE;                    ! Assume operator present
;  1313          2205  2     END;
;  1314          2206  2 !
;  1315          2207  2 ! If there is no outstanding request, then submit a request.
;  1316          2208  2 !
;  1317          2209  2 IF NOT .REPLY_PENDING
;  1318          2210  2 THEN
;  1319          2211  3     BEGIN
;  1320          2212  3     !
;  1321          2213  3     ! Set up the output descriptor and format the volume label string.
;  1322          2214  3     !
;  1323          2215  3     VOLUME_DESC [DSC$W_LENGTH]  = FAO_CTRL_SIZ;
;  1324          2216  3     VOLUME_DESC [DSC$B_DTYPE]   = DSC$K_DTYPE_T;
;  1325          2217  3     VOLUME_DESC [DSC$B_CLASS]   = DSC$K_CLASS_S;
;  1326          2218  3     VOLUME_DESC [DSC$A_POINTER] = VOLUME_BUFFER;
;  1327          2219  3     IF .LABEL_STRING[.DEVICE_INDEX*2] GTR 0
;  1328          2220  3     THEN
;  1329          2221  4         BEGIN
;  1330          2222  4         !
;  1331          2223  4         ! Set up the output descriptor and get the FAO control string.
;  1332          2224  4         !
;  1333          2225  4         VOLUME_FAO [DSC$W_LENGTH] = FAO_CTRL_SIZ;
;  1334          2226  4         VOLUME_FAO [DSC$B_DTYPE]  = DSC$K_DTYPE_T;
;  1335          2227  4         VOLUME_FAO [DSC$B_CLASS]  = DSC$K_CLASS_S;
;  1336          2228  4         VOLUME_FAO [DSC$A_POINTER] = VOLUME_BUF;
;  1337        P 2229  5         IF NOT (STATUS = $GETMSG (MSGID  = MOUN$_VOLNAME,
;  1338        P 2230  5                                 MSGLEN = VOLUME_FAO [DSC$W_LENGTH],
;  1339        P 2231  5                                 BUFADR = VOLUME_FAO,
;  1340        P 2232  5                                 FLAGS  = MSG_TEXT
;  1341          2233  5                                ))
;  1342          2234  4         THEN
;  1343          2235  4             ABORT_MOUNT (.STATUS, 0, .MOUNT_STATUS);
;  1344          2236  4         !
;  1345          2237  4         ! Format the volume label string,
;  1346          2238  4         !
;  1347        P 2239  4         $FAO    (VOLUME_FAO,
;  1348        P 2240  4                  VOLUME_DESC [DSC$W_LENGTH],
;  1349        P 2241  4                  VOLUME_DESC,
```

```
;  1350        P 2242  4                    LABEL_STRING [.DEVICE_INDEX*2]
;  1351          2243  4                   );
;  1352          2244  4        END
;  1353          2245  3    ELSE
;  1354          2246  3        VOLUME_DESC [DSC$W_LENGTH] = 0;          ! Set volume name null
;  1355          2247  3    !
;  1356          2248  3    ! Set up the desctiptors and get the FAO control string for the message.
;  1357          2249  3    !
;  1358          2250  3    MEDOFL_FAO [DSC$W_LENGTH]   = FAO_CTRL_SIZ;
;  1359          2251  3    MEDOFL_FAO [DSC$B_DTYPE]    = DSC$K_DTYPE_T;
;  1360          2252  3    MEDOFL_FAO [DSC$B_CLASS]    = DSC$K_CLASS_S;
;  1361          2253  3    MEDOFL_FAO [DSC$A_POINTER]  = MEDOFL_BUF;
;  1362        P 2254  3    $GETMSG     (MSGID  = MOUN$_MOUNTDEV,
;  1363        P 2255  3                 MSGLEN = MEDOFL_FAO [DSC$W_LENGTH],
;  1364        P 2256  3                 BUFADR = MEDOFL_FAO,
;  1365        P 2257  3                 FLAGS  = MSG_TEXT
;  1366          2258  3                );
;  1367          2259  3    !
;  1368          2260  3    ! Set up the output descriptor and format the operator request.
;  1369          2261  3    !
;  1370          2262  3    FAO_RESULT_DESC [DSC$W_LENGTH]  = FAO_BUFFER_SIZE;
;  1371          2263  3    FAO_RESULT_DESC [DSC$A_POINTER] = FAO_BUFFER;
;  1372        P 2264  3    $FAO (MEDOFL_FAO,
;  1373        P 2265  3          FAO_RESULT_DESC[DSC$W_LENGTH],
;  1374        P 2266  3          FAO_RESULT_DESC,
;  1375        P 2267  3          VOLUME_DESC,
;  1376        P 2268  3          PHYS_NAME [.DEVICE_INDEX*2],
;  1377        P 2269  3          COMMENT_STRING
;  1378          2270  3         );
;  1379          2271  3    !
;  1380          2272  3    ! Send the request to the operator.
;  1381          2273  3    !
;  1382          2274  3    SUBMIT_REQUEST (FAO_RESULT_DESC,EXPECT_REPLY);
;  1383          2275  2    END;
;  1384          2276  2
;  1385          2277  1 END;                                    ! End of MEDOFL_HNDLR


                                                    .EXTRN  LABEL_STRING

                                        003C 00000 MEDOFL_HNDLR:
                                                    .WORD   Save R2,R3,R4,R5                    ; 2128
                          55 00000000G   00  9E 00002       MOVAB   SYS$FAO, R5
                          54 00000000G   00  9E 00009       MOVAB   SYS$GETMSG, R4
                          53      0000G  CF  9E 00010       MOVAB   DEVICE_INDEX, R3
                          52      0000'  CF  9E 00015       MOVAB   MOUNT_STATUS, R2
                          5E      FCE8   CE  9E 0001A       MOVAB   -792(SP), SP
                50        62 F0000007   8F  CB 0001F       BICL3   #-268435449, MOUNT_STATUS, R0   ; 2191
                   00000108   8F              50  D1 00027       CMPL    R0, #264
                                              1C  13 0002E       BEQL    1$
                50        04   A2 F0000007   8F  CB 00030       BICL3   #-268435449, PREVIOUS_STATUS, R0 ; 2192
                   00000108   8F              50  D1 00039       CMPL    R0, #264
                                              0A  12 00040       BNEQ    1$
                         08   A2              63  D1 00042       CMPL    DEVICE_INDEX, PREVIOUS_DEV_IDX  ; 2193
                                              04  12 00046       BNEQ    1$
                         04   A2              62  D0 00048       MOVL    MOUNT_STATUS, PREVIOUS_STATUS   ; 2196
```

```
                08   A2              63   D1 0004C 1$:    CMPL    DEVICE_INDEX, PREVIOUS_DEV_IDX                  ; 2199
                                     06   12 00050        BNEQ    2$
                04   A2              62   D1 00052        CMPL    MOUNT_STATUS, PREVIOUS_STATUS                   ; 2200
                                     0B   13 00056        BEQL    3$
                                     7E   D4 00058 2$:     CLRL    -(SP)                                           ; 2203
              FE35   CF              01   FB 0005B        CALLS   #1, CANCEL_REQUEST
                F4   A2              01   D0 0005F        MOVL    #1, OPERATOR_PRESENT                            ; 2204
                     01        EC    A2   E9 00063 3$:     BLBC    REPLY_PENDING, 4$                               ; 2209
                                     04   00067           RET
              0100   CE 010E0100     8F   D0 00068 4$:     MOVL    #17694976, VOLUME_DESC                          ; 2215
              0104   CE              6E   9E 00071        MOVAB   VOLUME_BUFFER, VOLUME_DESC+4                    ; 2218
        50          63              01   78 00076        ASHL    #1, DEVICE_INDEX, R0                            ; 2219
                          0000GCF40  D5   0007A           TSTL    LABEL_STRING[R0]
                                     4E   15 0007F        BLEQ    6$
              FEF0   CD 010E0100     8F   D0 00081        MOVL    #17694976, VOLUME_FAO                           ; 2225
              FEF4   CD      0108    CE   9E 0008A        MOVAB   VOLUME_BUF, VOLUME_FAO+4                        ; 2228
                     7E              01   7D 00091        MOVQ    #1, -(SP)                                       ; 2233
                               FEF0  CD   9F 00094        PUSHAB  VOLUME_FAO
                               FEF0  CD   9F 00098        PUSHAB  VOLUME_FAO
                          0072A053   8F   DD 0009C        PUSHL   #7512147
                               64          05   FB 000A2        CALLS   #5, SYS$GETMSG
                               0D          50   E8 000A5        BLBS    STATUS, 5$
                                           62   DD 000A8        PUSHL   MOUNT_STATUS                            ; 2235
                                           7E   D4 000AA        CLRL    -(SP)
                                           50   DD 000AC        PUSHL   STATUS
           000000000G          00          03   FB 000AF        CALLS   #3, LIB$STOP
        50          63              01   78 000B5 5$:     ASHL    #1, DEVICE_INDEX, R0                            ; 2243
                          0000GCF40  DF   000B9           PUSHAL  LABEL_STRING[R0]
                               0104  CE   9F 000BE        PUSHAB  VOLUME_DESC
                               0108  CE   9F 000C2        PUSHAB  VOLUME_DESC
                               FEF0  CD   9F 000C6        PUSHAB  VOLUME_FAO
                               65          04   FB 000CA        CALLS   #4, SYS$FAO
                                           04   11 000CD        BRB     7$                                      ; 2219
                          0100   CE   B4   000CF 6$:     CLRW    VOLUME_DESC                                     ; 2246
                    F8    AD 010E0100     8F   D0 000D3 7$:     MOVL    #17694976, MEDOFL_FAO                   ; 2250
                    FC    AD      FEF8    CD   9E 000DB        MOVAB   MEDOFL_BUF, MEDOFL_FAO+4                ; 2253
                          7E              01   7D 000E1        MOVQ    #1, -(SP)                               ; 2258
                                    F8    AD   9F 000E4        PUSHAB  MEDOFL_FAO
                                    F8    AD   9F 000E7        PUSHAB  MEDOFL_FAO
                               0072A04B   8F   DD 000EA        PUSHL   #7512139
                               64          05   FB 000F0        CALLS   #5, SYS$GETMSG
                    0398   C2        0200  8F   B0 000F3        MOVW    #512, FAO_RESULT_DESC                   ; 2262
                    039C   C2        0198  C2   9E 000FA        MOVAB   FAO_BUFFER, FAO_RESULT_DESC+4           ; 2263
                                      0000G CF   9F 00101        PUSHAB  COMMENT_STRING                          ; 2270
                  50          63              01   78 00105        ASHL    #1, DEVICE_INDEX, R0
                                    0000GCF40  DF   00109        PUSHAL  PHYS_NAME[R0]
                                    0108   CE   9F 0010E        PUSHAB  VOLUME_DESC
                                    0398   C2   9F 00112        PUSHAB  FAO_RESULT_DESC
                                    0398   C2   9F 00116        PUSHAB  FAO_RESULT_DESC
                                      F8   AD   9F 0011A        PUSHAB  MEDOFL_FAO
                                    65          06   FB 0011D        CALLS   #6, SYS$FAO
                                                01   DD 00120        PUSHL   #1                                      ; 2274
                                    0398   C2   9F 00122        PUSHAB  FAO_RESULT_DESC
                    FBD4   CF               02   FB 00126        CALLS   #2, SUBMIT_REQUEST                      ;
                                                04   0012B        RET                                             ; 2277

; Routine Size:  300 bytes,    Routine Base:  $CODE$ + 0526
```

```
;  1387     2278  1 ROUTINE WRONGVOL_HNDLR : NOVALUE =
;  1388     2279  1
;  1389     2280  1 !++
;  1390     2281  1 ! Functional Description:
;  1391     2282  1 !
;  1392     2283  1 !       This routine will attempt to recover from an SS$_INCVOLLABEL
;  1393     2284  1 !       condition, which implies that the label of the volume presently
;  1394     2285  1 !       in the drive does not match the volume label specified by the user.
;  1395     2286  1 !
;  1396     2287  1 ! Input:
;  1397     2288  1 !
;  1398     2289  1 !       None.
;  1399     2290  1 !
;  1400     2291  1 ! Output:
;  1401     2292  1 !
;  1402     2293  1 !       None.
;  1403     2294  1 !
;  1404     2295  1 ! Implicit Input:
;  1405     2296  1 !
;  1406     2297  1 !       MOUNT_STATUS  = status of the current mount attempt
;  1407     2298  1 !       REPLY_PENDING = TRUE if an operator request is outstanding
;  1408     2299  1 !       The MOUNT data base.
;  1409     2300  1 !
;  1410     2301  1 ! Implict Output:
;  1411     2302  1 !
;  1412     2303  1 !       The MOUNT data base may be changed as
;  1413     2304  1 !       the result of operator intervention.
;  1414     2305  1 !--
;  1415     2306  1
;  1416     2307  2 BEGIN                                           ! Start of WRONGVOL_HNDLR
;  1417     2308  2
;  1418     2309  2 EXTERNAL
;  1419     2310  2
;  1420     2311  2         PHYS_NAME       : VECTOR VOLATILE,      ! Physical device name descriptor
;  1421     2312  2         DEVICE_INDEX    : LONG VOLATILE,        ! Index into DEVICE_STRING vector
;  1422     2313  2         LABEL_STRING    : VECTOR VOLATILE;      ! Vector of volume labels
;  1423     2314  2
;  1424     2315  2 LITERAL
;  1425     2316  2
;  1426     2317  2         FAO_CTRL_SIZ    = FAO_BUFFER_SIZE/2;    ! FAO control string size
;  1427     2318  2
;  1428     2319  2 LOCAL
;  1429     2320  2
;  1430     2321  2         WRONGVOL_FAO    : BBLOCK [DSC$K_S_BLN],
;  1431     2322  2         WRONGVOL_BUF    : BBLOCK [FAO_CTRL_SIZ],
;  1432     2323  2         STATUS          : LONG;
;  1433     2324  2
;  1434     2325  2
;  1435     2326  2 !
;  1436     2327  2 ! If this condition is different from the one signaled previously,
;  1437     2328  2 ! cancel any outstanding requests before handling this condition.
;  1438     2329  2 ! Otherwise do nothing.
;  1439     2330  2 !
;  1440     2331  3 IF (.MOUNT_STATUS NEQ .PREVIOUS_STATUS)
;  1441     2332  3 OR (.DEVICE_INDEX NEQ .PREVIOUS_DEV_IDX)
;  1442     2333  2 THEN
;  1443     2334  3     BEGIN
```

```
;  1444          2335  3      CANCEL_REQUEST (REQUEST_NOT_SATISFIED);
;  1445          2336  3      OPERATOR_PRESENT = TRUE;                        ! Assume operator present
;  1446          2337  3      !
;  1447          2338  3      ! Set up the output descriptor and get the FAO control string.
;  1448          2339  3      !
;  1449          2340  3      WRONGVOL_FAO [DSC$W_LENGTH]  = FAO_CTRL_SIZ;
;  1450          2341  3      WRONGVOL_FAO [DSC$B_DTYPE]   = DSC$K_DTYPE_T;
;  1451          2342  3      WRONGVOL_FAO [DSC$B_CLASS]   = DSC$K_CLASS_S;
;  1452          2343  3      WRONGVOL_FAO [DSC$A_POINTER] = WRONGVOL_BUF;
;  1453        P 2344  4      IF NOT (STATUS = $GETMSG    (MSGID  = MOUN$_WRONGVOL,
;  1454        P 2345  4                                   MSGLEN = WRONGVOL_FAO [DSC$W_LENGTH],
;  1455        P 2346  4                                   BUFADR = WRONGVOL_FAO,
;  1456        P 2347  4                                   FLAGS  = MSG_TEXT
;  1457          2348  4                                  ))
;  1458          2349  3      THEN
;  1459          2350  3          ABORT_MOUNT (.STATUS, 0, .MOUNT_STATUS);
;  1460          2351  3      !
;  1461          2352  3      ! Set up the output descriptor and format the operator request.
;  1462          2353  3      !
;  1463          2354  3      FAO_RESULT_DESC[DSC$A_POINTER] = FAO_BUFFER;
;  1464          2355  3      FAO_RESULT_DESC[DSC$W_LENGTH]  = FAO_BUFFER_SIZE;
;  1465        P 2356  3      $FAO (WRONGVOL_FAO,
;  1466        P 2357  3            FAO_RESULT_DESC [DSC$W_LENGTH],
;  1467        P 2358  3            FAO_RESULT_DESC,
;  1468        P 2359  3            PHYS_NAME [.DEVICE_INDEX*2]
;  1469          2360  3           );
;  1470          2361  3      !
;  1471          2362  3      ! Inform the user and all interested operators that the drive contains
;  1472          2363  3      ! the wrong volume.  Note that this is just a message, and that no
;  1473          2364  3      ! reply is expected.
;  1474          2365  3      !
;  1475          2366  3      SUBMIT_REQUEST (FAO_RESULT_DESC,NO_REPLY);
;  1476          2367  3      !
;  1477          2368  3      ! Call the medium offline handler to request that the correct volume
;  1478          2369  3      ! be mounted in the specified drive.  The previous condition context
;  1479          2370  3      ! must be reset manually, as SUBMIT_REQUEST will not do so when sending
;  1480          2371  3      ! messages (instead of requests).
;  1481          2372  3      !
;  1482          2373  3      PREVIOUS_STATUS = .MOUNT_STATUS;
;  1483          2374  3      MEDOFL_HNDLR ();
;  1484          2375  2      END;
;  1485          2376  2
;  1486          2377  1 END;                                                  ! End of WRONGVOL_HNDLR


                                        0004 00000 WRONGVOL_HNDLR:
                                                   .WORD   Save R2                                          ; 2278
                              52  0000'  CF  9E 00002      MOVAB   FAO_RESULT_DESC, R2
                              5E  FEF8   CE  9E 00007      MOVAB   -264(SP), SP
                        FC6C  C2  FC68   C2  D1 0000C      CMPL    MOUNT_STATUS, PREVIOUS_STATUS            ; 2331
                                         09  12 00013      BNEQ    1$
                        FC70  C2  0000G  CF  D1 00015      CMPL    DEVICE_INDEX, PREVIOUS_DEV_IDX           ; 2332
                                         79  13 0001C      BEQL    3$
                                         7E  D4 0001E 1$:   CLRL    -(SP)                                    ; 2335
```

```
                FD43   CF                  01   FB 00020          CALLS   #1, CANCEL_REQUEST                          ;
                FC5C   C2                  01   D0 00025          MOVL    #1, OPERATOR_PRESENT                       ; 2336
                  F8   AD 010E0100         8F   D0 0002A          MOVL    #17694976, WRONGVOL_FAO                    ; 2340
                  FC   AD                  6E   9E 00032          MOVAB   WRONGVOL_BUF, WRONGVOL_FAO+4               ; 2343
                       7E                  01   7D 00036          MOVQ    #1, -(SP)                                  ; 2348
                             F8   AD       9F   00039          PUSHAB  WRONGVOL_FAO                               ;
                             F8   AD       9F   0003C          PUSHAB  WRONGVOL_FAO                               ;
                          0072A06B   8F    DD   0003F          PUSHL   #7512171                                   ;
          00000000G       00               05   FB 00045          CALLS   #5, SYS$GETMSG                             ;
                          0F               50   E8 0004C          BLBC    STATUS, 2$                                 ;
                                FC68       C2   DD 0004F          PUSHL   MOUNT_STATUS                               ; 2350
                                           7E   D4 00053          CLRL    -(SP)                                      ;
                                           50   DD 00055          PUSHL   STATUS                                     ;
          00000000G       00               03   FB 00057          CALLS   #3, LIB$STOP                               ;
                  04      A2       FE00    C2   9E 0005E  2$:     MOVAB   FAO_BUFFER, FAO_RESULT_DESC+4              ; 2354
                          62       0200    8F   B0 00064          MOVW    #512, FAO_RESULT_DESC                      ; 2355
              50      0000G    CF          01   78 00069          ASHL    #1, DEVICE_INDEX, R0                       ; 2360
                                0000GCF40    DF   0006F          PUSHAL  PHYS_NAME[R0]                              ;
                                           52   DD 00074          PUSHL   R2                                         ;
                                           52   DD 00076          PUSHL   R2                                         ;
                                     F8    AD   9F   00078          PUSHAB  WRONGVOL_FAO                               ;
          00000000G       00               04   FB 0007B          CALLS   #4, SYS$FAO                                ;
                                           7E   D4 00082          CLRL    -(SP)                                      ; 2366
                                           52   DD 00084          PUSHL   R2                                         ;
                FB48   CF                  02   FB 00086          CALLS   #2, SUBMIT_REQUEST                         ;
                FC6C   C2        FC68      C2   D0 0008B          MOVL    MOUNT_STATUS, PREVIOUS_STATUS              ; 2373
                FE3D   CF                  00   FB 00092          CALLS   #0, MEDOFL_HNDLR                           ; 2374
                                           04   00097  3$:       RET                                                ; 2377

; Routine Size:  152 bytes,    Routine Base:  $CODE$ + 0652
```

```
;  1488          2378  1 ROUTINE PRINT_REPLY : NOVALUE =
;  1489          2379  1
;  1490          2380  1 !++
;  1491          2381  1 ! Funtional description:
;  1492          2382  1 !
;  1493          2383  1 !       This routine is a local utility routine used by PARSE_REPLY
;  1494          2384  1 !       to output the operator reply the user (SYS$OUTPUT).
;  1495          2385  1 !
;  1496          2386  1 ! Input:
;  1497          2387  1 !
;  1498          2388  1 !       None.
;  1499          2389  1 !
;  1500          2390  1 ! Output:
;  1501          2391  1 !
;  1502          2392  1 !       None.
;  1503          2393  1 !
;  1504          2394  1 ! Implicit input:
;  1505          2395  1 !
;  1506          2396  1 !       None.
;  1507          2397  1 !
;  1508          2398  1 ! Implicit output:
;  1509          2399  1 !
;  1510          2400  1 !       The operator reply, if any, is written to SYS$OUTPUT.
;  1511          2401  1 !
;  1512          2402  1 ! Side effects:
;  1513          2403  1 !
;  1514          2404  1 !       None.
;  1515          2405  1 !
;  1516          2406  1 ! Routine value:
;  1517          2407  1 !
;  1518          2408  1 !       None.
;  1519          2409  1 !
;  1520          2410  1 !--
;  1521          2411  1
;  1522          2412  2 BEGIN                                           ! Start of PRINT_REPLY
;  1523          2413  2
;  1524          2414  2 LOCAL
;  1525          2415  2         TEXT_DESC       : BBLOCK [DSC$K_S_BLN]; ! String descriptor
;  1526          2416  2
;  1527          2417  2 !
;  1528          2418  2 ! If the operator reply is greater than 8 bytes, then
;  1529          2419  2 ! it had some text to it.  If this is the case, inform
;  1530          2420  2 ! the user of the operator reply.  Note that the 8 bytes
;  1531          2421  2 ! of message overhead are not printed.  A temporary string
;  1532          2422  2 ! descriptor must be used so that $FAO will not replace
;  1533          2423  2 ! the any nonprinting ASCII characters with blanks.
;  1534          2424  2 !
;  1535          2425  2 IF .REPLY_IOSB[2,0,16,0] GTR $BYTEOFFSET (OPC$L_MS_TEXT)
;  1536          2426  2 THEN
;  1537          2427  3     BEGIN
;  1538          2428  3     TEXT_DESC [DSC$W_LENGTH]    = .REPLY_IOSB [2,0,16,0] - $BYTEOFFSET (OPC$L_MS_TEXT);
;  1539          2429  3     TEXT_DESC [DSC$B_DTYPE]     = DSC$K_DTYPE_T;
;  1540          2430  3     TEXT_DESC [DSC$B_CLASS]     = DSC$K_CLASS_S;
;  1541          2431  3     TEXT_DESC [DSC$A_POINTER]   = .REPLY_DESC [DSC$A_POINTER] + $BYTEOFFSET (OPC$L_MS_TEXT);
;  1542          2432  3     SIGNAL (MOUN$_OPREPLY, 1, TEXT_DESC);
;  1543          2433  2     END;
;  1544          2434  2
```

```
; 1545          2435  1 END;                                    ! End of PRINT_REPLY



                                                0000 00000 PRINT_REPLY:
                                                           .WORD   Save nothing                                    ; 2378
                                          5E            08  C2 00002          SUBL2   #8, SP                      ;
                                          08      0000' CF  B1 00005          CMPW    REPLY_IOSB+2, #8            ; 2425
                                                        24  1B 0000A          BLEQU   1$                          ;
                                    6E      0000' CF          08  A3 0000C    SUBW3   #8, REPLY_IOSB+2, TEXT_DESC ; 2428
                                      02    AE      010E      8F  B0 00012    MOVW    #270, TEXT_DESC+2           ; 2429
                                04    AE      0000' CF          08  C1 00018    ADDL3   #8, REPLY_DESC+4, TEXT_DESC+4 ; 2431
                                                        5E  DD 0001F          PUSHL   SP                          ; 2432
                                                        01  DD 00021          PUSHL   #1                          ;
                                              0072A02B  8F  DD 00023          PUSHL   #7512107                    ;
                                  00000000G   00            03  FB 00029    CALLS   #3, LIB$SIGNAL              ;
                                                        04  00030 1$:         RET                                 ; 2435

; Routine Size:  49 bytes,    Routine Base:  $CODE$ + 06EA
```

```
;1547     2436  1 ROUTINE PARSE_REPLY : NOVALUE =
;1548     2437  1
;1549     2438  1 !++
;1550     2439  1 ! Functional Description:
;1551     2440  1 !
;1552     2441  1 !       This routine will parse the operator reply in the context
;1553     2442  1 !       of the conditon that spawned it, and then do the appropriate
;1554     2443  1 !       thing, based on the operator's reply.
;1555     2444  1 !
;1556     2445  1 ! Input:
;1557     2446  1 !
;1558     2447  1 !       None.
;1559     2448  1 !
;1560     2449  1 ! Output:
;1561     2450  1 !
;1562     2451  1 !       None.
;1563     2452  1 !
;1564     2453  1 ! Implicit Inputs:
;1565     2454  1 !
;1566     2455  1 !       REPLY_DESC = string descriptor of the operator's reply.
;1567     2456  1 !       REPLY_BUFFER = buffer holding the operator's reply.
;1568     2457  1 !       MOUNT data base.
;1569     2458  1 !
;1570     2459  1 ! Implicit Outputs:
;1571     2460  1 !
;1572     2461  1 !       The MOUNT data base may be updated as a result of the operator's reply.
;1573     2462  1 !--
;1574     2463  1
;1575     2464  2 BEGIN                                           ! Start of PARSE_REPLY
;1576     2465  2
;1577     2466  2 EXTERNAL ROUTINE
;1578     2467  2
;1579     2468  2         LIB$TPARSE      : ADDRESSING_MODE (GENERAL);    ! Used to parse operator reply
;1580     2469  2
;1581     2470  2 PSECT   GLOBAL = $GLOBAL$;
;1582     2471  2 GLOBAL
;1583     2472  2         NEWLINE         : DESCRIP (%CHAR (13,10));      ! Descriptor for newline string
;1584     2473  2
;1585     2474  2 BIND
;1586     2475  2         !
;1587     2476  2         ! Create the character translation table that will be used by the
;1588     2477  2         ! CH$TRANSLATE function.  he table is st up so that all lower-cag
;1589     2478  2         ! alphabetic characters are translated to their upper-case equivalent.
;1590     2479  2         !
;1591     2480  2         TRANS_TABLE     = CH$TRANSTABLE
;1592     2481  2                           (0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,
;1593     2482  2                            20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,
;1594     2483  2                            37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,
;1595     2484  2                            54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,
;1596     2485  2                            71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,
;1597     2486  2                            88,89,90,91,92,93,94,95,96,65,66,67,68,69,70,71,72,
;1598     2487  2                            73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,
;1599     2488  2                            90,123,124,125,126,127
;1600     2489  2                           );
;1601     2490  2
;1602     2491  2 LOCAL
;1603     2492  2
```

```
;  1604     2493  2          PTR             : LONG,                         ! Character pointer
;  1605     2494  2          STATUS          : LONG;
;  1606     2495  2
;  1607     2496  2  !
;  1608     2497  2  ! Check the status of the mailbox read.  If
;  1609     2498  2  ! not successful, then abort the mount.
;  1610     2499  2  !
;  1611     2500  2  IF NOT .REPLY_IOSB[0,0,16,0]
;  1612     2501  2  THEN
;  1613     2502  3      BEGIN
;  1614     2503  3      REPLY_PENDING = FALSE;
;  1615     2504  3      ABORT_MOUNT (MOUN$_MBXRDERR, 0, .REPLY_IOSB[0,0,16,0]);
;  1616     2505  2      END;
;  1617     2506  2
;  1618     2507  2  !
;  1619     2508  2  ! Decide what to do based on the type of operator reply.
;  1620     2509  2  ! The OPC$_xxxxx status codes are longer than a word, so
;  1621     2510  2  ! they are masked off to word size befor comparing them
;  1622     2511  2  ! to the reply status.
;  1623     2512  2  !
;  1624     2513  2  SELECTONEU .REPLY_BUFFER[OPC$W_MS_STATUS] OF
;  1625     2514  2      SET
;  1626     2515  3      [(OPC$_NOPERATOR AND %X'0FFFF')]    : BEGIN
;  1627     2516  3                                            !
;  1628     2517  3                                            ! No operator was enabled to receive the request.
;  1629     2518  3                                            !
;  1630     2519  3                                            REPLY_PENDING = FALSE;
;  1631     2520  3                                            IF NOT INTERACTIVE_JOB ()
;  1632     2521  3                                            THEN
;  1633     2522  3                                                !
;  1634     2523  3                                                ! Abort the mount, as no one is can service the request.
;  1635     2524  3                                                !
;  1636     2525  3                                                ABORT_MOUNT (MOUN$_BATCHNOOPR)
;  1637     2526  3                                            ELSE
;  1638     2527  4                                                BEGIN
;  1639     2528  4                                                !
;  1640     2529  4                                                ! If this is the first time through this code for this conditi
;  1641     2530  4                                                ! for this device, then inform the user that no operator is en
;  1642     2531  4                                                ! to receive the request.
;  1643     2532  4                                                !
;  1644     2533  4                                                IF .OPERATOR_PRESENT
;  1645     2534  4                                                THEN
;  1646     2535  4                                                    SIGNAL (MOUN$_NOOPR);
;  1647     2536  4                                                OPERATOR_PRESENT = FALSE;
;  1648     2537  4                                                !
;  1649     2538  4                                                ! Re-issue the request, in the hope that an operator will even
;  1650     2539  4                                                ! be enabled to receive and service the request.
;  1651     2540  4                                                !
;  1652     2541  5                                                IF NOT (STATUS = $SNDOPR (MSGBUF=OP_MSG_DESC, CHAN=.REPLY_CHAN
;  1653     2542  4                                                THEN
;  1654     2543  4                                                    ABORT_MOUNT (MOUN$_OPRSNDERR, 0, .STATUS);
;  1655     2544  4                                                !
;  1656     2545  4                                                ! If the request was sent, re-issue a read to the reply mailbo
;  1657     2546  4                                                !
;  1658     2547  4                                                IF .STATUS NEQ OPC$_NOPERATOR
;  1659     2548  4                                                THEN
;  1660     2549  5                                                    BEGIN
```

```
;  1661     2550  5                                                    POST_READ_TO_MBX ();
;  1662     2551  5                                                    REPLY_PENDING = TRUE;
;  1663     2552  4                                                    END;
;  1664     2553  3                                                END;
;  1665     2554  2                                            END;
;  1666     2555  2
;  1667     2556  3        [(OPC$_RQSTCMPLTE AND %X'0FFFF')]   : BEGIN
;  1668     2557  3                                            !
;  1669     2558  3                                            ! The operator replied to our request.
;  1670     2559  3                                            !
;  1671     2560  3                                            PRINT_REPLY ();
;  1672     2561  3                                            PREVIOUS_STATUS = -1;
;  1673     2562  3                                            REPLY_PENDING = FALSE;
;  1674     2563  3                                            OPERATOR_PRESENT = TRUE;
;  1675     2564  3                                            !
;  1676     2565  3                                            ! If there is no operator reply text, then return.
;  1677     2566  3                                            !
;  1678     2567  4                                            IF (.REPLY_IOSB [2,0,16,0] EQL $BYTEOFFSET (OPC$L_MS_TEXT))
;  1679     2568  3                                            THEN
;  1680     2569  3                                                RETURN;
;  1681     2570  3                                            !
;  1682     2571  3                                            ! Create a string descriptor for the operator reply text.
;  1683     2572  3                                            !
;  1684     2573  3                                            TPARSE_BLOCK [TPA$L_STRINGCNT] = .REPLY_IOSB [2,0,16,0] - $BYTEOFF
;  1685     2574  3                                            TPARSE_BLOCK [TPA$L_STRINGPTR] = .REPLY_DESC [DSC$A_POINTER]+$BYTE
;  1686     2575  3                                            !
;  1687     2576  3                                            ! The reply text may contain a NEWLINE character.  If so, the inte
;  1688     2577  3                                            ! is BEFORE  the NEWLINE character.  Note that the NEWLINE charact
;  1689     2578  3                                            ! two characters, a carriage-return followed by a line-feed (<cr><
;  1690     2579  3                                            !
;  1691     2580  3                                            PTR   = CH$FIND_SUB   (.TPARSE_BLOCK [TPA$L_STRINGCNT],
;  1692     2581  3                                                                   .TPARSE_BLOCK [TPA$L_STRINGPTR],
;  1693     2582  3                                                                   .NEWLINE [DSC$W_LENGTH],
;  1694     2583  3                                                                   .NEWLINE [DSC$A_POINTER]
;  1695     2584  3                                                                  );
;  1696     2585  3                                            !
;  1697     2586  3                                            ! If a NEWLINE was found, set the string descriptor
;  1698     2587  3                                            ! so that the text BEFORE the NEWLINE is parsed.
;  1699     2588  3                                            !
;  1700     2589  3                                            IF NOT CH$FAIL (.PTR)
;  1701     2590  3                                            THEN
;  1702     2591  3                                                TPARSE_BLOCK [TPA$L_STRINGCNT] = .PTR - .TPARSE_BLOCK [TPA$L_S
;  1703     2592  3                                            !
;  1704     2593  3                                            ! If there is no text before the NEWLINE, then there is no operato
;  1705     2594  3                                            !
;  1706     2595  3                                            IF .TPARSE_BLOCK [TPA$L_STRINGCNT] EQL 0
;  1707     2596  3                                            THEN
;  1708     2597  3                                                RETURN;
;  1709     2598  3                                            !
;  1710     2599  3                                            ! Convert the reply to upper case, so TPARSE will work correctly.
;  1711     2600  3                                            !
;  1712     2601  3                                            CH$TRANSLATE  (TRANS_TABLE,
;  1713     2602  3                                                           .TPARSE_BLOCK [TPA$L_STRINGCNT],
;  1714     2603  3                                                           .TPARSE_BLOCK [TPA$L_STRINGPTR],
;  1715     2604  3                                                           0,
;  1716     2605  3                                                           .TPARSE_BLOCK [TPA$L_STRINGCNT],
;  1717     2606  3                                                           .TPARSE_BLOCK [TPA$L_STRINGPTR]
```

```
;  1718     2607  3                                                    );
;  1719     2608  3                                                !
;  1720     2609  3                                                ! Parse the operator response and perform whatever action is neces
;  1721     2610  3                                                !
;  1722     2611  4                                                IF NOT (STATUS = LIB$TPARSE (TPARSE_BLOCK, STATE_TABLE, KEY_TABLE)
;  1723     2612  3                                                THEN
;  1724     2613  3                                                    ABORT_MOUNT (.STATUS, 0, .MOUNT_STATUS);
;  1725     2614  2                                                END;
;  1726     2615  2
;  1727     2616  3      [(OPC$_RQSTPEND AND %X'0FFFF')]           : BEGIN
;  1728     2617  3                                                !
;  1729     2618  3                                                ! The operator did a REPLY/PENDING.  The orginal
;  1730     2619  3                                                ! request is still active, so issue another read
;  1731     2620  3                                                ! to the reply mailbox.
;  1732     2621  3                                                !
;  1733     2622  3                                                PRINT_REPLY ();
;  1734     2623  3                                                OPERATOR_PRESENT = TRUE;
;  1735     2624  3                                                POST_READ_TO_MBX ();
;  1736     2625  2                                                END;
;  1737     2626  2
;  1738     2627  3      [(OPC$_RQSTABORT AND %X'0FFFF')]          : BEGIN
;  1739     2628  3                                                !
;  1740     2629  3                                                ! The operator has aborted the mount request.
;  1741     2630  3                                                !
;  1742     2631  3                                                PRINT_REPLY ();
;  1743     2632  3                                                REPLY_PENDING = FALSE;
;  1744     2633  3                                                OPERATOR_PRESENT = TRUE;
;  1745     2634  3                                                ABORT_MOUNT (MOUN$_OPRABORT);
;  1746     2635  2                                                END;
;  1747     2636  2
;  1748     2637  2      [(OPC$_RQSTCAN  AND %X'0FFFF'),
;  1749     2638  3       (OPC$_RQSTDONE AND %X'0FFFF')]           : BEGIN
;  1750     2639  3                                                !
;  1751     2640  3                                                ! The user has canceled the requst, and
;  1752     2641  3                                                ! the operator is acknowledging it.
;  1753     2642  3                                                !
;  1754     2643  3                                                PREVIOUS_STATUS = -1;
;  1755     2644  3                                                REPLY_PENDING = FALSE;
;  1756     2645  3                                                OPERATOR_PRESENT = TRUE;
;  1757     2646  2                                                END;
;  1758     2647  2
;  1759     2648  2      [(OPC$_BLANKTAPE AND %X'0FFFF'),
;  1760     2649  3       (OPC$_INITAPE   AND %X'0FFFF')]          : BEGIN
;  1761     2650  3                                                !
;  1762     2651  3                                                ! These messages may be sent by mistake.  Notify
;  1763     2652  3                                                ! the interested parties, and let MOUNT try again.
;  1764     2653  3                                                !
;  1765     2654  3                                                PREVIOUS_STATUS = -1;
;  1766     2655  3                                                REPLY_PENDING = FALSE;
;  1767     2656  3                                                OPERATOR_PRESENT = TRUE;
;  1768     2657  3                                                INVALID_COMMAND ();
;  1769     2658  2                                                END;
;  1770     2659  2
;  1771     2660  3      [OTHERWISE]                               : BEGIN
;  1772     2661  3                                                !
;  1773     2662  3                                                ! This is an unknown response type.
;  1774     2663  3                                                ! Abort the mount and print the bad message.
```

```
;  1775          2664  3
;  1776          2665  3
;  1777          2666  3
;  1778        P 2667  3
;  1779        P 2668  3
;  1780        P 2669  3
;  1781        P 2670  3
;  1782        P 2671  3
;  1783        P 2672  3
;  1784        P 2673  3
;  1785          2674  3
;  1786          2675  2
;  1787          2676  2     TES;
;  1788          2677  2
;  1789          2678  1 END;
```

```
                !
                REPLY_PENDING = FALSE;
                OPERATOR_PRESENT = TRUE;
                ABORT_MOUNT   (MOUN$_BADREPLY,                  ! Error code
                               5,                               ! FAO count
                               .REPLY_BUFFER[OPC$B_MS_TYPE],    ! Message type
                               .REPLY_BUFFER[OPC$W_MS_STATUS],  ! Message status
                               .REPLY_BUFFER[OPC$L_MS_RPLYID],  ! Message Ident
                               .REPLY_DESC[DSC$W_LENGTH]  - $BYTEOFFSET (OPC$L_MS_
                               .REPLY_DESC[DSC$A_POINTER] + $BYTEOFFSET (OPC$L_MS_
                              );
                END;

                     ! End of PARSE_REPLY


                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                                                    0A  0D  00008 P.AAB:  .ASCII  <13><10>                                ;
                                                                0000A         .BLKB   2
0E  0D  0C  0B  0A  09  08  07  06  05  04  03  02  01  00  0000C P.AAC:  .BYTE   0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, - ;
1D  1C  1B  1A  19  18  17  16  15  14  13  12  11  10  0F  0001B                13, 14, 15, 16, 17, 18, 19, 20, 21, 22, -   ;
2C  2B  2A  29  28  27  26  25  24  23  22  21  20  1F  1E  0002A                23, 24, 25, 26, 27, 28, 29, 30, 31, 32, -   ;
3B  3A  39  38  37  36  35  34  33  32  31  30  2F  2E  2D  00039                33, 34, 35, 36, 37, 38, 39, 40, 41, 42, -   ;
4A  49  48  47  46  45  44  43  42  41  40  3F  3E  3D  3C  00048                43, 44, 45, 46, 47, 48, 49, 50, 51, 52, -   ;
59  58  57  56  55  54  53  52  51  50  4F  4E  4D  4C  4B  00057                53, 54, 55, 56, 57, 58, 59, 60, 61, 62, -   ;
48  47  46  45  44  43  42  41  60  5F  5E  5D  5C  5B  5A  00066                63, 64, 65, 66, 67, 68, 69, 70, 71, 72, -   ;
57  56  55  54  53  52  51  50  4F  4E  4D  4C  4B  4A  49  00075                73, 74, 75, 76, 77, 78, 79, 80, 81, 82, -   ;
                        7F  7E  7D  7C  7B  5A  59  58  00084                83, 84, 85, 86, 87, 88, 89, 90, 91, 92, -   ;
                                                                                    93, 94, 95, 96, 65, 66, 67, 68, 69, 70, -   ;
                                                                                    71, 72, 73, 74, 75, 76, 77, 78, 79, 80, -   ;
                                                                                    81, 82, 83, 84, 85, 86, 87, 88, 89, 90, -   ;
                                                                                    123, 124, 125, 126, 127                     ;

                                                .PSECT  $GLOBAL$,NOEXE,2

                                            0002  00000 NEWLINE::
                                                        .WORD   2                                                       ;
                                              0E  00002         .BYTE   14                                             ;
                                              01  00003         .BYTE   1                                              ;
                                      00000000'   00004         .ADDRESS P.AAB                                        ;

                                                    TRANS_TABLE=        P.AAC
                                                        .EXTRN  LIB$TPARSE

                                                        .PSECT  $CODE$,NOWRT,2

                                            03FC  00000 PARSE_REPLY:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9                            ; 2436
                                    59        CA    AF  9E  00002         MOVAB   PRINT_REPLY, R9                       ;
                                    58 00000000G    00  9E  00006         MOVAB   LIB$STOP, R8                          ;
                                    57      0000'   CF  9E  0000D         MOVAB   REPLY_PENDING, R7                     ;
                                    11        3C    A7  E8  00012         BLBS    REPLY_IOSB, 1$                        ; 2500
                                                    67  D4  00016         CLRL    REPLY_PENDING                         ; 2503
                                    7E        3C    A7  3C  00018         MOVZWL  REPLY_IOSB, -(SP)                     ; 2504
```

```
                                7E  D4 0001C          CLRL    -(SP)                                                                   ;
                     007281DC   8F  DD 0001E          PUSHL   #7504348                                                                ;
                             68  03  FB 00024          CALLS   #3, LIB$STOP                                                            ;
                             52      46  A7  3C 00027 1$:      MOVZWL  REPLY_BUFFER+2, R2                                              ; 2513
                           8061  8F      52  B1 0002B          CMPW    R2, #32865                                                      ; 2515
                                         5A  12 00030          BNEQ    5$                                                              ;
                                         67  D4 00032          CLRL    REPLY_PENDING                                                   ; 2519
                           FB03  C9      00  FB 00034          CALLS   #0, INTERACTIVE_JOB                                             ; 2520
                                 09      50  E8 00039          BLBS    R0, 2$                                                          ;
                                    007281FC 8F  DD 0003C      PUSHL   #7504380                                                        ; 2525
                                       00FD  31  00042         BRW     14$                                                             ;
                                 0D      08  A7  E9 00045 2$:  BLBC    OPERATOR_PRESENT, 3$                                            ; 2533
                                    0072A03B 8F  DD 00049      PUSHL   #7512123                                                        ; 2535
                        00000000G    00      01  FB 0004F     CALLS   #1, LIB$SIGNAL                                                  ;
                                         08  A7  D4 00056 3$:  CLRL    OPERATOR_PRESENT                                                ; 2536
                                         38  A7  DD 00059      PUSHL   REPLY_CHANNEL                                                   ; 2541
                                       0180  C7  9F 0005C      PUSHAB  OP_MSG_DESC                                                     ;
                        00000000G    00      02  FB 00060     CALLS   #2, SYS$SNDOPR                                                  ;
                                 56      50  D0 00067          MOVL    R0, STATUS                                                      ;
                                 0D      56  E8 0006A          BLBS    STATUS, 4$                                                      ;
                                         56  DD 0006D          PUSHL   STATUS                                                          ; 2543
                                         7E  D4 0006F          CLRL    -(SP)                                                           ;
                                    007281EC 8F  DD 00071      PUSHL   #7504364                                                        ;
                                 68      03  FB 00077          CALLS   #3, LIB$STOP                                                    ;
                        00058061 8F      56  D1 0007A 4$:      CMPL    STATUS, #360545                                                 ; 2547
                                         60  13 00081          BEQL    9$                                                              ;
                           FACB  C9      00  FB 00083          CALLS   #0, POST_READ_TO_MBX                                            ; 2550
                                 67      01  D0 00088          MOVL    #1, REPLY_PENDING                                               ; 2551
                                         04  00088B           RET                                                                     ; 2513
                           8029  8F          52  B1 0008C 5$:  CMPW    R2, #32809                                                      ; 2556
                                             03  13 00091      BEQL    6$                                                              ;
                                           0082  31 00093      BRW     12$                                                             ;
                                 69          00  FB 00096 6$:  CALLS   #0, PRINT_REPLY                                                ; 2560
                                     18      A7  01  CE 00099  MNEGL   #1, PREVIOUS_STATUS                                             ; 2561
                                             67  D4 0009D      CLRL    REPLY_PENDING                                                   ; 2562
                                     08      A7  01  D0 0009F  MOVL    #1, OPERATOR_PRESENT                                            ; 2563
                                         08      3E  A7  B1 000A3  CMPW    REPLY_IOSB+2, #8                                            ; 2567
                                                 3A  13 000A7      BEQL    9$                                                          ;
                                   00DC  C7      3E  A7  3C 000A9  MOVZWL  REPLY_IOSB+2, TPARSE_BLOCK+8                                ; 2573
                                   00DC  C7          08  C2 000AF  SUBL2   #8, TPARSE_BLOCK+8                                          ;
                             00E0  C7    00D0  C7      08  C1 000B4  ADDL3   #8, REPLY_DESC+4, TPARSE_BLOCK+12                        ; 2574
                                         54  0000'     CF  3C 000BC  MOVZWL  NEWLINE, R4                                             ; 2582
                        00E0  D7   00DC  C7  0000'     DF      54  39 000C1  MATCHC  R4, @NEWLINE+4, TPARSE_BLOCK+8, -                 ; 2583
                                                                                 @TPARSE_BLOCK+12                                       ;
                                             03  13 000CC      BEQL    7$                                                          ;
                                         53      54  D0 000CE  MOVL    R4, R3                                                      ;
                                         53      54  C2 000D1  7$.     SUBL2   R4, R3                                              ;
                                             08  13 000D4      BEQL    8$                                                          ; 2589
                                   00DC  C7      53  C3 000D6  SUBL3   TPARSE_BLOCK+12, PTR, TPARSE_BLOCK+8                        ; 2591
                                         50  00DC  C7  D0 000DE  8$:     MOVL    TPARSE_BLOCK+8, R0                                  ; 2595
                                             01  12 000E3  9$:     BNEQ    10$                                                         ;
                                                 04 000E5       RET                                                                     ;
                        0000'  CF        00  00E0  D7      50  2E 000E6  10$:    MOVTC   R0, @TPARSE_BLOCK+12, #0, TRANS_TABLE, R0, - ; 2606
                                                 00E0  D7      50      000EF                  @TPARSE_BLOCK+12                        ;
                                                         0000V CF  9F 000F3  PUSHAB  KEY_TABLE                                       ; 2611
                                                         0000V CF  9F 000F7  PUSHAB  STATE_TABLE                                     ;
                                                         00D4  C7  9F 000FB  PUSHAB  TPARSE_BLOCK                                    ;
```

```
                00000000G  00              03  FB 000FF          CALLS   #3, LIB$TPARSE                          ;
                           56              50  D0 00106          MOVL    R0, STATUS                             ;
                           01              56  E9 00109          BLBC    STATUS, 11$                            ;
                                           04  0010C           RET                                            ;
                                    14     A7  DD 0010D 11$:     PUSHL   MOUNT_STATUS                           ; 2613
                                           7E  D4 00110          CLRL    -(SP)                                  ;
                                           56  DD 00112          PUSHL   STATUS                                 ;
                           68              03  FB 00114          CALLS   #3, LIB$STOP                           ;
                                           04  00117           RET                                            ; 2513
                      8021 8F              52  B1 00118 12$:     CMPW    R2, #32801                             ; 2616
                                           0D  12 0011D          BNEQ    13$                                    ;
                           69              00  FB 0011F          CALLS   #0, PRINT_REPLY                        ; 2622
                        08 A7              01  D0 00122          MOVL    #1, OPERATOR_PRESENT                   ; 2623
                      FACB C9              00  FB 00126          CALLS   #0, POST_READ_TO_MBX                   ; 2624
                                           04  0012B           RET                                            ; 2513
                      801C 8F              52  B1 0012C 13$:     CMPW    R2, #32796                             ; 2627
                                           13  12 00131          BNEQ    15$                                    ;
                           69              00  FB 00133          CALLS   #0, PRINT_REPLY                        ; 2631
                                           67  D4 00136          CLRL    REPLY_PENDING                          ; 2632
                        08 A7              01  D0 00138          MOVL    #1, OPERATOR_PRESENT                   ; 2633
                                 007281F4  8F  DD 0013C          PUSHL   #7504372                               ; 2634
                           68              01  FB 00142 14$:     CALLS   #1, LIB$STOP                           ;
                                           04  00145           RET                                            ; 2513
                      8084 8F              52  B1 00146 15$:     CMPW    R2, #32900                             ; 2637
                                           07  13 0014B          BEQL    16$                                    ;
                      81DB 8F              52  B1 0014D          CMPW    R2, #33243                             ; 2638
                                           0B  12 00152          BNEQ    17$                                    ;
                        18 A7              01  CE 00154 16$:     MNEGL   #1, PREVIOUS_STATUS                    ; 2643
                                           67  D4 00158          CLRL    REPLY_PENDING                          ; 2644
                        08 A7              01  D0 0015A          MOVL    #1, OPERATOR_PRESENT                   ; 2645
                                           04  0015E           RET                                            ; 2513
                      81D3 8F              52  B1 0015F 17$:     CMPW    R2, #33235                             ; 2649
                                           07  13 00164          BEQL    18$                                    ;
                      81E3 8F              52  B1 00166          CMPW    R2, #33251                             ; 2648
                                           10  12 0016B          BNEQ    19$                                    ;
                        18 A7              01  CE 0016D 18$:     MNEGL   #1, PREVIOUS_STATUS                    ; 2654
                                           67  D4 00171          CLRL    REPLY_PENDING                          ; 2655
                        08 A7              01  D0 00173          MOVL    #1, OPERATOR_PRESENT                   ; 2656
                      0000V CF             00  FB 00177          CALLS   #0, INVALID_COMMAND                    ; 2657
                                           04  0017C           RET                                            ; 2513
                                           67  D4 0017D 19$:     CLRL    REPLY_PENDING                          ; 2665
                        08 A7              01  D0 0017F          MOVL    #1, OPERATOR_PRESENT                   ; 2666
                   7E   00D0 C7            1A  C1 00183          ADDL3   #26, REPLY_DESC+4, -(SP)               ; 2674
                             7E      00CC  C7  3C 00189          MOVZWL  REPLY_DESC, -(SP)                      ;
                             6E            1A  C2 0018E          SUBL2   #26, (SP)                              ;
                                     48    A7  DD 00191          PUSHL   REPLY_BUFFER+4                         ;
                             7E      46    A7  3C 00194          MOVZWL  REPLY_BUFFER+2, -(SP)                  ;
                             7E      44    A7  9A 00198          MOVZBL  REPLY_BUFFER, -(SP)                    ;
                                           05  DD 0019C          PUSHL   #5                                     ;
                                 007281E4  8F  DD 0019E          PUSHL   #7504356                               ;
                           68              07  FB 001A4          CALLS   #7, LIB$STOP                           ;
                                           04  001A7           RET                                            ; 2678

; Routine Size:  424 bytes,    Routine Base:  $CODE$ + 071B
```

```
;  1791          2679  1 ROUTINE SAVE_DEVICE =
;  1792          2680  1
;  1793          2681  1 !++
;  1794          2682  1 ! Functional description:
;  1795          2683  1 !
;  1796          2684  1 !       This is a TPARSE action routine that is called
;  1797          2685  1 !       to create a string descriptor for the token
;  1798          2686  1 !       just parsed.  The token is a device name.
;  1799          2687  1 !
;  1800          2688  1 ! Input:
;  1801          2689  1 !
;  1802          2690  1 !       None.
;  1803          2691  1 !
;  1804          2692  1 ! Output:
;  1805          2693  1 !
;  1806          2694  1 !       None.
;  1807          2695  1 !
;  1808          2696  1 ! Implicit Inputs:
;  1809          2697  1 !
;  1810          2698  1 !       TPARSE_BLOCK = data structure defining TPARSE context.
;  1811          2699  1 !
;  1812          2700  1 ! Implicit outputs:
;  1813          2701  1 !
;  1814          2702  1 !       DEVICE_DESC = string descriptor of device name.
;  1815          2703  1 !
;  1816          2704  1 ! Routine Value:
;  1817          2705  1 !
;  1818          2706  1 !       1 If the device name length is within tolerance,
;  1819          2707  1 !       0 if not.
;  1820          2708  1 !
;  1821          2709  1 !--
;  1822          2710  1
;  1823          2711  2 BEGIN                                           ! Start of SAVE_DEVICE
;  1824          2712  2
;  1825          2713  2
;  1826          2714  2 EXTERNAL
;  1827          2715  2
;  1828          2716  2         DEVICE_DESC     : BBLOCK,               ! Device string descriptor
;  1829          2717  2         TPARSE_BLOCK    : BBLOCK;               ! TPARSE context data structure
;  1830          2718  2
;  1831          2719  2 IF .TPARSE_BLOCK[TPA$L_TOKENCNT] GTR MAX_DEV_LENGTH          ! Check for device name too long
;  1832          2720  2 THEN
;  1833          2721  2     0                                                       ! Return failure
;  1834          2722  2 ELSE
;  1835          2723  3     BEGIN
;  1836          2724  3     DEVICE_DESC[DSC$W_LENGTH]  = .TPARSE_BLOCK[TPA$L_TOKENCNT];
;  1837          2725  3     DEVICE_DESC[DSC$A_POINTER] = .TPARSE_BLOCK[TPA$L_TOKENPTR];
;  1838          2726  3     1                                                       ! Return success
;  1839          2727  3     END
;  1840          2728  3
;  1841          2729  1 END;                                           ! End of SAVE_DEVICE


                                0000 00000 SAVE_DEVICE:
```

```
                                                        .WORD   Save nothing                                    ; 2679
                        3F      0000G   CF      D1 00002        CMPL    TPARSE_BLOCK+16, #63                    ; 2719
                                        03      15 00007        BLEQ    1$                                      ;
                                        50      D4 00009        CLRL    R0                                      ;
                                                04 0000B        RET                                             ;
                0000G   CF      0000G   CF      B0 0000C 1$:    MOVW    TPARSE_BLOCK+16, DEVICE_DESC            ; 2724
                0000G   CF      0000G   CF      D0 00013        MOVL    TPARSE_BLOCK+20, DEVICE_DESC+4          ; 2725
                        50              01      D0 0001A        MOVL    #1, R0                                  ; 2723
                                                04 0001D        RET                                             ; 2729

; Routine Size:  30 bytes,    Routine Base:  $CODE$ + 08C3
```

```
;  1843          2730  1 ROUTINE DO_SUBSTITUTE =
;  1844          2731  1
;  1845          2732  1 !++
;  1846          2733  1 ! Funtional desctiption:
;  1847          2734  1 !
;  1848          2735  1 !       This routine is merely a shell so that $COPY_INFO may be
;  1849          2736  1 !       called during the TPARSE operation to copy the new device
;  1850          2737  1 !       name to the mount data base.
;  1851          2738  1 !
;  1852          2739  1 !       Note that the previous device must be deallocated before
;  1853          2740  1 !       we copy the new device name into the data base.
;  1854          2741  1 !
;  1855          2742  1 ! Input:
;  1856          2743  1 !
;  1857          2744  1 !       None.
;  1858          2745  1 !
;  1859          2746  1 ! Output:
;  1860          2747  1 !
;  1861          2748  1 !       None.
;  1862          2749  1 !
;  1863          2750  1 ! Implict input:
;  1864          2751  1 !
;  1865          2752  1 !       DEVICE_DESC     : a device name descriptor
;  1866          2753  1 !       DEVICE_INDEX    : the current device index into the DEVICE_STRING vector
;  1867          2754  1 !
;  1868          2755  1 ! Implict output:
;  1869          2756  1 !
;  1870          2757  1 !       The mount data base may be modified.
;  1871          2758  1 !
;  1872          2759  1 ! Routine value:
;  1873          2760  1 !
;  1874          2761  1 !       See the description of $COPY_INFO.
;  1875          2762  1 !--
;  1876          2763  1
;  1877          2764  2 BEGIN                                             ! Start of DO_SUBSTITUE
;  1878          2765  2
;  1879          2766  2 EXTERNAL
;  1880          2767  2         DEVICE_INDEX    : LONG,
;  1881          2768  2         DEVICE_DESC     : BBLOCK;
;  1882          2769  2
;  1883          2770  2 EXTERNAL ROUTINE
;  1884          2771  2         $DALLOC_DEVS$U  : ADDRESSING_MODE (GENERAL),    ! Address of the transfer vector
;  1885          2772  2         $COPY_INFO$U    : ADDRESSING_MODE (GENERAL);    ! Address of the transfer vector
;  1886          2773  2
;  1887          2774  2 $DALLOC_DEVS$U (1);                                 ! Deallocate old device
;  1888          2775  2 $COPY_INFO$U (.DEVICE_INDEX, DEVICE_DESC)          ! Copy string and return status
;  1889          2776  2
;  1890          2777  1 END;                                               ! End of DO_SUBSTITUTE


                                                        .EXTRN  $COPY_INFO$U

                                        0000 00000 DO_SUBSTITUTE:
                                                        .WORD   Save nothing                    ; 2730
                                        01  DD 00002         PUSHL   #1                      ; 2774
                    00000000G   00      01  FB 00004         CALLS   #1, $DALLOC_DEVS$U      ;
```

```
                          0000G  CF  9F 0000B         PUSHAB  DEVICE_DESC                              ; 2775
                          0000G  CF  DD 0000F         PUSHL   DEVICE_INDEX                             ;
                00000000G  00        02  FB 00013    CALLS   #2, $COPY_INFO$U                         ;
                                     04  0001A       RET                                              ; 2777

; Routine Size:  27 bytes,    Routine Base:  $CODE$ + 08E1
```

```
;  1892          2778  1 ROUTINE INVALID_COMMAND =
;  1893          2779  1
;  1894          2780  1 !++
;  1895          2781  1 ! Functional Description:
;  1896          2782  1 !
;  1897          2783  1 !       This routine is the TPARSE action routine that implements
;  1898          2784  1 !       invalid command handling and reporting.  If we get here,
;  1899          2785  1 !       it means that TPARSE has detected a bogus operator reply.
;  1900          2786  1 !       The user is notified that the operator response was invalid,
;  1901          2787  1 !       and the mount operation continues.  If the condition that
;  1902          2788  1 !       caused the initial error still exists, then MOUNT will issue
;  1903          2789  1 !       another request to the operator.  The reason the operator is
;  1904          2790  1 !       not notified of his mistake is that there is no way to target
;  1905          2791  1 !       a message to specific operator.
;  1906          2792  1 !
;  1907          2793  1 ! Input:
;  1908          2794  1 !
;  1909          2795  1 !       None.
;  1910          2796  1 !
;  1911          2797  1 ! Output:
;  1912          2798  1 !
;  1913          2799  1 !       None.
;  1914          2800  1 !
;  1915          2801  1 ! Implicit Inputs:
;  1916          2802  1 !
;  1917          2803  1 !       None.
;  1918          2804  1 !
;  1919          2805  1 ! Implicit Outputs:
;  1920          2806  1 !
;  1921          2807  1 !       The user is informed of the operator's mistake.
;  1922          2808  1 !
;  1923          2809  1 ! Routine value:
;  1924          2810  1 !
;  1925          2811  1 !       Always 1.
;  1926          2812  1 !--
;  1927          2813  1
;  1928          2814  2 BEGIN                                     ! Start of INVALID_COMMAND
;  1929          2815  2
;  1930          2816  2 SIGNAL (MOUN$_INVLDRESP);
;  1931          2817  2
;  1932          2818  2 1
;  1933          2819  1 END;                                     ! End of INVALID_COMMAND



                                        0000 00000 INVALID_COMMAND:
                                                   .WORD   Save nothing                 ; 2778
                              0072A043   8F  DD 00002          PUSHL   #7512131          ; 2816
                   00000000G  00         01  FB 00008          CALLS   #1, LIB$SIGNAL    ;
                              50         01  D0 0000F          MOVL    #1, R0            ; 2819
                                         04 00012            RET                       ;

; Routine Size:  19 bytes,    Routine Base:  $CODE$ + 08FC
```

```
;  1935     2820  1 GLOBAL ROUTINE $COPY_INFO (DEV_INDEX, DEV_DESC) =
;  1936     2821  1
;  1937     2822  1 !++
;  1938     2823  1 ! Functional description:
;  1939     2824  1 !
;  1940     2825  1 !       This routine provides a secure way of copying a device name
;  1941     2826  1 !       string from the caller (in user mode) to MOUNT's protected
;  1942     2827  1 !       data base (in EXEC mode).
;  1943     2828  1 !
;  1944     2829  1 ! Input:
;  1945     2830  1 !
;  1946     2831  1 !       DEV_INDEX       : A number from 0 to .DEVICE_COUNT
;  1947     2832  1 !       DEV_DESC        : Address of a device name descriptor
;  1948     2833  1 !
;  1949     2834  1 ! Output:
;  1950     2835  1 !
;  1951     2836  1 !       None.
;  1952     2837  1 !
;  1953     2838  1 ! Implicit input:
;  1954     2839  1 !
;  1955     2840  1 !       DEVICE_STRING   : A vector of device name descriptors
;  1956     2841  1 !       DEVICE_COUNT    : The number of devices specified by the user.
;  1957     2842  1 !
;  1958     2843  1 ! Implicit output:
;  1959     2844  1 !
;  1960     2845  1 !       The DEVICE_STRING vector may be modified.
;  1961     2846  1 !
;  1962     2847  1 ! Routine value:
;  1963     2848  1 !
;  1964     2849  1 !       SS$_NORMAL      : Normal successful completion
;  1965     2850  1 !       SS$_ACCVIO      : The specified device name cannot be read.
;  1966     2851  1 !       SS$_BADPARAM    : The specified device name has a zero length,
;  1967     2852  1 !                         or is longer than LOG$C_NAMLENGTH bytes, or
;  1968     2853  1 !                         DEV_INDEX is not a reasonable value.
;  1969     2854  1 !--
;  1970     2855  1
;  1971     2856  2 BEGIN                                           ! Start of $COPY_INFO
;  1972     2857  2
;  1973     2858  2 EXTERNAL
;  1974     2859  2         DEVICE_COUNT    : LONG,                 ! # of drives
;  1975     2860  2         DEVICE_STRING   : VECTOR VOLATILE;      ! Descriptor list
;  1976     2861  2
;  1977     2862  2 BUILTIN
;  1978     2863  2         PROBER;                                 ! Probe for read access
;  1979     2864  2
;  1980     2865  2 LOCAL
;  1981     2866  2         DEV_NAME        : BBLOCK [DSC$K_S_BLN]; ! Local descriptor
;  1982     2867  2
;  1983     2868  2 !
;  1984     2869  2 ! Make sure DEV_INDEX is within a reasonable range.
;  1985     2870  2 !
;  1986     2871  3 IF (.DEV_INDEX LSS 0) OR (.DEV_INDEX GTR (.DEVICE_COUNT - 1))
;  1987     2872  2 THEN
;  1988     2873  2     RETURN (SS$_BADPARAM);
;  1989     2874  2
;  1990     2875  2 !
;  1991     2876  2 ! Probe the actual descriptor for read access.
```

```
;  1992          2877  2 !
;  1993          2878  2 IF NOT PROBER (%REF (0), %REF (DSC$K_S_BLN), .DEV_DESC)
;  1994          2879  2 THEN
;  1995          2880  2     RETURN (SS$_ACCVIO);
;  1996          2881  2
;  1997          2882  2 !
;  1998          2883  2 ! Copy the descriptor to internal storage and then probe the
;  1999          2884  2 ! device name for read access, and make sure that the device
;  2000          2885  2 ! name length is reasonable.
;  2001          2886  2 !
;  2002          2887  2 CH$MOVE (DSC$K_S_BLN, .DEV_DESC, DEV_NAME);
;  2003          2888  3 IF (.DEV_NAME [DSC$W_LENGTH] LEQ 0)
;  2004          2889  3 OR (.DEV_NAME [DSC$W_LENGTH] GTR 63)
;  2005          2890  2 THEN
;  2006          2891  2     RETURN (SS$_BADPARAM);
;  2007          2892  2 IF NOT PROBER (%REF (0), DEV_NAME [DSC$W_LENGTH], .DEV_NAME [DSC$A_POINTER])
;  2008          2893  2 THEN
;  2009          2894  2     RETURN (SS$_ACCVIO);
;  2010          2895  2
;  2011          2896  2 !
;  2012          2897  2 ! Copy the new device name to the mount data base,
;  2013          2898  2 ! and update the descriptor in DEVICE_STRING.
;  2014          2899  2 !
;  2015          2900  2 DEVICE_STRING [(.DEV_INDEX*2)] = .DEV_NAME [DSC$W_LENGTH];
;  2016          2901  2 CH$MOVE (.DEV_NAME [DSC$W_LENGTH],
;  2017          2902  2           .DEV_NAME [DSC$A_POINTER],
;  2018          2903  2           .DEVICE_STRING [(.DEV_INDEX*2)+1]
;  2019          2904  2          );
;  2020          2905  2
;  2021          2906  2 SS$_NORMAL
;  2022          2907  2
;  2023          2908  1 END;                                    ! End of $COPY_INFO


                                                .EXTRN  DEVICE_COUNT, DEVICE_STRING

                                 007C 00000     .ENTRY  $COPY_INFO, Save R2,R3,R4,R5,R6        ;  2820
                          5E     08   C2 00002  SUBL2   #8, SP                                 ;
                          56  04 AC   D0 00005  MOVL    DEV_INDEX, R6                          ;  2871
                              21   19 00009     BLSS    1$                                     ;
                    50    0000G CF    01   C3 0000B  SUBL3   #1, DEVICE_COUNT, R0             ;
                          50    56   D1 00011  CMPL    R6, R0                                 ;
                                16   14 00014  BGTR    1$                                     ;
                08  BC    08    00   0C 00016  PROBER  #0, #8, @DEV_DESC                      ;  2878
                                1A   13 0001B  BEQL    3$                                     ;
                    6E    08 BC   08   28 0001D  MOVC3   #8, @DEV_DESC, DEV_NAME              ;  2887
                          51    6E   3C 00022  MOVZWL  DEV_NAME, R1                           ;  2888
                                05   15 00025  BLEQ    1$                                     ;
                          3F    51   B1 00027  CMPW    R1, #63                                ;  2889
                                04   1B 0002A  BLEQU   2$                                     ;
                          50    14   D0 0002C  1$:    MOVL    #20, R0                        ;  2891
                                     04 0002F  RET                                           ;
                04  BE    6E    00   0C 00030  2$:    PROBER  #0, DEV_NAME, @DEV_NAME+4      ;  2892
                                04   12 00035  BNEQ    4$                                     ;
                          50    0C   D0 00037  3$:    MOVL    #12, R0                        ;  2894
                                     04 0003A  RET                                           ;
```

```
                50              56          01  78 0003B 4$:    ASHL    #1, R6, R0                          ; 2900
                          0000GCF40          51  D0 0003F        MOVL    R1, DEVICE_STRING[R0]               ;
                                50    0000GCF40  D0 00045        MOVL    DEVICE_STRING+4[R0], P0             ; 2903
                60        04    BE          51  28 0004B        MOVC3   R1, @DEV_NAME+4, (R0)               ;
                                50          01  D0 00050        MOVL    #1, R0                              ; 2908
                                                04 00053        RET                                         ;

; Routine Size:  84 bytes,    Routine Base:  $CODE$ + 090F
```

```
;  2025     2909  1 GLOBAL ROUTINE $CHANGE_PROT =
;  2026     2910  1
;  2027     2911  1 !++
;  2028     2912  1 ! Functional description:
;  2029     2913  1 !
;  2030     2914  1 !       This routine will change the page protection of this module's
;  2031     2915  1 !       OWN storage so that it may be written to in USER mode.
;  2032     2916  1 !
;  2033     2917  1 ! Input:
;  2034     2918  1 !
;  2035     2919  1 !       None.
;  2036     2920  1 !
;  2037     2921  1 ! Output:
;  2038     2922  1 !
;  2039     2923  1 !       None.
;  2040     2924  1 !
;  2041     2925  1 ! Implicit input:
;  2042     2926  1 !
;  2043     2927  1 !       1) The current access mode is EXEC or KERNEL.
;  2044     2928  1 !       2) VA_RANGE is a vector of two longword elements, containing the starting
;  2045     2929  1 !          and ending virtual addresses of the range of pages to work on.
;  2046     2930  1 !
;  2047     2931  1 ! Implicit output:
;  2048     2932  1 !
;  2049     2933  1 !       The pages are made USER readable.
;  2050     2934  1 !
;  2051     2935  1 ! Routine value:
;  2052     2936  1 !
;  2053     2937  1 !       Whatever status value is returned by $SETPRT.
;  2054     2938  1 !--
;  2055     2939  1
;  2056     2940  2 BEGIN                                           ! Start of $CHANGE_PROT
;  2057     2941  2
;  2058     2942  2 EXTERNAL
;  2059     2943  2         DEVICE_INDEX,                           ! Index into PHYS_NAME bblock
;  2060     2944  2         DATA_BASE_READY,                        ! Boolean
;  2061     2945  2         STORED_CONTEXT;                         ! Bit vector
;  2062     2946  2
;  2063     2947  2
;  2064     2948  2 !
;  2065     2949  2 ! Initialize three important variables referenced in VMOUNT.  This
;  2066     2950  2 ! must be done here as they are zeroed only once per $MOUNT call,
;  2067     2951  2 ! and must be written while in EXEC mode.
;  2068     2952  2 !
;  2069     2953  2 DEVICE_INDEX = 0;
;  2070     2954  2 DATA_BASE_READY = 0;
;  2071     2955  2 STORED_CONTEXT = 0;
;  2072     2956  2
;  2073     2957  2 !
;  2074     2958  2 ! Set the page protection of this module's data to allow user
;  2075     2959  2 ! mode read/write access.  This must be done here, in EXEC mode, since
;  2076     2960  2 ! this image is INSTALLed as a protected shareable image, which has
;  2077     2961  2 ! the effect of setting the protection to be USER read, EXEC write.
;  2078     2962  2 ! Note that the data sits in a special PSECT, to avoid changing
;  2079     2963  2 ! the page protection on adjacent pages.
;  2080     2964  2 !
;  2081     2965  3 $SETPRT (INADR=VA_RANGE, PROT=PRT$C_UW)
```

```
; 2082          2966  3
; 2083          2967  1 END;                                    ! End of $CHANGE_PROT


                                                               .EXTRN  DATA_BASE_READY
                                                               .EXTRN  STORED_CONTEXT, SYS$SETPRT

                                                 0000 00000    .ENTRY  $CHANGE_PROT, Save nothing                      ; 2909
                                          0000G  CF   D4 00002 CLRL    DEVICE_INDEX                                    ; 2953
                                          0000G  CF   D4 00006 CLRL    DATA_BASE_READY                                 ; 2954
                                          0000G  CF   D4 0000A CLRL    STORED_CONTEXT                                  ; 2955
                                     7E          04   7D 0000E MOVQ    #4, -(SP)                                       ; 2965
                                                 7E   7C 00011 CLRQ    -(SP)                                           ;
                                          0000'  CF   9F 00013 PUSHAB  VA_RANGE                                        ;
                              00000000G  00          05   FB 00017 CALLS   #5, SYS$SETPRT                             ;
                                                     04 0001E    RET                                                   ; 2967

; Routine Size:  31 bytes,    Routine Base:  $CODE$ + 0963
```

```
;  2085          2968  1 GLOBAL ROUTINE $DALLOC_DEVS (SINGLE_DEVICE) =
;  2086          2969  1
;  2087          2970  1 !++
;  2088          2971  1 ! Functional description:
;  2089          2972  1 !
;  2090          2973  1 !       This routine will attempt to dealocate all devices that were
;  2091          2974  1 !       specified by the user that were not previously allocated.
;  2092          2975  1 !
;  2093          2976  1 ! Input:
;  2094          2977  1 !
;  2095          2978  1 !       SINGLE_DEVICE   : a longword boolean to  control whether all
;  2096          2979  1 !                         drives or just a single one is to be deallocated.
;  2097          2980  1 !                         If the latter, use DEVICE_INDEX to select the
;  2098          2981  1 !                         drive name from the PHYS_NAME vector.
;  2099          2982  1 !
;  2100          2983  1 ! Output:
;  2101          2984  1 !
;  2102          2985  1 !       None.
;  2103          2986  1 !
;  2104          2987  1 ! Implicit input:
;  2105          2988  1 !
;  2106          2989  1 !       CLEANUP_ALLOC   : a bit vector where each bit represents an
;  2107          2990  1 !                         an entry in PHYS_NAME that was not previously
;  2108          2991  1 !                         allocated by the user.
;  2109          2992  1 !       DEVICE_INDEX    : index into PHYS_NAME vector
;  2110          2993  1 !       PHYS_NAME       : a vector of device name descriptors for all
;  2111          2994  1 !                         devices specified by the user.
;  2112          2995  1 !       PHYS_COUNT      : a high-water mark that indicates the number
;  2113          2996  1 !                         of devices actually used in the mount.
;  2114          2997  1 !
;  2115          2998  1 ! Implicit output:
;  2116          2999  1 !
;  2117          3000  1 !       All devices not mounted or not previously allocated are deallocated.
;  2118          3001  1 !
;  2119          3002  1 ! Routine value:
;  2120          3003  1 !
;  2121          3004  1 !       SS$_NORMAL      : Normal successful completion
;  2122          3005  1 !--
;  2123          3006  1
;  2124          3007  2 BEGIN                                           ! Start of $DALLOC_DEVS
;  2125          3008  2
;  2126          3009  2 EXTERNAL
;  2127          3010  2         CLEANUP_ALLOC   : BITVECTOR VOLATILE,  ! cleanup bit vector
;  2128          3011  2         DEV_ALLOCATED   : BITVECTOR VOLATILE,  ! device allready allocated
;  2129          3012  2         DEV_ACQUIRED    : BITVECTOR VOLATILE,  ! device is interlocked
;  2130          3013  2         DEVICE_INDEX    : LONG,                ! current device
;  2131          3014  2         PHYS_COUNT      : LONG,                ! count of physical devices
;  2132          3015  2         PHYS_NAME       : VECTOR VOLATILE,     ! device descriptor array
;  2133          3016  2         MOUNT_OPTIONS   : BITVECTOR,           ! mount options and modifiers
;  2134          3017  2         STORED_CONTEXT  : BITVECTOR;           ! special mount context
;  2135          3018  2
;  2136          3019  2
;  2137          3020  2 IF .SINGLE_DEVICE
;  2138          3021  2 THEN
;  2139          3022  2     !
;  2140          3023  2     ! Deallocate a specific device.  This is used to deallocate a
;  2141          3024  2     ! previously allocated device when the operator instructs us to
```

```
;  2142          3025  2      ! substitute another device in its place.
;  2143          3026  2      !
;  2144          3027  3      BEGIN
;  2145          3028  3      IF .CLEANUP_ALLOC[.DEVICE_INDEX]
;  2146          3029  3      THEN
;  2147          3030  4          BEGIN
;  2148          3031  4          $DALLOC(DEVNAM = PHYS_NAME[.DEVICE_INDEX*2]);
;  2149          3032  4          CLEANUP_ALLOC[.DEVICE_INDEX] = 0;
;  2150          3033  3          END;
;  2151          3034  3      DEV_ALLOCATED[.DEVICE_INDEX] = 0;
;  2152          3035  3      DEV_ACQUIRED[.DEVICE_INDEX] = 0;
;  2153          3036  3      PHYS_COUNT = .DEVICE_INDEX;
;  2154          3037  3      END
;  2155          3038  2  ELSE
;  2156          3039  3      BEGIN
;  2157          3040  3      !
;  2158          3041  3      ! Deallocate every device listed in the PHYS_NAME device name descriptor
;  2159          3042  3      ! array, that was not previously allocated by the user.  If the device is
;  2160          3043  3      ! mounted, it will not be deallocated (this check is done in the $DALLOC
;  2161          3044  3      ! system service).
;  2162          3045  3      !
;  2163          3046  3      INCR I FROM 0 TO .PHYS_COUNT-1 DO
;  2164          3047  3          IF .CLEANUP_ALLOC[.I]
;  2165          3048  3          THEN
;  2166          3049  4              BEGIN
;  2167          3050  4              $DALLOC(DEVNAM = PHYS_NAME[.I*2]);
;  2168          3051  4              DEV_ALLOCATED[.I] = 0;
;  2169          3052  4              DEV_ACQUIRED[.I] = 0;
;  2170          3053  4              CLEANUP_ALLOC[.I] = 0;
;  2171          3054  3              END;
;  2172          3055  2      END;
;  2173          3056  2
;  2174          3057  2  SS$_NORMAL
;  2175          3058  2
;  2176          3059  1  END;                                    ! End of $DALLOC_DEVS


                                                                .EXTRN  CLEANUP_ALLOC, DEV_ALLOCATED
                                                                .EXTRN  DEV_ACQUIRED, PHYS_COUNT
                                                                .EXTRN  SYS$DALLOC

                                                007C 00000      .ENTRY  $DALLOC_DEVS, Save R2,R3,R4,R5,R6       ; 2968
                                    56      0000G   CF   9E 00002  MOVAB   DEVICE_INDEX, R6                    ;
                                    55  00000000G   00   9E 00007  MOVAB   SYS$DALLOC, R5                      ;
                                    54      0000G   CF   9E 0000E  MOVAB   CLEANUP_ALLOC, R4                   ;
                                    2C        04    AC   E9 00013  BLBC    SINGLE_DEVICE, 4$                  ; 3020
                              12            64          66   E1 00017  BBC     DEVICE_INDEX, CLEANUP_ALLOC, 1$    ; 3028
                                                        7E   D4 0001B  CLRL    -(SP)                              ; 3031
                              50            66          01   78 0001D  ASHL    #1, DEVICE_INDEX, R0               ;
                                                0000GCF 40   DF 00021  PUSHAL  PHYS_NAME[R0]                      ;
                                    65                  02   FB 00026  CALLS   #2, SYS$DALLOC                  ;
                              00            64          66   E5 00029  BBCC    DEVICE_INDEX, CLEANUP_ALLOC, 1$    ; 3032
                                    50                  66   D0 0002D  1$:    MOVL    DEVICE_INDEX, R0           ; 3034
                              00    0000G   CF          50   E5 00030  BBCC    R0, DEV_ALLOCATED, 2$              ;
                              00    0000G   CF          50   E5 00036  2$:    BBCC    R0, DEV_ACQUIRED, 3$       ; 3035
                                    0000G   CF          50   D0 0003C  3$:    MOVL    R0, PHYS_COUNT             ; 3036
```

```
                                        30  11 00041          BRB     9$                                      ; 3020
                                53      CF  D0 00043 4$:      MOVL    PHYS_COUNT, R3                          ; 3046
                                52      01  CE 00048          MNEGL   #1, I
                                        22  11 0004B          BRB     8$
                        1E              52  E1 0004D 5$:      BBC     I, CLEANUP_ALLOC, 8$                    ; 3047
                                64      7E  D4 00051          CLRL    -(SP)                                   ; 3050
                        50              01  78 00053          ASHL    #1, I, R0
                                0000GCF 40  DF 00057          PUSHAL  PHYS_NAME[R0]
                                65      02  FB 0005C          CALLS   #2, SYS$DALLOC
                        00      0000G   CF  52 E5 0005F          BBCC    I, DEV_ALLOCATED, 6$                ; 3051
                        00      0000G   CF  52 E5 00065 6$:      BBCC    I, DEV_ACQUIRED, 7$                 ; 3052
                        00              64  52 E5 0006B 7$:      BBCC    I, CLEANUP_ALLOC, 8$                ; 3053
                        DA              52  53 F2 0006F 8$:      AOBLSS  R3, I, 5$                           ; 3047
                                50      01  D0 00073 9$:      MOVL    #1, R0                                  ; 3059
                                        04  00076          RET                                             ;

; Routine Size:  119 bytes,    Routine Base:  $CODE$ + 0982
```

```
;  2178     3060  1 ROUTINE EXIT_HANDLER : NOVALUE =
;  2179     3061  1
;  2180     3062  1 !++
;  2181     3063  1 ! Fucntional Description:
;  2182     3064  1 !
;  2183     3065  1 !       This routine is called by the OS on exit (for whatever reason) from
;  2184     3066  1 !       the MOUNT facility.  This routine will clean up any mess left by MOUNT.
;  2185     3067  1 !
;  2186     3068  1 ! Input Parameters:
;  2187     3069  1 !       none
;  2188     3070  1 !
;  2189     3071  1 ! Implicit Inputs:
;  2190     3072  1 !       none
;  2191     3073  1 !
;  2192     3074  1 ! Output Parameters:
;  2193     3075  1 !       none
;  2194     3076  1 !
;  2195     3077  1 ! Implicit Outputs:
;  2196     3078  1 !       none
;  2197     3079  1 !
;  2198     3080  1 !--
;  2199     3081  1
;  2200     3082  2 BEGIN                                           ! Start of EXIT_HANDLER
;  2201     3083  2
;  2202     3084  2 EXTERNAL ROUTINE
;  2203     3085  2         $DALLOC_DEVS$U  : ADDRESSING_MODE (GENERAL);    ! Address of transfer vector
;  2204     3086  2
;  2205     3087  2 !
;  2206     3088  2 ! Attempt to deallocate devices that are not mounted and
;  2207     3089  2 ! were not previously allocated.
;  2208     3090  2 !
;  2209     3091  2 $DALLOC_DEVS$U (0);                             ! Attempt to deallocate devices
;  2210     3092  2
;  2211     3093  2 IF .REPLY_PENDING
;  2212     3094  2 THEN
;  2213     3095  2     !
;  2214     3096  2     ! Cancel any outstanding operator requests.
;  2215     3097  2     !
;  2216     3098  2     CANCEL_REQUEST (REQUEST_NOT_SATISFIED);
;  2217     3099  2
;  2218     3100  2 $SETSFM (ENBFLG = .SS_FAIL_MODE);
;  2219     3101  2
;  2220     3102  1 END;                                           ! End of EXIT_HANDLER


                                0000 00000 EXIT_HANDLER:
                                           .WORD   Save nothing                    ;3060
                                7E  D4 00002           CLRL    -(SP)                    ;3091
00000000G   00                  01  FB 00004           CALLS   #1, $DALLOC_DEVS$U       ;
                07      0000'   CF  E9 0000B           BLBC    REPLY_PENDING, 1$        ;3093
                                7E  D4 00010           CLRL    -(SP)                    ;3098
    F9AA        CF              01  FB 00012           CALLS   #1, CANCEL_REQUEST       ;
                        0000'   CF  DD 00017 1$:       PUSHL   SS_FAIL_MODE             ;3100
00000000G   00                  01  FB 0001B           CALLS   #1, SYS$SETSFM           ;
```

```
                        04 00022          RET                                   ; 3102

; Routine Size:  35 bytes,    Routine Base:  $CODE$ + 09F9
```

```
;  2222          3103  1 !
;  2223          3104  1 ! The TPARSE tables are here because they mangle
;  2224          3105  1 ! PSECT definitions.
;  2225          3106  1 !
;  2226          3107  1
;  2227          3108  1 !
;  2228          3109  1 ! Define the TPARSE grammar of the possible operator replies.
;  2229          3110  1 !
;  2230          3111  1 $INIT_STATE     (STATE_TABLE,KEY_TABLE);
;  2231          3112  1
;  2232          3113  1 !
;  2233          3114  1 ! Initial state
;  2234          3115  1 !
;  2235        P 3116  1 $STATE  (START,
;  2236        P 3117  1          ((SUBSTITUTE_COMMAND),  TPA$_EXIT,      DO_SUBSTITUTE),
;  2237        P 3118  1          (TPA$_LAMBDA,           TPA$_EXIT,      INVALID_COMMAND)
;  2238          3119  1          );
;  2239          3120  1
;  2240          3121  1 !
;  2241          3122  1 ! SUBSTITUTE command.  'SUBSTITUTE'<TPA$_BLANK><DEVICE><TEXT>
;  2242          3123  1 !
;  2243        P 3124  1 $STATE  (SUBSTITUTE_COMMAND,
;  2244        P 3125  1          ('SUBSTITUTE')
;  2245          3126  1          );
;  2246          3127  1
;  2247          3128  1 !$STATE (,
;  2248          3129  1 !        (TPA$_BLANK)
;  2249          3130  1 !        );
;  2250          3131  1
;  2251        P 3132  1 $STATE  (,
;  2252        P 3133  1          ((DEVICE),              TPA$_EXIT,      SAVE_DEVICE)
;  2253          3134  1          );
;  2254          3135  1
;  2255          3136  1 !$STATE (,
;  2256          3137  1 !        ((TEXT),                TPA$_EXIT)
;  2257          3138  1 !        );
;  2258          3139  1
;  2259          3140  1 !
;  2260          3141  1 ! Device name.  It may be a device spec or a logical name string.
;  2261          3142  1 !
;  2262        P 3143  1 $STATE  (DEVICE,
;  2263        P 3144  1          (TPA$_SYMBOL)
;  2264          3145  1          );
;  2265          3146  1
;  2266        P 3147  1 $STATE  (,
;  2267        P 3148  1          (':',                   TPA$_EXIT),
;  2268        P 3149  1          (TPA$_LAMBDA,           TPA$_EXIT)
;  2269          3150  1          );
;  2270          3151  1 !
;  2271          3152  1 ! Text.  The remainder of the operator response is treated
;  2272          3153  1 ! as a comment, and has no effect on the mount.  If there is
;  2273          3154  1 ! a comment, at least one blank must separate it from the
;  2274          3155  1 ! previous section of the operator response.
;  2275          3156  1 !
;  2276        P 3157  1 $STATE  (TEXT,
;  2277        P 3158  1          (TPA$_BLANK,            MORE_TEXT),
;  2278        P 3159  1          (TPA$_EOS,              TPA$_EXIT)
```

```
;  2279          3160  1          );
;  2280          3161  1
;  2281        P 3162  1 $STATE   (MORE_TEXT,
;  2282        P 3163  1          (TPA$_ANY,              MORE_TEXT),
;  2283        P 3164  1          (TPA$_EOS,              TPA$_EXIT)
;  2284          3165  1          );
;  2285          3166  1 END
;  2286          3167  0 ELUDOM


                                                .PSECT  _LIB$KEY1$,NOWRT,  SHR,  PIC,1

                                          00000 ;TPA$KEYSTO
                                                U.10:   .BLKB   0
                45  54  55  54  49  54  53  42  55  53  00000 ;TPA$KEYST
                                                U.12:   .ASCII  \SUBSTITUTE\                    ;
                                            FF  0000A           .BYTE   -1                      ;
                                            FF  0000B ;TPA$KEYFILL
                                                U.14:   .BYTE   -1                              ;

                                                .PSECT  _LIB$STATE$,NOWRT,  SHR,  PIC,1

                                          00000 STATE_TABLE::
                                                .BLKB   0
                                          00000 START:  .BLKB   0
                                        99F8  00000 ;TPA$TYPE
                                                U.2:    .WORD   -26120                          ;
                                        0000* 00002 ;TPA$SUBEXP
                                                U.4:    .WORD   <<U.3-U.4>-2>                   ;
                                    00000000* 00004 ;TPA$ACTION
                                                U.5:    .LONG   <<DO_SUBSTITUTE-U.5>-4>         ;
                                        FFFF  00008 ;TPA$TARGET
                                                U.6:    .WORD   -1                              ;
                                        95F6  0000A ;TPA$TYPE
                                                U.7:    .WORD   -27146                          ;
                                    00000000* 0000C ;TPA$ACTION
                                                U.8:    .LONG   <<INVALID_COMMAND-U.8>-4>       ;
                                        FFFF  00010 ;TPA$TARGET
                                                U.9:    .WORD   -1                              ;
                                              00012 ;SUBSTITUTE_COMMAND
                                                U.3:    .BLKB   0
                                        0500  00012 ;TPA$TYPE
                                                U.13:   .WORD   1280                            ;
                                        9DF8  00014 ;TPA$TYPE
                                                U.15:   .WORD   -25096                          ;
                                        0000* 00016 ;TPA$SUBEXP
                                                U.17:   .WORD   <<U.16-U.17>-2>                 ;
                                    00000000* 00018 ;TPA$ACTION
                                                U.18:   .LONG   <<SAVE_DEVICE-U.18>-4>          ;
                                        FFFF  0001C ;TPA$TARGET
                                                U.19:   .WORD   -1                              ;
                                              0001E ;DEVICE
                                                U.16:   .BLKB   0
                                        05F1  0001E ;TPA$TYPE
                                                U.20:   .WORD   1521                            ;
                                        103A  00020 ;TPA$TYPE
```

```
                        U.21:   .WORD   4154                                    ;
FFFF    00022 ;TPA$TARGET
                        U.22:   .WORD   -1                                      ;
15F6    00024 ;TPA$TYPE
                        U.23:   .WORD   5622                                    ;
FFFF    00026 ;TPA$TARGET
                        U.24:   .WORD   -1                                      ;
              00028 TEXT:   .BLKB   0
11F2    00028 ;TPA$TYPE
                        U.25:   .WORD   4594                                    ;
0000*   0002A ;TPA$TARGET
                        U.27:   .WORD   <<U.26-U.27>-2>                         ;
15F7    0002C ;TPA$TYPE
                        U.28:   .WORD   5623                                    ;
FFFF    0002E ;TPA$TARGET
                        U.29:   .WORD   -1                                      ;
              00030 ;MORE_TEXT
                        U.26:   .BLKB   0
11ED    00030 ;TPA$TYPE
                        U.30:   .WORD   4589                                    ;
0000*   00032 ;TPA$TARGET
                        U.31:   .WORD   <<U.26-U.31>-2>                         ;
15F7    00034 ;TPA$TYPE
                        U.32:   .WORD   5623                                    ;
FFFF    00036 ;TPA$TARGET
                        U.33:   .WORD   -1                                      ;

                                .PSECT  _LIB$KEY0$,NOWRT,  SHR,  PIC,1

                    00000 KEY_TABLE::
                                .BLKB   0
                    00000 ;TPA$KEY0
                        U.1:    .BLKB   0
                0000*   00000 ;TPA$KEY
                        U.11:   .WORD   <U.10-U.1>                              ;

                                .EXTRN  LIB$SIGNAL, LIB$STOP

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;   $USER_DATA$                 1032  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(9)
;   $PLIT$                       140  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $CODE$                      2588  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   . ABS .                        0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)
;   $GLOBAL$                       8  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   _LIB$KEY0$                     2  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;   _LIB$STATE$                   56  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;   _LIB$KEY1$                    12  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)


;                           Library Statistics
;
```

```
;                                  -------- Symbols --------      Pages       Processing
;       File                       Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1    18619       113         0        1000        00:02.0
;  _$255$DUA28:[SYSLIB]TPAMAC.L32;1    42        27        64          14        00:00.2
```

```
;                                  COMMAND QUALIFIERS

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

; Size:         2588 code + 1250 data bytes
; Run Time:        01:01.8
; Elapsed Time:    02:09.2
; Lines/CPU Min:    3077
; Lexemes/CPU-Min: 33437
; Memory Used:  233 pages
; Compilation Complete
```

BINDVL
LIS

MOUNT

SYSFAO
LIS

MOUNTSHR
MAP

TEMPLATE
LIS

VMOUNT
MAP

ASSIST
LIS

ALLOCM
LIS

MOUDEF
B32