C 6

```
MMM         MMM     OOOOOOOOO     UUU         UUU  NNN         NNN  TTTTTTTTTTTTTTT
MMM         MMM     OOOOOOOOO     UUU         UUU  NNN         NNN  TTTTTTTTTTTTTTT
MMM         MMM     OOOOOOOOO     UUU         UUU  NNN         NNN  TTTTTTTTTTTTTTT
MMMMMM   MMMMMM  OOO         OOO  UUU         UUU  NNN         NNN        TTT
MMMMMM   MMMMMM  OOO         OOO  UUU         UUU  NNN         NNN        TTT
MMMMMM   MMMMMM  OOO         OOO  UUU         UUU  NNN         NNN        TTT
MMM   MMM   MMM  OOO         OOO  UUU         UUU  NNNNNN      NNN        TTT
MMM   MMM   MMM  OOO         OOO  UUU         UUU  NNNNNN      NNN        TTT
MMM   MMM   MMM  OOO         OOO  UUU         UUU  NNNNNN      NNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN   NNN   NNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN   NNN   NNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN   NNN   NNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN      NNNNNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN      NNNNNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN      NNNNNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN         NNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN         NNN        TTT
MMM         MMM  OOO         OOO  UUU         UUU  NNN         NNN        TTT
MMM         MMM     OOOOOOOOO     UUUUUUUUUUUUUUU  NNN         NNN        TTT
MMM         MMM     OOOOOOOOO     UUUUUUUUUUUUUUU  NNN         NNN        TTT
MMM         MMM     OOOOOOOOO     UUUUUUUUUUUUUUU  NNN         NNN        TTT
```

```
**FILE**ID**ASSIST                                F 10
```

```
  AAAAAA      SSSSSSSS    SSSSSSSS    IIIIII      SSSSSSSS  TTTTTTTTTT
  AAAAAA      SSSSSSSS    SSSSSSSS    IIIIII      SSSSSSSS  TTTTTTTTTT
AA      AA  SS          SS              II      SS              TT
AA      AA  SS          SS              II      SS              TT
AA      AA  SS          SS              II      SS              TT
AA      AA  SS          SS              II      SS              TT
AA      AA    SSSSSS      SSSSSS        II        SSSSSS        TT
AA      AA    SSSSSS      SSSSSS        II        SSSSSS        TT
AAAAAAAAAA          SS          SS      II              SS      TT
AAAAAAAAAA          SS          SS      II              SS      TT
AA      AA          SS          SS      II              SS      TT          ....
AA      AA          SS          SS      II              SS      TT          ....
AA      AA  SSSSSSSS    SSSSSSSS      IIIIII    SSSSSSSS        TT          ....
AA      AA  SSSSSSSS    SSSSSSSS      IIIIII    SSSSSSSS        TT          ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
ASS
VO4
```

```
MODULE ASSIST   (
                LANGUAGE (BLISS32),
                IDENT = 'V04-001'
                ) =

BEGIN

!****************************************************************************
!*                                                                          *
!*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
!*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
!*  ALL RIGHTS RESERVED.                                                    *
!*                                                                          *
!*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
!*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
!*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
!*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
!*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
!*  TRANSFERRED.                                                            *
!*                                                                          *
!*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
!*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
!*  CORPORATION.                                                            *
!*                                                                          *
!*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
!*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************

!++
!
! FACILITY:
!
!       MOUNT
!
! ABSTRACT:
!
!       This module contains the routines to
!       implement operator assisted mount.
!
! ENVIRONMENT:
!
!       VAX/VMS operating system.
!
! AUTHOR:
!
!       Steven T. Jeffreys
!
! CREATION DATE:
!
!       October 9, 1980
!
! MODIFIED BY:
!
!       V04-001 HH0056          Hai Huang               11-Sep-1984
!               Do limited number of retries on VOLINV error.
```

```
 1 !
 1 !      V03-007 HH0041          Hai Huang               24-Jul-1984
 1 !              Remove REQUIRE  'LIBD$:[VMSLIB.OBJ]MOUNTMSG.B32'.
 1 !
 1 !      V03-006 CWH3001         CW Hobbs                        30-Jul-1983
 1 !              Various and sundry things to make OPCOM distributed
 1 !              across the cluster.
 1 !
 1 !      V03-005 TCM0002         Trudy C. Matthews       28-Jul-1983
 1 !              Add DEV_ACQUIRED flag that indicates whether mount interlock
 1 !              has been taken out for this device.  Remove DEALLOCATE_DEVICE
 1 !              routine, since devices mounted /SHARE, /SYSTEM or /GROUP are
 1 !              no longer allocated.  Remove temporary change introduced in
 1 !              TCM0001.
 1 !
 1 !      V03-004 TCM0001         Trudy C. Matthews       18-Jul-1983
 1 !              Make SS$_NOTQUEUED status (received from the $ENQ system
 1 !              service when we cannot take out a cluster-wide allocation
 1 !              lock on this device) one of the status codes acted on by
 1 !              operator-assisted mount.
 1 !
 1 !      V03-003 STJ50311        Steven T. Jeffreys      10-Feb-1983
 1 !              - Make all uses of PHYS_NAME indexed by DEVICE_INDEX.
 1 !              - Reset PREVIOUS_STATUS after an operator reply arrives.
 1 !              - If the mount failed with an operator request outstanding,
 1 !                signal MOUN$_OPRQSTCAN instead of MOUN$_RQSTDON.
 1 !              - Define and use routine $DALLOC_DEVS.
 1 !
 1 !      V03-002 STJ0244         Steven T. Jeffreys      04-Apr-1982
 1 !              - Use common I/O routines, and make the code more
 1 !                tolerant to random event flag setting and clearing.
 1 !              - Issue the MOUN$_RQSTDON status if the mount completes
 1 !                successfully while we have an operator request outstanding.
 1 !
 1 !      V03-001 BLS0160         Benn Schreiber          18-Mar-1982
 1 !              Get OPCDEFTMP from SHRLIB$.
 1 !
 1 !      V02-011 STJ0229         Steven T. Jeffreys      01-Mar-1982
 1 !              - Set the inhibit message bit in the exit status
 1 !                code if the message output via $PUTMSG.
 1 !
 1 !      V02-010 STJ0218         Steven T. Jeffreys      16-Feb-1982
 1 !              - Cancel exit handler before declaring it.
 1 !              - Clear system service failure exception mode and
 1 !                restore it on exit.
 1 !
 1 !      V02-009 STJ0214         Steven T. Jeffreys      11-Feb-1982
 1 !              Add support for the /COMMENT switch.
 1 !
 1 !      V02-008 STJ0206         Steven T. Jeffreys      08-Feb-1982
 1 !              Set mailbox access rights to allow SYSTEM and OWNER
 1 !              read and write privileges.
 1 !
 1 !      V02-007 STJ0189         Steven T. Jeffreys      02-Feb-1982
 1 !              Initalize GLOBAL storage at run time, and fix various bugs.
 1 !
 1 !      V02-006 STJ174          Steven T. Jeffreys      19-Jan-1982
```

```
1 !                     Made most of the GLOBAL routines in to local routines.
1 !
1 !             V02-005 STJ162          Steven T. Jeffreys      04-Jan-1982
1 !                     Removed copy of INTERCEPT_SIGNAL.
1 !
1 !             V02-004 STJ0150         Steven T. Jeffreys
1 !                     Extensive rewrite to support the $MOUNT system service.
1 !
1 !             V02-003 STJ0112         Steven T. Jeffreys
1 !                     - Use general addressing mode for library routines.
1 !                     - Fixed SET_TARGET_MASK.
1 !                     - Fixed SUBMIT_REQUEST to calculate actual message size.
1 !                     - Added support for alternate cancellation message.
1 !                     - Handle REPLY/BLANK_TAPE and REPLY/INITIALIZE_TAPE operator replies.
1 !
1 !             V02-002 STJ0083         Steven T. Jeffreys
1 !                     - Changed $DELMBX call in CANCEL_REQUEST to $DASSGN to properly
1 !                       delete the mailbox and free up the channel.
1 !                     - Changed error recovery handlers to use the physical device
1 !                        name string when referring to the device.
1 !                     - Added logic to recover from an SS$_INCVOLLABEL error, which
1 !                       occurs when the label of the volume present in the drive does
1 !                       not match the volume label specified by the user.
1 !
1 !--
1
1 LIBRARY 'SYS$LIBRARY:LIB.L32';
1 LIBRARY 'SYS$LIBRARY:TPAMAC';
1 REQUIRE 'LIBD$:[VMSLIB.OBJ]INITMSG.REQ';
1 REQUIRE 'SHRLIB$:OPCDEFTMP';     ! *** TEMPORARY
1 REQUIRE 'SRC$:MOUDEF.B32';
1
1 FORWARD ROUTINE
1
1               SYS$MOUNT,                              ! Main entry point of $MOUNT
1               INTERCEPT_SIGNAL,                       ! Main condition handler
1               SUBMIT_REQUEST   : NOVALUE,             ! Send request to operator
1               SET_TARGET_MASK  : NOVALUE,             ! Sets operator target mask
1               POST_READ_TO_MBX: NOVALUE,              ! Post read to reply mailbox
1               INTERACTIVE_JOB,                        ! Determines if we're a batch job
1               PRINT_REPLY      : NOVALUE,             ! Print the operator reply
1               PARSE_REPLY      : NOVALUE,             ! Parse the oprator's reply
1               CANCEL_REQUEST   : NOVALUE,             ! Cancel the operator request
1               CHECK_FOR_REPLY  : NOVALUE,             ! Check for operator response
1               ALLOCFAIL_HNDLR  : NOVALUE,             ! Handle device allocation failures
1               MEDOFL_HNDLR     : NOVALUE,             ! Handle SS$_MEDOFL condition
1               WRONGVOL_HNDLR   : NOVALUE,             ! Handle SS$_INCVOLLABEL condition
1               INVALID_COMMAND,                        ! Notify user/operator of invalid reply
1               EXIT_HANDLER     : NOVALUE;             ! Exit handler
1
1 FORWARD
1
1               STATE_TABLE      : VECTOR [0],          ! TPARSE state table
1               KEY_TABLE        : VECTOR [0];          ! TPARSE key table
1
1 STRUCTURE
1
```

```
                EXIT_CTRL_BLK [I ; N] =                 ! exit handler descriptor
                    [(4+N)*4]                           ! N = # of arguments ( N <= 1)
                    (EXIT_CTRL_BLK+I*4)<0,32,0>;        ! the block is a longword array

        MACRO
                !
                ! Abort the mount operation.
                !
                ABORT_MOUNT (CODE) =
                        SIGNAL_STOP         (%IF NOT %NULL (CODE) %THEN CODE %ELSE 0 %FI
                                             %IF NOT %NULL (%REMAINING) %THEN , %REMAINING %FI
                                            )%;

        MACRO
                !
                ! Generate a static string descriptor
                !
                DESCRIP (STRING) =
                        BBLOCK  [DSC$K_S_BLN]
                        INITIAL (WORD (%CHARCOUNT (STRING)),
                                 BYTE (DSC$K_DTYPE_T),
                                 BYTE (DSC$K_CLASS_S),
                                 LONG (UPLIT BYTE (STRING))
                                )%;

        MACRO
                !
                ! 3 byte operator mask field definition.
                !
                TARGET_FIELD = $BYTEOFFSET(OPC$B_MS_TARGET), 0, 24, 0%;

        MACRO
                !
                ! For documentation purposes, define a boolean variable
                ! that can only take on the values TRUE or FALSE.
                !
                BOOLEAN = LONG%;

        LITERAL
                FAO_BUFFER_SIZE = 512,          ! Max length of FAO result string
                MAX_DEV_LENGTH  = 63,           ! Max length of device name
                !
                ! Create the reply mailbox protection mask.  Allow only
                ! OWNER(read) and SYSTEM(read,write) access.  See documentation
                ! of the $CREMBX system service for more info.
                !
                MAILBOX_PROTECTION = %X'FF00',
                !
                ! The following are boolean values that are used to make the
                ! code more readable.  They are used as input to CANCEL_REQUEST.
                !
                REQUEST_SATISFIED        = 1,   ! The request completed w/o opertor intervention
                REQUEST_NOT_SATISFIED    = 0,   ! The request is being cancled for some reason
                !
                ! The following are mask definitions used for retrieving specified
                ! portions of a message via the $GETMSG system service.
                !
                MSG_TEXT        = 1,            ! Include message text
                MSG_ID          = 2,            ! Include message identifier
                MSG_SEVERITY    = 4,            ! Include severity indicator
                MSG_FACILITY    = 8,            ! Include message facility name
                !
                ! The following are indexes into the Exit Handler Control Block
                !
                XHNDLR_ADDRESS  = 1,            ! exit handler address
                XHNDLR_ARGCNT   = 2,            ! exit handler argument count
                XHNDLR_STSADDR  = 3,            ! system exit status address
                !
                TRUE            = 1,            ! Boolean value
                FALSE           = 0,            ! Boolean value
                !
                WAIT            = 1,            ! Enable wait for reply
                NO_WAIT         = 0,            ! Disable wait for reply
                !
                REPLY_FLAG      = MOUNT_EFN,    ! A local event flag #
                TIMER_FLAG      = TIMER_EFN,    ! A local event flag #
                TIMER_ID        = 999,          ! Timer identification #
                !
                EXPECT_REPLY    = 1,            ! Indicates that we expect a reply
                NO_REPLY        = 0;            ! Indecates that we don't desire a reply

        GLOBAL LITERAL
                VOLINV_LIMIT    = 20;           ! VOLINV retry limit

        !
        ! Define the static storage used by this module.  Note that the
        ! virtual pages on which this data resides must be USER writable.
        ! It is important that this data start on a page boundary, so that
        ! the $SETPRT call does not make pages writable that were not meant
        ! to be.
        !

        PSECT GLOBAL = $USER_DATA$ (WRITE, NOEXECUTE, NOSHARE, ALIGN (9));

        GLOBAL
                VA_START        : VECTOR [0] ALIGN (9),                 ! Start of 'user data'
                VOLINV_COUNT    : LONG,                 ! VOLINV retry counter
                !
                ! Declare boolean variables.
                !
                REPLY_PENDING   : BOOLEAN VOLATILE,     ! Determines if response outstanding
                MOUNT_FAILED    : BOOLEAN VOLATILE,     ! Used in conjunction with MOUNT_STATUS
                OPERATOR_PRESENT: BOOLEAN VOLATILE,     ! Determines operator presence
                RETRY_COUNTER   : LONG VOLATILE,        ! Number of retries
                SS_FAIL_MODE    : BOOLEAN,              ! System service failure mode
                !
                ! Declare condition context variables.
                !
                MOUNT_STATUS    : BBLOCK[4] VOLATILE,   ! Primary condition
                PREVIOUS_STATUS : BBLOCK[4] VOLATILE,   ! Previous primary condition
                PREVIOUS_DEV_IDX: LONG VOLATILE,        ! Previous device index #
                OPERATOR_MASK   : LONG VOLATILE,        ! Mask of operators to receive requests
                REQUEST_ID      : LONG VOLATILE,        ! Operator request #
                !
```

```
1       ! Declare exit handler control block.
1
1       EXIT_HNDLR_DSC  : EXIT_CTRL_BLK [0],     ! Define exit handler descriptor
1       !
1       ! Declare storge related to the operator reply message.
1
1       REPLY_CHANNEL   : LONG VOLATILE,         ! Channel of reply mailbox
1       REPLY_IOSB      : BBLOCK [8] VOLATILE,   ! IOSB  for operator reply read
1       REPLY_BUFFER    : BBLOCK [OPC$S_MS_OTEXT+8] VOLATILE,
1       REPLY_DESC      : BBLOCK [DSC$K_S_BLN] VOLATILE
1                         INITIAL (WORD (OPC$S_MS_OTEXT+8),
1                                  BYTE (DSC$K_DTYPE_T),
1                                  BYTE (DSC$K_CLASS_S),
1                                  LONG (REPLY_BUFFER)
1                                 ),
1
1       !
1       ! Define the TPARSE control block.
1       !
1       TPARSE_BLOCK    : BBLOCK [TPA$K_LENGTH0]
1                         INITIAL (TPA$K_COUNT0,TPA$M_ABBREV),
1
1       !
1       ! Define the device name descriptor that is used as an implicit
1       ! output to a TPARSE action routine.
1       !
1       DEVICE_DESC     : BBLOCK [DSC$K_S_BLN]   ! Descriptor for device name
1                         INITIAL (WORD (MAX_DEV_LENGTH),
1                                  BYTE (DSC$K_DTYPE_T),
1                                  BYTE (DSC$K_CLASS_S),
1                                  LONG (0)
1                                 ),
1       !
1       ! Declare storage for operator message and its descriptor.
1
1       OP_MSG_BUF      : BBLOCK [OPC$S_MS_OTEXT] ! Buffer for op. request ms
1                         INITIAL (BYTE (OPC$_RQ_RQST)),
1
1       OP_MSG_DESC     : BBLOCK [DSC$K_S_BLN]   ! Descriptor for op. request
1                         INITIAL (WORD (OPC$S_MS_OTEXT),
1                                  BYTE (DSC$K_DTYPE_T),
1                                  BYTE (DSC$K_CLASS_S),
1                                  LONG (OP_MSG_BUF)
1                                 ),
1
1       CANCEL_MSG_BUF  : BBLOCK [OPC$K_HDR_SIZE]                ! Cancel message
1                         INITIAL (BYTE (OPC$_X_CANCEL),         ! Set cancellation code
1                                  BYTE (OPC$K_UNSPEC)           ! Set SCOPE unspecified
1                                 ),
1
1       CANCEL_MSG_DESC : BBLOCK [DSC$K_S_BLN]   ! Cancel message descriptor
1                         INITIAL (WORD (OPC$K_HDR_SIZE),
1                                  BYTE (DSC$K_DTYPE_T),
1                                  BYTE (DSC$K_CLASS_S),
1                                  LONG (CANCEL_MSG_BUF)
1                                 ),
1       !
```

```
        ! Declare storage for FAO resultant string buffer and descriptor.
        !
        FAO_BUFFER        : BBLOCK [FAO_BUFFER_SIZE],
        FAO_RESULT_DESC : BBLOCK [DSC$K_S_BLN]
                          INITIAL (WORD ([LOG$C_NAMLENGTH),
                                   BYTE (DSC$K_DTYPE_T),
                                   BYTE (DSC$K_CLASS_S),
                                   LONG (FAO_BUFFER)
                                  ),
        !
        ! Define the INADR vector used in the $SETPRT call.
        ! Note that VA_RANGE is on the next virtual page after VA_END.
        !
        VA_END          : VECTOR [0],              ! End of 'user data'
        VA_RANGE        : VECTOR [2] INITIAL (VA_START, VA_END) ALIGN (9);


BIND
        !
        ! This is the delta-time value for all timers used.
        ! The time is a quadword value, is currently set for 5 seconds.
        !
        DELTA_TIME      = UPLIT (-5 * 10000000, -1);
```

```
GLOBAL ROUTINE SYS$MOUNT  (ITEM_LIST) =
!++
! Functional description:
!
!       This routine is the main entry point of the $MOUNT system service,
!       and executes in the access mode of the caller.  Usually this will
!       be USER mode.  This routine others defined in this module implement
!       the logic for "operator assisted mount".  This code must execute
!       in  USER mode, to allow users to CTRL\Y out of a mount request.
!
! Input:
!
!       ITEM_LIST          : Address of a $GETJPI-like item list
!
! Output:
!
!       None.
!
! Implicit Inputs:
!
!       The MOUNT data base.
!
! Implicit Outputs:
!
!       The MOUNT data base may be altered as
!       the result of operator intervention.
!
!--

BEGIN                                                   ! Start of OPERATOR_ASSIST

BUILTIN
        FP,
        AP,
        CALLG;

LOCAL
        STATUS;

EXTERNAL
        MOUNT_OPTIONS   : BITVECTOR VOLATILE;   ! Mount options bit vector

EXTERNAL ROUTINE
        $DALLOC_DEVS$U  : ADDRESSING_MODE (GENERAL),    ! Address of transfer vector
        $CHANGE_PROT$U  : ADDRESSING_MODE (GENERAL),    ! Address of the transfer vector
        SYS$VMOUNT$U    : ADDRESSING_MODE (GENERAL);    ! Address of the transfer vector

!
! Enable a condition handler that will force the primary
! condition code facility-code to the MOUNT facility.
!
ENABLE INTERCEPT_SIGNAL;

!
! Set the page protection of this module's data to allow user
! mode read/write access.  This must be done here, since this
! image is INSTALLed as a protected shareable image, which has
```

```
2 ! the effect of setting the protection to be USER read, EXEC write.
2 ! Note that the data sits in a specail PSECT, to avoid changing
2 ! the page protection on adjacent pages.
2 !
3 IF NOT (MOUNT_STATUS = $CHANGE_PROT$U ())
2 THEN
2     RETURN (.MOUNT_STATUS);
2 !
2 ! Initialize the necessary variables.  Most of the
2 ! descriptors are not significantly changed, and do
2 ! not have to be initialized at run time.
2 !
2 REPLY_PENDING = FALSE;
2 MOUNT_FAILED = TRUE;
2 OPERATOR_PRESENT = TRUE;
2 PREVIOUS_STATUS = -1;
2 PREVIOUS_DEV_IDX = -1;
2 RETRY_COUNTER = 0;
2 SS_FAIL_MODE = 0;
2
2 !
2 ! Clear the system service failure exception flag, but save it's state.
2 !
2 STATUS = $SETSFM (ENBFLG=0);
3 IF (.STATUS EQL SS$_WASSET)
2 THEN
2     SS_FAIL_MODE = 1;
2
2 !
2 ! Set up the exit handler descriptor and declare the handler.
2 !
2 EXIT_HNDLR_DSC[XHNDLR_ADDRESS] = EXIT_HANDLER;
2 EXIT_HNDLR_DSC[XHNDLR_ARGCNT]  = 1;
2 EXIT_HNDLR_DSC[XHNDLR_STSADDR] = MOUNT_STATUS;
2 $CANEXH (DESBLK = EXIT_HNDLR_DSC);
2 $DCLEXH (DESBLK=EXIT_HNDLR_DSC);
2
2 !
2 ! Perform the mount request.  If it fails, attempt to recover
2 ! via some operator assistance.  If that is not possible, or the
2 ! operator or user aborts the mount, die gracefully and return the
2 ! status to the user.
2 !
2 MOUNT_STATUS = 0;
2 VOLINV_COUNT = 0;
2 WHILE NOT .MOUNT_STATUS DO
3     BEGIN
4     IF NOT (MOUNT_STATUS = CALLG (.AP, SYS$VMOUNT$U))
3     THEN
3         IF NOT .MOUNT_OPTIONS [OPT_ASSIST]
3         THEN
3
4             BEGIN
4             !
4             ! If the mount operation failed for some reason other than VOLINV,
4             ! exit loop with the error status. Else, do a limited number of
4             ! retries. This automatic retry is implemented due to a race
```

```
4                    ! between mount and mount-verification. If mount is in progress
4                    ! and some event (e.g. cluster state transition) triggers mount-
4                    ! verification, mount-verification will clear the volume-valid
4                    ! bit in the UCB, causing mount to fail with a VOLINV error.
4                    !
4                    ! Not that the VOLINV error message will be suppressed (in module
4                    ! VMOUNT) unless the last retry fails with a VOLINV error.
4                    !
5                   IF (.MOUNT_STATUS AND STS$M_MSG_NO) NEQ (SS$_VOLINV AND STS$M_MSG_NO)
4                   THEN
4                       EXITLOOP;
4                   VOLINV_COUNT = .VOLINV_COUNT + 1;
4                   IF .VOLINV_COUNT GEQ VOLINV_LIMIT
4                   THEN
4                       EXITLOOP;
4                   END
4
3               ELSE
3
4                   BEGIN
4                    !
4                    ! SELECT an error recovery handler based on the mount status value.
4                    ! Use only the message number and the facility code in the comparisons.
4                    !
4                   SELECTONEU (.MOUNT_STATUS AND STS$M_MSG_NO) OF
4                       SET
4                       [SS$_DEVALLOC     AND STS$M_MSG_NO]       : ALLOCFAIL_HNDLR ();
4                       [SS$_MEDOFL       AND STS$M_MSG_NO]       : MEDOFL_HNDLR ();
4                       [SS$_VOLINV       AND STS$M_MSG_NO]       : MEDOFL_HNDLR ();
4                       [SS$_NODEVAVL     AND STS$M_MSG_NO]       : ALLOCFAIL_HNDLR ();
4                       [SS$_NOSUCHDEV    AND STS$M_MSG_NO]       : ALLOCFAIL_HNDLR ();
4                       [SS$_INCVOLLABEL AND STS$M_MSG_NO]        : WRONGVOL_HNDLR ();
4                       [OTHERWISE]                               : EXITLOOP;
4                       TES;
4
4                    !
4                    ! Check for a reply to the operator request.  If it has
4                    ! arrived, it will be processed.  If it hasn't, wait for
4                    ! a few seconds and try again.
4                    !
4                   CHECK_FOR_REPLY ();
3                   END;
2           END;
2
2    !
2    ! Attempt to deallocate devices that are not mounted and
2    ! were not previously allocated.
2    !
2    ! If the mount interlock on this device is still in effect, dequeue it now.
2    !
2    ! Cancel the any outstanding requests and the exit handler.
2    ! Also restore the system service failure exception flag to its
2    ! original state, and disable the condition handler.
2    !
2 $DALLOC_DEVS$U (0);                                    ! Attempt to deallocate devices
2 CANCEL_REQUEST (REQUEST_SATISFIED);
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  00100          .LONG    0                                         ;
                                03  00104 OP_MSG_BUF::
                                                   .BYTE    3                                         ;
                                    00105          .BLKB    127
                              0080  00184 OP_MSG_DESC::
                                                   .WORD    128                                       ;
                                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
```

```
1 ROUTINE INTERCEPT_SIGNAL (SIGNAL, MECHANISM) =
1
1 !++
1 ! Functional Description:
1 !
1 !       This routine is a conditon handler whose sole
1 !       reason for existence is to force the primary
1 !       conditon code's facility-code to that of the
1 !       MOUNT facility.
1 !
1 ! Input:
1 !
1 !       SIGNAL    = Address of the signal array
1 !       MECHANISM = Address of the mechanism array
1 !
1 ! Output:
1 !
1 !       The condition facility code is equal to MOUN$_FACILITY
1 !--
1
2 BEGIN                                           ! Start of INTERCEPT_SIGNAL
2
2 MAP
2
2       SIGNAL          : REF BBLOCK,           ! Signal array
2       MECHANISM       : REF BBLOCK;           ! Mechanism array
2
2 EXTERNAL
2
2       MOUNT_OPTIONS   : BITVECTOR VOLATILE,   ! parser option flags
2       USER_STATUS     : VECTOR;               ! Status return of some routines
2
2
2 IF .SIGNAL[CHF$L_SIG_NAME] NEQ SS$_UNWIND
2 THEN
3     BEGIN
3     !
3     ! Make the facility code MOUN$_FCILITY.
3     !
3     IF .BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_FAC_NO] EQL 0
3     OR .BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_FAC_NO] EQL INIT$_FACILITY
3     THEN
3         BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_FAC_NO] = MOUN$_FACILITY;
3
3     IF .BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_MSG_NO] EQL 0
3     THEN
3         BBLOCK [SIGNAL[CHF$L_SIG_NAME], STS$V_MSG_NO] = .USER_STATUS [0] ^ (-$BITPOSITION (STS$V_MSG_NO));
3
3     !
3     ! If the caller requested it, print the message text associated with the message code.
3     !
3     IF .MOUNT_OPTIONS [OPT_MESSAGE]
3     THEN
4         BEGIN
4         SIGNAL [CHF$L_SIG_ARGS] = .SIGNAL [CHF$L_SIG_ARGS] - 2;
4         $PUTMSG (MSGVEC = SIGNAL [CHF$L_SIG_ARGS], ACTRTN=0, FACNAM=0);
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 00000072  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, #T3, (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
```

```
  1 ROUTINE POST_READ_TO_MBX (MBX_CHANNEL) : NOVALUE =
  1
  1 !++
  1 ! Functional description:
  1 !
  1 !        This routine will post a read to the reply mailbox.
  1 !        Instead of waiting for the I/O to complete, request
  1 !        that an event flag be set when the I/O is finally done.
  1 !
  1 ! Input:
  1 !
  1 !        None.
  1 !
  1 ! Implicit Input:
  1 !
  1 !        REPLY_CHANNEL   : Channel # of channel to the reply mailbox.
  1 !
  1 ! Output:
  1 !
  1 !        None.
  1 !
  1 ! Implict output:
  1 !
  1 !        REPLY_IOSB      : Address of an I/O status block to receive the status of the I/O.
  1 !        REPLY_BUFFER    : Address of buffer to receive the operator's reply.
  1 !
  1 ! Side effects:
  1 !
  1 !        If the $QIO fails, the user will be notified
  1 !        of the failure and the mount will be aborted.
  1 !
  1 ! Routine value:
  1 !
  1 !        None.
  1 !--
  1
  2 BEGIN                                                   ! Start of POST_READ_TO_MBX
  2
  2 LOCAL
  2         STATUS          : LONG;                         ! Hold status of $QIO call
  2
P 3 IF NOT (STATUS = $QIO   (FUNC = IO$_READVBLK,
P 3                          EFN  = REPLY_FLAG,
P 3                          CHAN = .REPLY_CHANNEL,
P 3                          IOSB = REPLY_IOSB,
P 3                          P1   = REPLY_BUFFER,
P 3                          P2   = ($BYTEOFFSET (OPC$S_MS_OTEXT) + $BYTEOFFSET (OPC$L_MS_TEXT))
  3                         ))
  2 THEN
  2     ABORT_MOUNT (MOUN$_MBXRDERR, 0, .STATUS);
  2
  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
```

```
1 ROUTINE INTERACTIVE_JOB =
1
1 !++
1 ! Functional Description:
1 !
1 !       This routine will determine if the current process is an
1 !       interactive process, and return that information to the
1 !       caller.  By definition, a process is interactive if it
1 !       has a terminal associated with it.
1 !
1 ! Input:
1 !
1 !       None.
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Routine Value:
1 !
1 !       1 if current process is an interactive process
1 !       0 if current process is not an interactive process
1 !--
1
2 BEGIN                                                 ! Start of INTERACTIVE_JOB
2
2 LOCAL
2       ITEM_LIST       : BBLOCK [16],                  ! Item list for $GETJPI
2       DEVICE_NAME     : BBLOCK [16],                  ! Device name buffer
2       NAME_LENGTH     : LONG;                         ! Cell for device name length
2
2 !
2 ! Build the $GETJPI item list and get the terminal name.
2 !
2 NAME_LENGTH = 0;                                      ! Zero the output cell
2 ITEM_LIST [0, 0, 16, 0] = 16;                         ! Set buffer length
2 ITEM_LIST [2, 0, 16, 0] = JPI$_TERMINAL;              ! Set item code
2 ITEM_LIST [4, 0, 32, 0] = DEVICE_NAME;                ! Set buffer address
2 ITEM_LIST [8, 0, 32, 0] = NAME_LENGTH;                ! Set result length address
2 ITEM_LIST [12,0, 32, 0] = 0;                          ! Set list terminator
2 $GETJPI (ITMLST = ITEM_LIST);
2
2 !
2 ! If a terminal is associated with the process, the terminal name
2 ! length should be nonzero.
2 !
2 IF .NAME_LENGTH NEQ 0
2 THEN
2     1                                                 ! Return TRUE
2 ELSE
2     0                                                 ! Return FALSE
2
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
```

```
1 ROUTINE SUBMIT_REQUEST (MSG_DESC,REPLY_EXPECTED) : NOVALUE =
1
1 !++
1 ! Functional Description:
1 !
1 !       This routine will send a request to all operators enabled
1 !       to receive disk and tape messages.  All requests that are
1 !       issued to the operator are echoed to the user.  Also, the
1 !       request context is saved so that when the operator replies
1 !       we can parse the reply in the context of the request.
1 !
1 ! Input:
1 !
1 !       MSG_DESC        = Address of a quadword string descriptor.
1 !                         The string is the operator request.
1 !
1 !       REPLY_EXPECTED  = Boolean value.  If true then an operator
1 !                         response is expected.
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Implicit Inputs:
1 !
1 !       MOUNT_STATUS    = status from current mount attempt
1 !
1 ! Implicit Outputs:
1 !
1 !       The request context is saved, the request is made.
1 !--
1
2 BEGIN                                           ! Start of SUBMIT_REQUEST
2
2 MAP
2
2       MSG_DESC        : REF BBLOCK;           ! Address of request descriptor
2
2 EXTERNAL
2
2       DEVICE_INDEX    : LONG VOLATILE;        ! Index into device list
2
2 LITERAL
2
2       BLANK           = %ASCII ' ',           ! Fill character
2       ZERO            = 0;                    ! Handy literal
2
2 LOCAL
2
2       STATUS          : LONG,                 ! Return status
2       MBX_CHAN        : LONG;                 ! Operator reply mailbox channel
2
2
2 !
2 ! If no mailbox exists, create one.
2 !
2 IF .REPLY_CHANNEL EQL ZERO
```

```
2 THEN
3     IF NOT (STATUS = $CREMBX (CHAN = REPLY_CHANNEL, PROMSK = MAILBOX_PROTECTION))
2     THEN
2         ABORT_MOUNT (MOUN$_MBXCRERR, 0, .STATUS);
2
2 !
2 ! Fill in the necessary fields in the request string.
2 ! Copy the message string to the operator message buffer.
2 !
2 REQUEST_ID = .REQUEST_ID + 1;                            ! Inc reqeust #
2 OP_MSG_BUF[OPC$L_MS_RQSTID]= .REQUEST_ID;                ! Set request #
2 !
2 CH$COPY (.MSG_DESC[DSC$W_LENGTH],                        ! Source length
2          .MSG_DESC[DSC$A_POINTER],                       ! Source pointer
2          BLANK,                                          ! Fill character
2          OPC$S_MS_OTEXT-$BYTEOFFSET(OPC$L_MS_TEXT),! Destination length
2          OP_MSG_BUF+$BYTEOFFSET(OPC$L_MS_TEXT)  ! Destination pointer
2         );
2 OP_MSG_DESC[DSC$W_LENGTH] = .MSG_DESC[DSC$W_LENGTH]+$BYTEOFFSET(OPC$L_MS_TEXT);
2
2 IF .REPLY_EXPECTED
2 THEN
3     BEGIN
3     !
3     ! An operator reply is expected.  Save the condition
3     ! context and set up the reply mailbox channel.
3     !
3     PREVIOUS_STATUS  = .MOUNT_STATUS;
3     PREVIOUS_DEV_IDX = .DEVICE_INDEX;
3     REPLY_PENDING    =  TRUE;
3     MBX_CHAN         = .REPLY_CHANNEL;
3     END
2 ELSE
2     !
2     ! An operator reply is not expected.
2     ! Indicate this to OPCOM by specifying a mailbox channel of zero.
2     !
2     MBX_CHAN = ZERO;
2
2 !
2 ! Set the operator target mask.
2 !
2 SET_TARGET_MASK ();
2 OP_MSG_BUF[TARGET_FIELD] = .OPERATOR_MASK;
2 !
2 ! Send the request to the operator.
2 !
3 IF NOT (STATUS = $SNDOPR (MSGBUF=OP_MSG_DESC, CHAN=.MBX_CHAN))
2 THEN
2     ABORT_MOUNT (MOUN$_OPRSNDERR, 0, .STATUS);
2 !
2 ! Echo the operator request to the user.  If no operator is
2 ! present, do not echo the request.  This interlock is necessary
2 ! to prevent repeatedly issuing the request if no OPCOM process
2 ! is present.
2 !
2 IF .OPERATOR_PRESENT
```

```
;   2 THEN
;   2     SIGNAL (MOUN$_OPRQST, 1, .MSG_DESC);
;   2 !
;   2 ! An alternate request status returned by $SNDOPR is SS$_NOPERATOR,
;   2 ! which indicates that there is no operator present to service the
;   2 ! request.  Taken in this context, it means that there is no OPCOM
;   2 ! process present on the system.
;   2 !
;   2 IF .STATUS EQL OPC$_NOPERATOR
;   2 THEN
;   3     BEGIN
;   3     REPLY_PENDING = FALSE;
;   3     IF NOT INTERACTIVE_JOB ()
;   3     THEN
;   3         !
;   3         ! Abort the mount, as no one can service the request.
;   3         !
;   3         ABORT_MOUNT (MOUN$_BATCHNOOPR)
;   3     ELSE
;   4         BEGIN
;   4         !
;   4         ! Inform the user that no operator is available to service
;   4         ! the request.  The user then has three courses of action:
;   4         !    - Abort the mount via CTRL-C
;   4         !    - Wait for an operator to enable himself to service the request
;   4         !    - Service the request himself.  (Hands-on environment)
;   4         !
;   4         ! Since the problem may go away in time, wait a short while after
;   4         ! informing the user before continuing the MOUNT operation.
;   4         !
;   4         IF .OPERATOR_PRESENT
;   4         THEN
;   4             SIGNAL (MOUN$_NOOPR);
;   4         OPERATOR_PRESENT = FALSE;
;   5         IF NOT (STATUS = $SETIMR (EFN=TIMER_FLAG, REQIDT=TIMER_ID, DAYTIM=DELTA_TIME))
;   4         THEN
;   4             ABORT_MOUNT (.STATUS);
;   4         $WAITFR (EFN = TIMER_FLAG);
;   4         $CANTIM (REQIDT = TIMER_ID);
;   4         $SETEF  (EFN = TIMER_FLAG);
;   3         END;
;   2     END;
;   2 !
;   2 ! If an operator reply is expected, then issue a read to the reply mailbox.
;   2 !
;   2 REPLY_IOSB = 0;
;   2 IF .REPLY_PENDING
;   2 THEN
;   2     POST_READ_TO_MBX ();
;   2
;   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
```

```
1 ROUTINE SET_TARGET_MASK : NOVALUE =
1
1 !++
1 ! Functional description:
1 !
1 !       Get the device characteristics and figure out which class
1 !       of operator is to receive the request.  If the device is a
1 !       tape, send the request to tape class operators.  If the
1 !       device is a disk, send the request to disk class operators.
1 !       If the device is neither tape or disk (ie. the user screwed
1 !       up the device name on the command line) then send the
1 !       request to both disk and tape class operators.  We remember
1 !       the operator class mask in case we later have to cancel
1 !       the request.
1 !
1 ! Input:
1 !
1 !       None.
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Implicit Input:
1 !
1 !       The MOUNT data base.  Note that:
1 !       DEVICE_STRING[.DEVICE_INDEX*2] = the address of string descriptor
1 !                                        of the device currently being mounted.
1 !
1 ! Implicit Output:
1 !
1 !       OPERATOR_MASK = mask of target operators.  Only
1 !                       the low 3 bytes are significant.
1 !
1 !--
1
2 BEGIN                                         ! Start of SET_TARGET_MASK
2
2 EXTERNAL
2       DEVICE_INDEX    : LONG VOLATILE,        ! Index into aforementioned vector
2       PHYS_NAME       : VECTOR VOLATILE;      ! Vector of device descriptors
2
2 LOCAL
2       DEVICE_CHAR     : BBLOCK [DIB$K_LENGTH],! Primary characteristics buffer
2       DEVICE_CHAR2    : BBLOCK [DIB$K_LENGTH],! Secondary characteristics buffer
2       DEVCHAR_DESC    : BBLOCK [DSC$K_S_BLN], ! Descriptor of primary char. buffer
2       DEVCHAR_DESC2   : BBLOCK [DSC$K_S_BLN], ! Descriptor of secondary char. buffer
2       STATUS          : LONG;
2
2 !
2 ! Set up the device characteristic buffer descriptors.
2 !
2 DEVCHAR_DESC [DSC$W_LENGTH]  = DIB$K_LENGTH;
2 DEVCHAR_DESC [DSC$B_DTYPE]   = DSC$K_DTYPE_T;
2 DEVCHAR_DESC [DSC$B_CLASS]   = DSC$K_CLASS_S;
2 DEVCHAR_DESC [DSC$A_POINTER] = DEVICE_CHAR;
2 DEVCHAR_DESC2 [DSC$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
```

```
ROUTINE CANCEL_REQUEST (REQUEST_STATUS) : NOVALUE =

!++
! Functional Description:
!
!       This routine will cancel an outstanding operator request.
!       The reply mailbox is deleted after the cancelation message
!       is sent so there will be no stale messages lying around to
!       confuse things later on.  The user is notified of the cancelation.
!
! Input:
!
!       REQUEST_STATUS  : A boolean value that describes the status of the
!                         operator request.  A value of 1 indicates the request
!                         has been successfully completed without  operator
!                         intervention, and the reason for the request no
!                         longer exists.  A value of 0 indicates that the
!                         request has not been satisfied, but is being canceled
!                         for some reason.
!
! Output:
!
!       None.
!
! Implicit Input:
!
!       REPLY_PENDING = TRUE if there is an outstanding operater request.
!
! Implicit Outputs:
!
!       REPLY_PENDING = FALSE
!--

BEGIN                                                   ! Start of CANCEL_REQUEST


IF .REPLY_PENDING
THEN
    BEGIN
    !
    ! Send cancelation notice to operator
    !
    BBLOCK [CANCEL_MSG_BUF [OPC$L_RQ_OPTIONS], OPC$V_RQSTDONE] = .REQUEST_STATUS;
    CANCEL_MSG_BUF[OPC$L_RQSTID] = .REQUEST_ID;
    CANCEL_MSG_BUF[OPC$L_ATTNMASK1] = .OPERATOR_MASK;
    $SNDOPR (MSGBUF=CANCEL_MSG_DESC, CHAN=.REPLY_CHANNEL);
    !
    ! Deassign the channel to the reply mailbox.  Since it
    ! is a temporary mailbox, it will be deleted.
    !
    $DASSGN (CHAN = .REPLY_CHANNEL);
    REPLY_CHANNEL = 0;
    REPLY_PENDING = FALSE;
    !
    ! Clear the reply event flag.
    !
    $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 3       THEN
; 1048          1943 3           SIGNAL (MOUN$_RQSTDON)
; 1049          1944 3       ELSE
; 1050          1945 3           SIGNAL (MOUN$_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   #7512179                                      ; 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
```

```
1 ROUTINE CHECK_FOR_REPLY : NOVALUE =
1
1 !++
1 ! Functional Description:
1 !
1 !       This routine will check to see if the operator
1 !       replied to a request after DELTA_TIME expired.
1 !       If so, the response must be parsed and acted upon.
1 !       Note that this might require undoing a successful mount.
1 !       If the request is still outstanding and the mount
1 !       completed successfully, then cancel the request.
1 !
1 ! Input:
1 !
1 !       WAIT_ENABLED = TRUE if we are to wait, FALSE if not.
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Implicit Inputs:
1 !
1 !       REPLY_PENDING = 1 if there is an outstanding request.
1 !       REPLY_DESC = string descriptor of the operator's reply.
1 !       REPLY_BUFFER = buffer holding the operator's reply.
1 !       MOUNT data base.
1 !
1 ! Implicit Outputs:
1 !
1 !       The MOUNT data base may be updated as a result of the operator's reply.
1 !--
1
2 BEGIN                                             ! Start of CHECK_FOR_REPLY
2
2 LOCAL
2
2       EF_STATE        : LONG,                     ! State of Event flags
2       STATUS          : LONG;
2
2 IF NOT .MOUNT_FAILED
2 THEN
2     !
2     ! The mount succeeded.  Operator intervention is
2     ! no longer necessary, so cancel the request.
2     !
2     CANCEL_REQUEST (REQUEST_SATISFIED)
2 ELSE
3     BEGIN
3     !
3     ! The mount failed (again).
3     !
3     ! If a reply was pending, wait for either the timer to go off or
3     ! for the reply to arrive, whichever comes first.  If no reply is
3     ! pending, then simply wait for the timer to go off.  Cancel the
3     ! timer on the way out, just to be thourough.
3     !
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
```

```
1 ROUTINE ALLOCFAIL_HNDLR : NOVALUE =
1
1 !++
1 ! Functional Description:
1 !
1 !       This routine will attempt to recover from a device
1 !       allocation failure.  This means that the device
1 !       specified by the user (or operator) cannot be
1 !       successfully allocated.  Notify the operator and
1 !       try again.  Current allocation failures handled are:
1 !
1 !               SS$_DEVALLOC    - device allocated to another user
1 !               SS$_NODEVAVL    - no devices of generic type are available
1 !               SS$_NOSUCHDEV   - incorrect device specifier
1 !
1 ! Input:
1 !
1 !       None.
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Implicit Input:
1 !
1 !       MOUNT_STATUS  = status of current mount attempt
1 !       REPLY_PENDING = TRUE if an operator request is outstanding
1 !       The MOUNT data base.
1 !
1 ! Implict Output:
1 !
1 !       The MOUNT data base may be changed as
1 !       the result of operator intervention.
1 !--
1
2 BEGIN                                           ! Start of ALLOCFAIL_HNDLR
2
2 EXTERNAL
2
2       COMMENT_STRING  : BBLOCK,               ! User comment string
2       DEVICE_INDEX    : LONG VOLATILE,        ! Index into device name vector
2       PHYS_NAME       : VECTOR VOLATILE;      ! Physical device name descriptor
2 LITERAL
2
2       FAO_CTRL_SIZ    = FAO_BUFFER_SIZE/2;    ! Maximum size for FAO control string
2
2 LOCAL
2
2       ALLOCFAIL_FAO   : BBLOCK [DSC$K_S_BLN], ! FAO control string descriptor
2       FAO_CTRL_BUF    : BBLOCK [FAO_CTRL_SIZ],! Buffer for FAO control string
2       STATUS          : LONG;
2
2
2 !
2 ! If this condition is different from the one signaled previously,
2 ! cancel any outstanding requests before handling this condition.
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
```

```
1 ROUTINE MEDOFL_HNDLR : NOVALUE =
1
1 !++
1 ! Functional Description:
1 !
1 !       This routine will attempt to recover from a medium
1 !       offline condition.  This usually means that the disk is
1 !       not spun up.  Notify the operator that the device
1 !       needs to be put online.
1 !
1 ! Input:
1 !
1 !       None.
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Implicit Input:
1 !
1 !       MOUNT_STATUS  = status of the current mount attempt
1 !       REPLY_PENDING = TRUE if an operator request is outstanding
1 !       The MOUNT data base.
1 !
1 ! Implict Output:
1 !
1 !       The MOUNT data base may be changed as
1 !       the result of operator intervention.
1 !--
1
2 BEGIN                                         ! Start of MEDOFL_HNDLR
2
2 EXTERNAL
2
2       COMMENT_STRING  : BBLOCK,               ! User comment string
2       LABEL_STRING    : VECTOR VOLATILE,      ! Vector of label descriptors
2       PHYS_NAME       : VECTOR VOLATILE,      ! Physical device name descriptor
2       DEVICE_INDEX    : LONG VOLATILE;        ! Index into DEVICE_STRING vector
2
2 LITERAL
2
2       FAO_CTRL_SIZ    = FAO_BUFFER_SIZE/2;    ! FAO control string size
2
2 LOCAL
2
2       MEDOFL_FAO      : BBLOCK [DSC$K_S_BLN],
2       MEDOFL_BUF      : BBLOCK [FAO_CTRL_SIZ],
2       VOLUME_FAO      : BBLOCK [DSC$K_S_BLN],
2       VOLUME_BUF      : BBLOCK [FAO_CTRL_SIZ],
2       VOLUME_DESC     : BBLOCK [DSC$K_S_BLN],
2       VOLUME_BUFFER   : BBLOCK [FAO_CTRL_SIZ],
2       STATUS          : LONG;
2
2
2
2 !
2 ! If this condition is different from the one signaled previously,
  2 ! cancel any outstanding requests before handling this condition.
  2 ! Note that if the previous condition was SS$_INCVOLLABEL, we do
  2 ! not cancel the request and issue another one.  This is to give
  2 ! the operator a chance to remove the incorrect volume from the drive
  2 ! and to (hopefully) insert the correct volume.
  2
  3 IF  ((.MOUNT_STATUS AND STS$M_COND_ID) NEQ (SS$_INCVOLLABEL AND STS$M_COND_ID))
  3 AND ((.PREVIOUS_STATUS AND STS$M_COND_ID) EQL (SS$_INCVOLLABEL AND STS$M_COND_ID))
  3 AND (.DEVICE_INDEX  EQL .PREVIOUS_DEV_IDX)
  2 THEN
  3     BEGIN
  3     PREVIOUS_STATUS = .MOUNT_STATUS;
  3     END;
  2
  3 IF (.DEVICE_INDEX NEQ .PREVIOUS_DEV_IDX)
  3 OR (.MOUNT_STATUS NEQ .PREVIOUS_STATUS)
  2 THEN
  3     BEGIN
  3     CANCEL_REQUEST (REQUEST_NOT_SATISFIED);
  3     OPERATOR_PRESENT = TRUE;                        ! Assume operator present
  3     END;
  2 !
  2 ! If there is no outstanding request, then submit a request.
  2 !
  2 IF NOT .REPLY_PENDING
  2 THEN
  3     BEGIN
  3     !
  3     ! Set up the output descriptor and format the volume label string.
  3     !
  3     VOLUME_DESC [DSC$W_LENGTH]  = FAO_CTRL_SIZ;
  3     VOLUME_DESC [DSC$B_DTYPE]   = DSC$K_DTYPE_T;
  3     VOLUME_DESC [DSC$B_CLASS]   = DSC$K_CLASS_S;
  3     VOLUME_DESC [DSC$A_POINTER] = VOLUME_BUFFER;
  3     IF .LABEL_STRING[.DEVICE_INDEX*2] GTR 0
  3     THEN
  4         BEGIN
  4         !
  4         ! Set up the output descriptor and get the FAO control string.
  4         !
  4         VOLUME_FAO [DSC$W_LENGTH] = FAO_CTRL_SIZ;
  4         VOLUME_FAO [DSC$B_DTYPE]  = DSC$K_DTYPE_T;
  4         VOLUME_FAO [DSC$B_CLASS]  = DSC$K_CLASS_S;
  4         VOLUME_FAO [DSC$A_POINTER] = VOLUME_BUF;
P 5         IF NOT (STATUS = $GETMSG (MSGID  = MOUN$_VOLNAME,
P 5                                   MSGLEN = VOLUME_FAO [DSC$W_LENGTH],
P 5                                   BUFADR = VOLUME_FAO,
P 5                                   FLAGS  = MSG_TEXT
  5                                  ))
  4         THEN
  4             ABORT_MOUNT (.STATUS, 0, .MOUNT_STATUS);
  4         !
  4         ! Format the volume label string,
  4         !
P 4         $FAO    (VOLUME_FAO,
P 4                  VOLUME_DESC [DSC$W_LENGTH],
P 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$:    CLRL    -(SP)                                   ; 2203
        FE35     CF                 01  FB 0005A        CALLS   #1, CANCEL_REQUEST                      ;
          F4     A2                 01  D0 0005F        MOVL    #1, OPERATOR_PRESENT                    ; 2204
                 01          EC     A2  E9 00063 3$:    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                                  ;
     00000000G   00                 03  FB 000AE        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
```

```
1 ROUTINE WRONGVOL_HNDLR : NOVALUE =
1
1 !++
1 ! Functional Description:
1 !
1 !       This routine will attempt to recover from an SS$_INCVOLLABEL
1 !       condition, which implies that the label of the volume presently
1 !       in the drive does not match the volume label specified by the user.
1 !
1 ! Input:
1 !
1 !       None.
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Implicit Input:
1 !
1 !       MOUNT_STATUS  = status of the current mount attempt
1 !       REPLY_PENDING = TRUE if an operator request is outstanding
1 !       The MOUNT data base.
1 !
1 ! Implict Output:
1 !
1 !       The MOUNT data base may be changed as
1 !       the result of operator intervention.
1 !--
1
2 BEGIN                                                 ! Start of WRONGVOL_HNDLR
2
2 EXTERNAL
2
2       PHYS_NAME       : VECTOR VOLATILE,      ! Physical device name descriptor
2       DEVICE_INDEX    : LONG VOLATILE,        ! Index into DEVICE_STRING vector
2       LABEL_STRING    : VECTOR VOLATILE;      ! Vector of volume labels
2
2 LITERAL
2
2       FAO_CTRL_SIZ    = FAO_BUFFER_SIZE/2;    ! FAO control string size
2
2 LOCAL
2
2       WRONGVOL_FAO    : BBLOCK [DSC$K_S_BLN],
2       WRONGVOL_BUF    : BBLOCK [FAO_CTRL_SIZ],
2       STATUS          : LONG;
2
2
2 !
2 ! If this condition is different from the one signaled previously,
2 ! cancel any outstanding requests before handling this condition.
2 ! Otherwise do nothing.
2 !
3 IF (.MOUNT_STATUS NEQ .PREVIOUS_STATUS)
3 OR (.DEVICE_INDEX NEQ .PREVIOUS_DEV_IDX)
2 THEN
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        BLBS     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
```

```
1 ROUTINE PRINT_REPLY : NOVALUE =
1
1 !++
1 ! Funtional description:
1 !
1 !       This routine is a local utility routine used by PARSE_REPLY
1 !       to output the operator reply the user (SYS$OUTPUT).
1 !
1 ! Input:
1 !
1 !       None.
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Implicit input:
1 !
1 !       None.
1 !
1 ! Implicit output:
1 !
1 !       The operator reply, if any, is written to SYS$OUTPUT.
1 !
1 ! Side effects:
1 !
1 !       None.
1 !
1 ! Routine value:
1 !
1 !       None.
1 !
1 !--
1
2 BEGIN                                         ! Start of PRINT_REPLY
2
2 LOCAL
2       TEXT_DESC       : BBLOCK [DSC$K_S_BLN]; ! String descriptor
2
2 !
2 ! If the operator reply is greater than 8 bytes, then
2 ! it had some text to it.  If this is the case, inform
2 ! the user of the operator reply.  Note that the 8 bytes
2 ! of message overhead are not printed.  A temporary string
2 ! descriptor must be used so that $FAO will not replace
2 ! the any nonprinting ASCII characters with blanks.
2
2 IF .REPLY_IOSB[2,0,16,0] GTR $BYTEOFFSET (OPC$L_MS_TEXT)
2 THEN
3     BEGIN
3     TEXT_DESC [DSC$W_LENGTH]    = .REPLY_IOSB [2,0,16,0] - $BYTEOFFSET (OPC$L_MS_TEXT);
3     TEXT_DESC [DSC$B_DTYPE]     = DSC$K_DTYPE_T;
3     TEXT_DESC [DSC$B_CLASS]     = DSC$K_CLASS_S;
3     TEXT_DESC [DSC$A_POINTER]   = .REPLY_DESC [DSC$A_POINTER] + $BYTEOFFSET (OPC$L_MS_TEXT);
3     SIGNAL (MOUN$_OPREPLY, 1, TEXT_DESC);
2     END;
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
```

```
1 ROUTINE PARSE_REPLY : NOVALUE =
1
1 !++
1 ! Functional Description:
1 !
1 !       This routine will parse the operator reply in the context
1 !       of the conditon that spawned it, and then do the appropriate
1 !       thing, based on the operator's reply.
1 !
1 ! Input:
1 !
1 !       None.
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Implicit Inputs:
1 !
1 !       REPLY_DESC = string descriptor of the operator's reply.
1 !       REPLY_BUFFER = buffer holding the operator's reply.
1 !       MOUNT data base.
1 !
1 ! Implicit Outputs:
1 !
1 !       The MOUNT data base may be updated as a result of the operator's reply.
1 !--
1
2 BEGIN                                           ! Start of PARSE_REPLY
2
2 EXTERNAL ROUTINE
2
2       LIB$TPARSE      : ADDRESSING_MODE (GENERAL);    ! Used to parse operator reply
2
2 PSECT GLOBAL = $GLOBAL$;
2 GLOBAL
2       NEWLINE         : DESCRIP (%CHAR (13,10));      ! Descriptor for newline string
2
2 BIND
2       !
2       ! Create the character translation table that will be used by the
2       ! CH$TRANSLATE function.  he table is st up so that all lower-cag
2       ! alphabetic characters are translated to their upper-case equivalent.
2       !
2       TRANS_TABLE     = CH$TRANSTABLE
2                         (0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,
2                          20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,
2                          37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,
2                          54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,
2                          71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,
2                          88,89,90,91,92,93,94,95,96,65,66,67,68,69,70,71,72,
2                          73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,
2                          90,123,124,125,126,127
2                         );
2
2 LOCAL
2
```

```
2           PTR             : LONG,                           ! Character pointer
2           STATUS          : LONG;
2
2   !
2   ! Check the status of the mailbox read.  If
2   ! not successful, then abort the mount.
2   !
2   IF NOT .REPLY_IOSB[0,0,16,0]
2   THEN
3       BEGIN
3       REPLY_PENDING = FALSE;
3       ABORT_MOUNT (MOUN$_MBXRDERR, 0, .REPLY_IOSB[0,0,16,0]);
2       END;
2
2   !
2   ! Decide what to do based on the type of operator reply.
2   ! The OPC$_xxxxx status codes are longer than a word, so
2   ! they are masked off to word size befor comparing them
2   ! to the reply status.
2   !
2   SELECTONEU .REPLY_BUFFER[OPC$W_MS_STATUS] OF
2       SET
3       [(OPC$_NOPERATOR AND %X'0FFFF')]    : BEGIN
3                                             !
3                                             ! No operator was enabled to receive the request.
3                                             !
3                                             REPLY_PENDING = FALSE;
3                                             IF NOT INTERACTIVE_JOB ()
3                                             THEN
3                                                 !
3                                                 ! Abort the mount, as no one is can service the request.
3                                                 !
3                                                 ABORT_MOUNT (MOUN$_BATCHNOOPR)
3                                             ELSE
4                                                 BEGIN
4                                                 !
4                                                 ! If this is the first time through this code for this conditi
4                                                 ! for this device, then inform the user that no operator is en
4                                                 ! to receive the request.
4                                                 !
4                                                 IF .OPERATOR_PRESENT
4                                                 THEN
4                                                     SIGNAL (MOUN$_NOOPR);
4                                                 OPERATOR_PRESENT = FALSE;
4                                                 !
4                                                 ! Re-issue the request, in the hope that an operator will even
4                                                 ! be enabled to receive and service the request.
4                                                 !
5                                                 IF NOT (STATUS = $SNDOPR (MSGBUF=OP_MSG_DESC, CHAN=.REPLY_CHAN
4                                                 THEN
4                                                     ABORT_MOUNT (MOUN$_OPRSNDERR, 0, .STATUS);
4                                                 !
4                                                 ! If the request was sent, re-issue a read to the reply mailbo
4                                                 !
4                                                 IF .STATUS NEQ OPC$_NOPERATOR
4                                                 THEN
5                                                     BEGIN
```

```
                POST_READ_TO_MBX ();
                REPLY_PENDING = TRUE;
                END;
            END;
        END;

    [(OPC$_RQSTCMPLTE AND %X'0FFFF')]   : BEGIN
                                          !
                                          ! The operator replied to our request.
                                          !
                                          PRINT_REPLY ();
                                          PREVIOUS_STATUS = -1;
                                          REPLY_PENDING = FALSE;
                                          OPERATOR_PRESENT = TRUE;
                                          !
                                          ! If there is no operator reply text, then return.
                                          !
                                          IF (.REPLY_IOSB [2,0,16,0] EQL $BYTEOFFSET (OPC$L_MS_TEXT))
                                          THEN
                                              RETURN;
                                          !
                                          ! Create a string descriptor for the operator reply text.
                                          !
                                          TPARSE_BLOCK [TPA$L_STRINGCNT] = .REPLY_IOSB [2,0,16,0] - $BYTEOFF
                                          TPARSE_BLOCK [TPA$L_STRINGPTR] = .REPLY_DESC [DSC$A_POINTER]+$BYTE
                                          !
                                          ! The reply text may contain a NEWLINE character.  If so, the inte
                                          ! is BEFORE  the NEWLINE character.  Note that the NEWLINE charact
                                          ! two characters, a carriage-return followed by a line-feed (<cr><
                                          !
                                          PTR    = CH$FIND_SUB    (.TPARSE_BLOCK [TPA$L_STRINGCNT],
                                                                   .TPARSE_BLOCK [TPA$L_STRINGPTR],
                                                                   .NEWLINE [DSC$W_LENGTH],
                                                                   .NEWLINE [DSC$A_POINTER]
                                                                  );
                                          !
                                          ! If a NEWLINE was found, set the string descriptor
                                          ! so that the text BEFORE the NEWLINE is parsed.
                                          !
                                          IF NOT CH$FAIL (.PTR)
                                          THEN
                                              TPARSE_BLOCK [TPA$L_STRINGCNT] = .PTR - .TPARSE_BLOCK [TPA$L_S
                                          !
                                          ! If there is no text before the NEWLINE, then there is no operato
                                          !
                                          IF .TPARSE_BLOCK [TPA$L_STRINGCNT] EQL 0
                                          THEN
                                              RETURN;
                                          !
                                          ! Convert the reply to upper case, so TPARSE will work correctly.
                                          !
                                          CH$TRANSLATE  (TRANS_TABLE,
                                                          .TPARSE_BLOCK [TPA$L_STRINGCNT],
                                                          .TPARSE_BLOCK [TPA$L_STRINGPTR],
                                                          0,
                                                          .TPARSE_BLOCK [TPA$L_STRINGCNT],
                                                          .TPARSE_BLOCK [TPA$L_STRINGPTR]
```

```
3                                                   );
3                                               !
3                                               ! Parse the operator response and perform whatever action is neces
3                                               !
4                                               IF NOT (STATUS = LIB$TPARSE (TPARSE_BLOCK, STATE_TABLE, KEY_TABLE)
3                                               THEN
3                                                   ABORT_MOUNT (.STATUS, 0, .MOUNT_STATUS);
2                                               END;
2
3       [(OPC$_RQSTPEND AND %X'0FFFF')]         : BEGIN
3                                               !
3                                               ! The operator did a REPLY/PENDING.  The orginal
3                                               ! request is still active, so issue another read
3                                               ! to the reply mailbox.
3                                               !
3                                               PRINT_REPLY ();
3                                               OPERATOR_PRESENT = TRUE;
3                                               POST_READ_TO_MBX ();
2                                               END;
2
3       [(OPC$_RQSTABORT AND %X'0FFFF')]        : BEGIN
3                                               !
3                                               ! The operator has aborted the mount request.
3                                               !
3                                               PRINT_REPLY ();
3                                               REPLY_PENDING = FALSE;
3                                               OPERATOR_PRESENT = TRUE;
3                                               ABORT_MOUNT (MOUN$_OPRABORT);
2                                               END;
2
2       [(OPC$_RQSTCAN  AND %X'0FFFF'),
3        (OPC$_RQSTDONE AND %X'0FFFF')]         : BEGIN
3                                               !
3                                               ! The user has canceled the requst, and
3                                               ! the operator is acknowledging it.
3                                               !
3                                               PREVIOUS_STATUS = -1;
3                                               REPLY_PENDING = FALSE;
3                                               OPERATOR_PRESENT = TRUE;
2                                               END;
2
2       [(OPC$_BLANKTAPE AND %X'0FFFF'),
3        (OPC$_INITAPE   AND %X'0FFFF')]        : BEGIN
3                                               !
3                                               ! These messages may be sent by mistake.  Notify
3                                               ! the interested parties, and let MOUNT try again.
3                                               !
3                                               PREVIOUS_STATUS = -1;
3                                               REPLY_PENDING = FALSE;
3                                               OPERATOR_PRESENT = TRUE;
3                                               INVALID_COMMAND ();
2                                               END;
2
3       [OTHERWISE]                             : BEGIN
3                                               !
3                                               ! This is an unknown response type.
3                                               ! Abort the mount and print the bad message.
```

```
; 1775          2664  3                                  !
; 1776          2665  3                                  REPLY_PENDING = FALSE;
; 1777          2666  3                                  OPERATOR_PRESENT = TRUE;
; 1778        P 2667  3                                  ABORT_MOUNT   (MOUN$_BADREPLY,                        ! Error code
; 1779        P 2668  3                                                 5,                                     ! FAO count
; 1780        P 2669  3                                                 .REPLY_BUFFER[OPC$B_MS_TYPE],          ! Message type
; 1781        P 2670  3                                                 .REPLY_BUFFER[OPC$W_MS_STATUS],        ! Message status
; 1782        P 2671  3                                                 .REPLY_BUFFER[OPC$L_MS_RPLYID],        ! Message Ident
; 1783        P 2672  3                                                 .REPLY_DESC[DSC$W_LENGTH]  - $BYTEOFFSET (OPC$L_MS_
; 1784        P 2673  3                                                 .REPLY_DESC[DSC$A_POINTER] + $BYTEOFFSET (OPC$L_MS_
; 1785          2674  3                                                );
; 1786          2675  2                                  END;
; 1787          2676  2       TES;
; 1788          2677  2
; 1789          2678  1 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 0008B           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       00E0    C7  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
```

```
1 ROUTINE SAVE_DEVICE =
1
1 !++
1 ! Functional description:
1 !
1 !       This is a TPARSE action routine that is called
1 !       to create a string descriptor for the token
1 !       just parsed.  The token is a device name.
1 !
1 ! Input:
1 !
1 !       None.
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Implicit Inputs:
1 !
1 !       TPARSE_BLOCK = data structure defining TPARSE context.
1 !
1 ! Implicit outputs:
1 !
1 !       DEVICE_DESC = string descriptor of device name.
1 !
1 ! Routine Value:
1 !
1 !       1 If the device name length is within tolerance,
1 !       0 if not.
1 !
1 !--
1
2 BEGIN                                           ! Start of SAVE_DEVICE
2
2
2 EXTERNAL
2
2       DEVICE_DESC     : BBLOCK,               ! Device string descriptor
2       TPARSE_BLOCK    : BBLOCK;               ! TPARSE context data structure
2
2 IF .TPARSE_BLOCK[TPA$L_TOKENCNT] GTR MAX_DEV_LENGTH           ! Check for device name too long
2 THEN
2     0                                                         ! Return failure
2 ELSE
3     BEGIN
3     DEVICE_DESC[DSC$W_LENGTH]  = .TPARSE_BLOCK[TPA$L_TOKENCNT];
3     DEVICE_DESC[DSC$A_POINTER] = .TPARSE_BLOCK[TPA$L_TOKENPTR];
3     1                                                         ! Return success
3     END
3
1 END;                                          ! End of SAVE_DEVICE


                                  0000 00000 SAVE_DEVICE:
```

```
                                                                       J 14
ASSIST                                                          16-Sep-1984 01:04:04    VAX-11 Bliss-32 V4.0-742                     Page  56
V04-001                                                         14-Sep-1984 12:45:15    DISK$VMSMASTER:[MOUNT.SRC]ASSIST.B32;2       (15)

                                                                        .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
```

```
1 GLOBAL ROUTINE $COPY_INFO (DEV_INDEX, DEV_DESC) =
1
1 !++
1 ! Functional description:
1 !
1 !       This routine provides a secure way of copying a device name
1 !       string from the caller (in user mode) to MOUNT's protected
1 !       data base (in EXEC mode).
1 !
1 ! Input:
1 !
1 !       DEV_INDEX       : A number from 0 to .DEVICE_COUNT
1 !       DEV_DESC        : Address of a device name descriptor
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Implicit input:
1 !
1 !       DEVICE_STRING   : A vector of device name descriptors
1 !       DEVICE_COUNT    : The number of devices specified by the user.
1 !
1 ! Implicit output:
1 !
1 !       The DEVICE_STRING vector may be modified.
1 !
1 ! Routine value:
1 !
1 !       SS$_NORMAL      : Normal successful completion
1 !       SS$_ACCVIO      : The specified device name cannot be read.
1 !       SS$_BADPARAM    : The specified device name has a zero length,
1 !                         or is longer than LOG$C_NAMLENGTH bytes, or
1 !                         DEV_INDEX is not a reasonable value.
1 !--
1
2 BEGIN                                                 ! Start of $COPY_INFO
2
2 EXTERNAL
2       DEVICE_COUNT    : LONG,                         ! # of drives
2       DEVICE_STRING   : VECTOR VOLATILE;              ! Descriptor list
2
2 BUILTIN
2       PROBER;                                         ! Probe for read access
2
2 LOCAL
2       DEV_NAME        : BBLOCK [DSC$K_S_BLN]; ! Local descriptor
2
2 !
2 ! Make sure DEV_INDEX is within a reasonable range.
2 !
3 IF (.DEV_INDEX LSS 0) OR (.DEV_INDEX GTR (.DEVICE_COUNT - 1))
2 THEN
2     RETURN (SS$_BADPARAM);
2
2 !
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], R0             ; 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
```

```
1 GLOBAL ROUTINE $CHANGE_PROT =
1
1 !++
1 ! Functional description:
1 !
1 !       This routine will change the page protection of this module's
1 !       OWN storage so that it may be written to in USER mode.
1 !
1 ! Input:
1 !
1 !       None.
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Implicit input:
1 !
1 !       1) The current access mode is EXEC or KERNEL.
1 !       2) VA_RANGE is a vector of two longword elements, containing the starting
1 !          and ending virtual addresses of the range of pages to work on.
1 !
1 ! Implicit output:
1 !
1 !       The pages are made USER readable.
1 !
1 ! Routine value:
1 !
1 !       Whatever status value is returned by $SETPRT.
1 !--
1
2 BEGIN                                                   ! Start of $CHANGE_PROT
2
2 EXTERNAL
2       DEVICE_INDEX,                                   ! Index into PHYS_NAME bblock
2       DATA_BASE_READY,                                ! Boolean
2       STORED_CONTEXT;                                 ! Bit vector
2
2
2 !
2 ! Initialize three important variables referenced in VMOUNT.  This
2 ! must be done here as they are zeroed only once per $MOUNT call,
2 ! and must be written while in EXEC mode.
2 !
2 DEVICE_INDEX = 0;
2 DATA_BASE_READY = 0;
2 STORED_CONTEXT = 0;
2
2 !
2 ! Set the page protection of this module's data to allow user
2 ! mode read/write access.  This must be done here, in EXEC mode, since
2 ! this image is INSTALLed as a protected shareable image, which has
2 ! the effect of setting the protection to be USER read, EXEC write.
2 ! Note that the data sits in a special PSECT, to avoid changing
2 ! the page protection on adjacent pages.
2 !
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
```

```
1 GLOBAL ROUTINE $DALLOC_DEVS (SINGLE_DEVICE) =
1
1 !++
1 ! Functional description:
1 !
1 !       This routine will attempt to dealocate all devices that were
1 !       specified by the user that were not previously allocated.
1 !
1 ! Input:
1 !
1 !       SINGLE_DEVICE   : a longword boolean to  control whether all
1 !                         drives or just a single one is to be deallocated.
1 !                         If the latter, use DEVICE_INDEX to select the
1 !                         drive name from the PHYS_NAME vector.
1 !
1 ! Output:
1 !
1 !       None.
1 !
1 ! Implicit input:
1 !
1 !       CLEANUP_ALLOC   : a bit vector where each bit represents an
1 !                         an entry in PHYS_NAME that was not previously
1 !                         allocated by the user.
1 !       DEVICE_INDEX    : index into PHYS_NAME vector
1 !       PHYS_NAME       : a vector of device name descriptors for all
1 !                         devices specified by the user.
1 !       PHYS_COUNT      : a high-water mark that indicates the number
1 !                         of devices actually used in the mount.
1 !
1 ! Implicit output:
1 !
1 !       All devices not mounted or not previously allocated are deallocated.
1 !
1 ! Routine value:
1 !
1 !       SS$_NORMAL      : Normal successful completion
1 !--
1
2 BEGIN                                           ! Start of $DALLOC_DEVS
2
2 EXTERNAL
2       CLEANUP_ALLOC   : BITVECTOR VOLATILE,   ! cleanup bit vector
2       DEV_ALLOCATED   : BITVECTOR VOLATILE,   ! device allready allocated
2       DEV_ACQUIRED    : BITVECTOR VOLATILE,   ! device is interlocked
2       DEVICE_INDEX    : LONG,                 ! current device
2       PHYS_COUNT      : LONG,                 ! count of physical devices
2       PHYS_NAME       : VECTOR VOLATILE,      ! device descriptor array
2       MOUNT_OPTIONS   : BITVECTOR,            ! mount options and modifiers
2       STORED_CONTEXT  : BITVECTOR;            ! special mount context
2
2
2 IF .SINGLE_DEVICE
2 THEN
2     !
2     ! Deallocate a specific device.  This is used to deallocate a
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                          ;
                          0000GCF40 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       0000G     CF  D0 00043 4$:      MOVL    PHYS_COUNT, R3                        ; 3046
                       52                 01  CE 00048          MNEGL   #1, I                                 ;
                                          22  11 0004B          BRB     8$                                    ;
          1E           64                 52  E1 0004D 5$:      BBC     I, CLEANUP_ALLOC, 8$                  ; 3047
                                          7E  D4 00051          CLRL    -(SP)                                 ; 3050
          50           52                 01  78 00053          ASHL    #1, I, R0                             ;
                                0000GCF40     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
              0000000G  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$:   PUSHL   SS_FAIL_MODE                              ; 3100
              0000000G  00              01  FB 0001B       CALLS   #1, SYS$SETSFM                            ;
```

```
                                   04 00022        RET                                                        ; 3102

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

```
 !
 ! The TPARSE tables are here because they mangle
 ! PSECT definitions.
 !

 !
 ! Define the TPARSE grammar of the possible operator replies.
 !
 $INIT_STATE     (STATE_TABLE,KEY_TABLE);

 !
 ! Initial state
 !
 $STATE  (START,
         ((SUBSTITUTE_COMMAND),  TPA$_EXIT,      DO_SUBSTITUTE),
         (TPA$_LAMBDA,           TPA$_EXIT,      INVALID_COMMAND)
         );

 !
 ! SUBSTITUTE command.  'SUBSTITUTE'<TPA$_BLANK><DEVICE><TEXT>
 !
 $STATE  (SUBSTITUTE_COMMAND,
         ('SUBSTITUTE')
         );

 !$STATE (,
 !       (TPA$_BLANK)
 !       );

 $STATE  (,
         ((DEVICE),              TPA$_EXIT,      SAVE_DEVICE)
         );

 !$STATE (,
 !       ((TEXT),                TPA$_EXIT)
 !       );

 !
 ! Device name.  It may be a device spec or a logical name string.
 !
 $STATE  (DEVICE,
         (TPA$_SYMBOL)
         );

 $STATE  (,
         (':',                   TPA$_EXIT),
         (TPA$_LAMBDA,           TPA$_EXIT)
         );
 !
 ! Text.  The remainder of the operator response is treated
 ! as a comment, and has no effect on the mount.  If there is
 ! a comment, at least one blank must separate it from the
 ! previous section of the operator response.
 !
 $STATE  (TEXT,
         (TPA$_BLANK,            MORE_TEXT),
         (TPA$_EOS,              TPA$_EXIT)
```

```
          1            );
          1
  P       1 $STATE    (MORE_TEXT,
  P       1            (TPA$_ANY,              MORE_TEXT),
  P       1            (TPA$_EOS,              TPA$_EXIT)
          1            );
          1 END
          0 ELUDOM


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

                                                    00000 ;TPA$KEYST0
                                                          U.10:    .BLKB   0
           45  54  55  54  49  54  53  42  55  53  00000 ;TPA$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
```

0243 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY