```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 OOOOOOOOO        AAAAAAAAA
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 OOOOOOOOO        AAAAAAAAA
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 OOOOOOOOO        AAAAAAAAA
SSS              YYY         YYY  SSS              LLL              OOO         OOO  AAA         AAA
SSS              YYY         YYY  SSS              LLL              OOO         OOO  AAA         AAA
SSS              YYY         YYY  SSS              LLL              OOO         OOO  AAA         AAA
SSS                 YYY   YYY     SSS              LLL              OOO         OOO  AAA         AAA
SSS                 YYY   YYY     SSS              LLL              OOO         OOO  AAA         AAA
SSS                 YYY   YYY     SSS              LLL              OOO         OOO  AAA         AAA
   SSSSSSSSS           YYY           SSSSSSSSS     LLL              OOO         OOO  AAA         AAA
   SSSSSSSSS           YYY           SSSSSSSSS     LLL              OOO         OOO  AAA         AAA
   SSSSSSSSS           YYY           SSSSSSSSS     LLL              OOO         OOO  AAA         AAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAAAAAAAAAAAAAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAAAAAAAAAAAAAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAAAAAAAAAAAAAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAA         AAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAA         AAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     OOOOOOOOO     AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     OOOOOOOOO     AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     OOOOOOOOO     AAA         AAA
```

```
_$2

Sym
---
SS$
SS$

SS$
SS$
SS$
SS$
SS$
SS$
SYS
SYS
SYS
TRY
UNL
WRI
```

```
I 4
**FILE**ID**CSPMOUNT
```

```
  CCCCCCCC    SSSSSSSS  PPPPPPPP    MM      MM    000000    UU      UU  NN      NN  TTTTTTTTTT
  CCCCCCCC    SSSSSSSS  PPPPPPPP    MM      MM    000000    UU      UU  NN      NN  TTTTTTTTTT
CC          SS          PP      PP  MMMM  MMMM  00      00  UU      UU  NN      NN      TT
CC          SS          PP      PP  MMMM  MMMM  00      00  UU      UU  NN      NN      TT
CC          SS          PP      PP  MM  MM  MM  00      00  UU      UU  NNNN    NN      TT
CC          SS          PP      PP  MM  MM  MM  00      00  UU      UU  NNNN    NN      TT
CC            SSSSSS    PPPPPPPP    MM      MM  00      00  UU      UU  NN  NN  NN      TT
CC            SSSSSS    PPPPPPPP    MM      MM  00      00  UU      UU  NN  NN  NN      TT
CC                  SS  PP          MM      MM  00      00  UU      UU  NN    NNNN      TT
CC                  SS  PP          MM      MM  00      00  UU      UU  NN    NNNN      TT
CC                  SS  PP          MM      MM  00      00  UU      UU  NN      NN      TT           ....
CC                  SS  PP          MM      MM  00      00  UU      UU  NN      NN      TT           ....
  CCCCCCCC  SSSSSSSS    PP          MM      MM    000000    UUUUUUUUUU  NN      NN      TT           ....
  CCCCCCCC  SSSSSSSS    PP          MM      MM    000000    UUUUUUUUUU  NN      NN      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
```

```
0001 0 MODULE  CSPMOUNT
0002 0                  (LANGUAGE (BLISS32)
0003 0                  ,IDENT = 'V04-000'
0004 0                  ) =
0005 0
0006 0 !****************************************************************************
0007 0 !*                                                                          *
0008 0 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0009 0 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0010 0 !*  ALL RIGHTS RESERVED.                                                    *
0011 0 !*                                                                          *
0012 0 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0013 0 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0014 0 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0015 0 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0016 0 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0017 0 !*  TRANSFERRED.                                                            *
0018 0 !*                                                                          *
0019 0 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0020 0 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0021 0 !*  CORPORATION.                                                            *
0022 0 !*                                                                          *
0023 0 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0024 0 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0025 0 !*                                                                          *
0026 0 !*                                                                          *
0027 0 !****************************************************************************
0028 0
0029 0 !++
0030 0 !
0031 0 ! FACILITY:     MOUNT,CSP
0032 0 !
0033 0 ! ABSTRACT:
0034 0 !
0035 0 !       This module contains the cluster server action routine for
0036 0 !       MOUNT and is part of the Cluster Server Process (CSP).
0037 0 !
0038 0 ! Environment:
0039 0 !
0040 0 !       Full process context capable of kernel mode.
0041 0 !
0042 0 ! Author:
0043 0 !
0044 0 !       Hai Huang
0045 0 !
0046 0 ! Creation date:
0047 0 !
0048 0 !       28 Feb 1984
0049 0 !
0050 0 !
0051 0 ! Revision history:
0052 0 !
0053 0 !       V03-003 HH0022          Hai Huang       17-May-1984
0054 0 !               Dismiss the mount request if the device is not
0055 0 !               cluster-wide, or if the device is already mounted.
0056 0 !
0057 0 !       V03-002 HH0007          Hai Huang       16-Mar-1984
```

```
0 !                     Add cluster-wide group-volume support.
0 !
0 !             V03-001 HH0004          Hai Huang       01-Mar-1984
0 !                     Add cluster-wide mount support.
0 !
0 !--
0
1 BEGIN                                                 ! Start of CSPMOUNT
1
1 LIBRARY 'SYS$LIBRARY:LIB.L32' ;
1 REQUIRE 'LIB$:CSPDEF' ;
1
1 LINKAGE
1               JSB_2   = JSB (REGISTER=2) ;
1
1 FORWARD ROUTINE
1
1               CSP$MOUNT               : JSB_2,
1               CSP_MOUNT_DECIPHER      : NOVALUE,
1               CSP_DISMOUNT_DECIPHER   : NOVALUE,
1               GET_UIC,
1               SET_UIC,
1               CHECK_DEVICE;
1
1
```

```
1
1 %SBTTL 'CSP$MOUNT      - MOUNT client support for CSP'
1 GLOBAL ROUTINE CSP$MOUNT
1                       (CSD : REF BLOCK [,BYTE]) : JSB_2 =
1
1 !+
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine performs the CSP mount client action routine.
1 !       The possible actions are mount and dismount, depending on
1 !       the parameter specified in the CSD packet.
1 !
1 ! INPUTS:
1 !
1 !       CSD     : Pointer to the address of the received CSD
1 !
1 ! OUTPUTS:
1 !
1 !       None.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       None.
1 !
1 ! OUTPUT PARAMETERS:
1 !
1 !       None.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       Mount or dismount system service issued.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       1         : If successful
1 !       Otherwise : Error status from mount/dismount system service
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None.
1 !
1 !-
1
1
2 BEGIN                                         ! Start of CSP$MOUNT
2
2 LOCAL
2       UIC,
2       STATUS,
2       BUFFER  : REF BLOCK;
2
2
2 BUFFER = .CSD [CSD$L_SENDOFF];                ! Get address of message
2
3 IF ((UIC = .CSD [CSD$L_P1]) NEQ 0)            ! A non-zero P1 is a mount request
2 THEN
```

```
;  141         0333 2
;  142         0334 3      BEGIN
;  143         0335 3
;  144         0336 3      LOCAL
;  145         0337 3          ARG     : VECTOR [2],
;  146         0338 3          OLD_UIC;
;  147         0339 3
;  148         0340 3      CSP_MOUNT_DECIPHER (.BUFFER);               ! Decipher cluster-mount packet
;  149         0341 3                                                  ! into a mount item list
;  150         0342 3      STATUS = CHECK_DEVICE (.BUFFER);            ! See if the mount should be processed
;  151         0343 3      IF NOT .STATUS                              ! If not, dismiss request
;  152         0344 3      THEN
;  153         0345 3          RETURN SS$_NORMAL;
;  154         0346 3      OLD_UIC = $CMKRNL (ROUTIN = GET_UIC);       ! Get original UIC
;  155         0347 3      ARG [0] = 1;                                ! Set up arglst
;  156         0348 3      ARG [1] = .UIC;                             ! Set new UIC
;  157         0349 3      $CMKRNL (ROUTIN = SET_UIC, ARGLST = ARG);   !   ...
;  158         0350 3      STATUS = $MOUNT (ITMLST = .BUFFER);         ! Mount
;  159         0351 3      ARG [1] = .OLD_UIC;                         ! Restore original UIC
;  160         0352 3      $CMKRNL (ROUTIN = SET_UIC, ARGLST = ARG);   !   ...
;  161         0353 3
;  162         0354 3      END
;  163         0355 3
;  164         0356 2 ELSE                                              ! P1=0 is a dismount request
;  165         0357 2
;  166         0358 3      BEGIN
;  167         0359 3
;  168         0360 3      LOCAL
;  169         0361 3          DEV_DSC,
;  170         0362 3          DISM_FLAGS;
;  171         0363 3
;  172         0364 3      CSP_DISMOUNT_DECIPHER ( .BUFFER, DEV_DSC, DISM_FLAGS );  ! Decipher the cluster-
;  173         0365 3                                                  ! dismount packet
;  174         0366 3      STATUS = $DISMOU ( DEVNAM=.DEV_DSC, FLAGS=.DISM_FLAGS ); ! Dismount
;  175         0367 3
;  176         0368 2      END;
;  177         0369 2
;  178         0370 2
;  179         0371 2 RETURN .STATUS;
;  180         0372 1 END ;



                                                        .TITLE  CSPMOUNT
                                                        .IDENT  \V04-000\

                                                        .EXTRN  SYS$CMKRNL, SYS$MOUNT
                                                        .EXTRN  SYS$DISMOU

                                                        .PSECT  $CODE$,NOWRT,2

                                   3C  BB 00000 CSP$MOUNT::
                                                        PUSHR   #^M<R2,R3,R4,R5>                          ; 0278
                     5E            10  C2 00002         SUBL2   #16, SP                                   ;
                     53       16   A2  D0 00005         MOVL    22(CSD), BUFFER                           ; 0329
                     52       52   A2  D0 00009         MOVL    82(CSD), UIC                              ; 0331
                                   5F  13 0000D         BEQL    2$                                        ;
                                   53  DD 0000F         PUSHL   BUFFER                                    ; 0340
```

```
                                                          N  4
CSPMOUNT                                                16-Sep-1984 01:14:34    VAX-11 Bliss-32 V4.0-742                 Page   5
V04-000        CSP$MOUNT - MOUNT client support for CSP  14-Sep-1984 13:18:02    [SYSLOA.SRC]CSPMOUNT.B32;1                      (2)

                         0000V  CF              01  FB 00011         CALLS   #1, CSP_MOUNT_DECIPHER                       ;
                                                53  DD 00016         PUSHL   BUFFER                                       ; 0342
                         0000V  CF              01  FB 00018         CALLS   #1, CHECK_DEVICE                             ;
                                55              50  D0 0001D         MOVL    R0, STATUS                                   ;
                                05              55  E8 00020         BLBS    STATUS, 1$                                   ; 0343
                                50              01  D0 00023         MOVL    #1, R0                                       ; 0345
                                                64  11 00026         BRB     4$                                           ;
                                                7E  D4 00028 1$:     CLRL    -(SP)                                        ; 0346
                                       0000V    CF  9F 0002A         PUSHAB  GET_UIC                                      ;
                      00000000G 00              02  FB 0002E         CALLS   #2, SYS$CMKRNL                               ;
                                54              50  D0 00035         MOVL    R0, OLD_UIC                                  ;
                            08  AE              01  D0 00038         MOVL    #1, ARG                                      ; 0347
                            0C  AE              52  D0 0003C         MOVL    UIC, ARG+4                                   ; 0348
                                          08    AE  9F 00040         PUSHAB  ARG                                          ; 0349
                                       0000V    CF  9F 00043         PUSHAB  SET_UIC                                      ;
                      00000000G 00              02  FB 00047         CALLS   #2, SYS$CMKRNL                               ;
                                                53  DD 0004E         PUSHL   BUFFER                                       ; 0350
                      00000000G 00              01  FB 00050         CALLS   #1, SYS$MOUNT                                ;
                                55              50  D0 00057         MOVL    R0, STATUS                                   ;
                            0C  AE              54  D0 0005A         MOVL    OLD_UIC, ARG+4                               ; 0351
                                          08    AE  9F 0005E         PUSHAB  ARG                                          ; 0352
                                       0000V    CF  9F 00061         PUSHAB  SET_UIC                                      ;
                      00000000G 00              02  FB 00065         CALLS   #2, SYS$CMKRNL                               ;
                                                1B  11 0006C         BRB     3$                                           ; 0331
                                                5E  DD 0006E 2$:     PUSHL   SP                                           ; 0364
                                          08    AE  9F 00070         PUSHAB  DEV_DSC                                      ;
                                                53  DD 00073         PUSHL   BUFFER                                       ;
                         0000V  CF              03  FB 00075         CALLS   #3, CSP_DISMOUNT_DECIPHER                    ;
                                                6E  DD 0007A         PUSHL   DISM_FLAGS                                   ; 0366
                                          08    AE  DD 0007C         PUSHL   DEV_DSC                                      ;
                      00000000G 00              02  FB 0007F         CALLS   #2, SYS$DISMOU                               ;
                                55              50  D0 00086         MOVL    R0, STATUS                                   ;
                                50              55  D0 00089 3$:     MOVL    STATUS, R0                                   ; 0371
                                5E              10  C0 0008C 4$:     ADDL2   #16, SP                                      ; 0372
                                                3C  BA 0008F         POPR    #^M<R2,R3,R4,R5>                             ;
                                                    05 00091         RSB                                                  ;

; Routine Size:  146 bytes,    Routine Base:  $CODE$ + 0000


;  181          0373  1
```

```
CSPM
V04-



; Ro

:  6
:  6
:  6



:
:
:
:
:  $
:  $


:
:
:
:
:
:
:  -



:

:

:  Si
:  Ru
:  El
:  Li
:  Le
:  Me
:  Co
```

```
1
1 %SBTTL 'CSP_MOUNT_DECIPHER      -Deciphers a packet into MOUNT itemlist'
1 ROUTINE CSP_MOUNT_DECIPHER ( BUFFER ) : NOVALUE =
1
1 !+
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine takes a cluster-mount packet and returns
1 !       an item list.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       CSP_MOUNT_DECIPHER (ARG1)
1 !
1 ! INPUTS:
1 !
1 !       ARG1    : Address of the input buffer
1 !
1 ! OUTPUTS:
1 !
1 !       None.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       None.
1 !
1 ! OUTPUT PARAMETERS:
1 !
1 !       None.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       None.
1 !
1 ! ROUTINE VALUES:
1 !
1 !       None.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       The cluster-mount packet in the buffer is transformed into
1 !       a mount item list.
1 !
1 !
1 ! NOTES:
1 !
1 ! This decipher routine takes the given cluster-mount packet of the form
1 ! shown below and transforms the packet into an item list.
1 !
1 !                                    Offset
1 !       +---------------+
1 !       | code1 | len1  |  0  ITEM_LENG item_desc_1
1 !       +---------------+
1 !       |offset to str_1|  4  ITEM_ADDR
1 !       +---------------+
1 !       |    unused     |  8  ITEM_NULL
```

```
1 !         +---------------+
1 !         | code2 | len2  |  0  ITEM_LENG item_desc_2
1 !         +---------------+
1 !         |offset to str_2|  4  ITEM_ADDR
1 !         +---------------+
1 !         |    unused     |  8  ITEM_NULL
1 !         +---------------+
1 !         .               .
1 !         .               .
1 !         .               .
1 !         +---------------+
1 !         |       0       |     End of item decsiptors
1 !         +---------------+
1 !         |     str_1     |
1 !         +---------------+
1 !         |     .....     |
1 !         +---------------+
1 !         |     str_2     |
1 !         +---------------+
1 !         |     .....     |
1 !         +---------------+
1 !
1 !
1 !         1. Each address in the item descriptor is "relocated" to be the
1 !            offset from the beginning of the packet (i.e. self-relative).
1 !            The transformation is simply to calculate the address in each
1 !            item descriptor.
1 !
1 !-
1
1
2 BEGIN                                                ! Start of CSP_MOUNT_DECIPHER
2
2 MAP
2         BUFFER          : REF BLOCK [,BYTE];
2
2 LOCAL
2         ITEM            : REF BLOCK [,BYTE];         ! Pointer to item descriptor
2
2
2 MACRO ITEM_LENG  =  0,0,16,0%;                       ! Define buffer offsets
2 MACRO ITEM_CODE  =  2,0,16,0%;
2 MACRO ITEM_ADDR  =  4,0,32,0%;
2 MACRO ITEM_NULL  =  8,0,32,0%;
2 LITERAL ITEM_SIZE = 12;
2
2 !
2 ! For each item descriptor, calculate the real address of the item.
2 !
2
2 ITEM = .BUFFER;                                      ! Point to the beginning of buffer
2 WHILE ( .ITEM [ITEM_CODE] NEQ 0 ) DO
3 BEGIN
3     ITEM [ITEM_ADDR] = .ITEM [ITEM_ADDR] + .BUFFER; ! Calculate the real address
3                                                      ! of the item string
3     ITEM = .ITEM + ITEM_SIZE;                        ! Bump item desciptor pointer
2 END;
```

```
;  297          0488  2
;  298          0489  2 RETURN;
;  299          0490  2
;  300          0491  1 END;                                          ! End of CSP_MOUNT_DECIPHER



                                               0000 00000 CSP_MOUNT_DECIPHER:
                                                                    .WORD   Save nothing                              ; 0376
                              50         04   AC  D0 00002          MOVL    BUFFER, ITEM                              ; 0481
                                         02   A0  B5 00006 1$:      TSTW    2(ITEM)                                   ; 0482
                                              0A  13 00009          BEQL    2$                                        ;
                         04   A0         04   AC  C0 0000B          ADDL2   BUFFER, 4(ITEM)                           ; 0484
                              50              0C  C0 00010          ADDL2   #12, ITEM                                 ; 0486
                                              F1  11 00013          BRB     1$                                        ; 0482
                                                  04 00015 2$:      RET                                               ; 0491

; Routine Size:  22 bytes,    Routine Base:  $CODE$ + 0092


;  301          0492  1
;  302          0493  1
```

```
1
1 %SBTTL 'CSP_DISMOUNT_DECIPHER   -Deciphers a packet into DISMOU arguments'
1 ROUTINE CSP_DISMOUNT_DECIPHER ( BUFFER, DEV_DSC, FLAGS ) : NOVALUE =
1
1 !+
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine takes a cluster-dismount packet and returns
1 !       a device descriptor and the dismount flags.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       CSP_DISMOUNT_DECIPHER (ARG1, ARG2, ARG3)
1 !
1 ! INPUTS:
1 !
1 !       ARG1    : Address of the input buffer
1 !
1 ! OUTPUTS:
1 !
1 !       None.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       None.
1 !
1 ! OUTPUT PARAMETERS:
1 !
1 !       ARG2    : Address of a longword to recieve the address
1 !                   of the device descriptor
1 !       ARG3    : Address of a longword to receive the flags
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       None.
1 !
1 ! ROUTINE VALUES:
1 !
1 !       None.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None.
1 !
1 !
1 ! NOTES:
1 !
1 ! This decipher routine takes the given cluster-dismount packet of the form
1 ! shown below and returns a device descriptor and the dismount flags.
1 !
1 !                                       Offset
1 !       +----------------+
1 !       :     flags      :  0  BUF_FLAGS
1 !       +----------------+
1 !       : dev descrptor  :  4  BUF_DSC
1 !       +----------------+
;  361       0551  1 !          :               :  8
;  362       0552  1 !          +---------------+
;  363       0553  1 !          : device string :  12  BUF_STR
;  364       0554  1 !          +---------------+
;  365       0555  1 !          :      ...      :
;  366       0556  1 !          +---------------+
;  367       0557  1 !
;  368       0558  1 !-
;  369       0559  1
;  370       0560  2 BEGIN                                                ! Start of CSP_DISMOUNT_DECIPHER
;  371       0561  2
;  372       0562  2 MAP
;  373       0563  2         BUFFER  : REF BLOCK [,BYTE] ;
;  374       0564  2
;  375       0565  2 LOCAL
;  376       0566  2         LOC_DSC : REF BLOCK [,BYTE] ;
;  377       0567  2
;  378       0568  2
;  379       0569  2 MACRO BUF_FLAG   =  0,0,32,0%;                        ! Define buffer offsets
;  380       0570  2 MACRO BUF_DSC    =  4,0,32,0%;
;  381       0571  2 MACRO BUF_STR    = 12,0,32,0%;
;  382       0572  2 LITERAL BUF_HDR_LEN = 12;
;  383       0573  2
;  384       0574  2 .FLAGS = .BUFFER[BUF_FLAG];                          ! Get flags from buffer
;  385       0575  2 LOC_DSC = BUFFER[BUF_DSC];                           ! Point to device descriptor
;  386       0576  2 LOC_DSC[DSC$A_POINTER] = .LOC_DSC[DSC$A_POINTER] + .BUFFER; ! 'Relocate' address
;  387       0577  2                                                      !   in device descriptor
;  388       0578  2 .DEV_DSC = .LOC_DSC;                                 ! Return address of device dsc
;  389       0579  2
;  390       0580  2 RETURN;
;  391       0581  1 END;                                                 ! End of CSP_DISMOUNT_DECIPHER



                                        0000 00000 CSP_DISMOUNT_DECIPHER:
                                                           .WORD   Save nothing                          ; 0496
                     0C   BC      04   BC  D0 00002        MOVL    @BUFFER, @FLAGS                       ; 0574
              50     04   AC           04  C1 00007        ADDL3   #4, BUFFER, LOC_DSC                   ; 0575
                     04   A0      04   AC  C0 0000C        ADDL2   BUFFER, 4(LOC_DSC)                    ; 0576
                     08   BC           50  D0 00011        MOVL    LOC_DSC, @DEV_DSC                     ; 0578
                                           04 00015        RET                                           ; 0581

; Routine Size:  22 bytes,    Routine Base:  $CODE$ + 00A8


;  392       0582  1
```

```
:  394         0583  1
:  395         0584  1 %SBTTL 'GET_UIC        - Get our process UIC'
:  396         0585  1 ROUTINE GET_UIC =
:  397         0586  1
:  398         0587  1 !++
:  399         0588  1 !
:  400         0589  1 ! FUNCTIONAL DESCRIPTION:
:  401         0590  1 !
:  402         0591  1 !       This is a kernel-mode routine to get the UIC of a process.
:  403         0592  1 !
:  404         0593  1 ! CALLING SEQUENCE:
:  405         0594  1 !
:  406         0595  1 !       GET_UIC ()
:  407         0596  1 !
:  408         0597  1 ! INPUT PARAMETERS:
:  409         0598  1 !
:  410         0599  1 !       None.
:  411         0600  1 !
:  412         0601  1 ! IMPLICIT INPUTS:
:  413         0602  1 !
:  414         0603  1 !       None.
:  415         0604  1 !
:  416         0605  1 ! OUTPUT PARAMETERS:
:  417         0606  1 !
:  418         0607  1 !       None.
:  419         0608  1 !
:  420         0609  1 ! IMPLICIT OUTPUTS:
:  421         0610  1 !
:  422         0611  1 !       None.
:  423         0612  1 !
:  424         0613  1 ! ROUTINE VALUE:
:  425         0614  1 !
:  426         0615  1 !       UIC of this process.
:  427         0616  1 !
:  428         0617  1 ! SIDE ErFECTS:
:  429         0618  1 !
:  430         0619  1 !       None.
:  431         0620  1 !
:  432         0621  1 !--
:  433         0622  1
:  434         0623  2 BEGIN
:  435         0624  2
:  436         0625  2 EXTERNAL
:  437         0626  2         SCH$GL_CURPCB   : REF BLOCK [, BYTE] ADDRESSING_MODE (ABSOLUTE);
:  438         0627  2                                                 ! system address of process PCB
:  439         0628  2
:  440         0629  2 RETURN (.SCH$GL_CURPCB[PCB$L_UIC]);
:  441         0630  2
:  442         0631  1 END;                                            ! End of routine GET_UIC


                                                                .EXTRN  SCH$GL_CURPCB

                                            0000 00000 GET_UIC:.WORD    Save nothing                                ; 0585
                              50 00000000G   9F  D0 00002       MOVL    @#SCH$GL_CURPCB, R0                         ; 0629
                              50      00BC   C0  D0 00009       MOVL    188(R0), R0                                 ;
```

```
                                                            04 0000E         RET                                             ; 0631

; Routine Size:  15 bytes,    Routine Base:  $CODE$ + 00BE

;  443          0632  1
;   445           0633  1
;   446           0634  1 %SBTTL 'SET_UIC       - Set our process UIC'
;   447           0635  1 ROUTINE SET_UIC ( UIC ) =
;   448           0636  1
;   449           0637  1 !++
;   450           0638  1 !
;   451           0639  1 ! FUNCTIONAL DESCRIPTION:
;   452           0640  1 !
;   453           0641  1 !       This is a kernel-mode routine to set the UIC of a process.
;   454           0642  1 !
;   455           0643  1 ! CALLING SEQUENCE:
;   456           0644  1 !
;   457           0645  1 !       SET_UIC (ARG1)
;   458           0646  1 !
;   459           0647  1 ! INPUT PARAMETERS:
;   460           0648  1 !
;   461           0649  1 !       ARG1    : Desired UIC
;   462           0650  1 !
;   463           0651  1 ! IMPLICIT INPUTS:
;   464           0652  1 !
;   465           0653  1 !       None.
;   466           0654  1 !
;   467           0655  1 ! OUTPUT PARAMETERS:
;   468           0656  1 !
;   469           0657  1 !       None.
;   470           0658  1 !
;   471           0659  1 ! IMPLICIT OUTPUTS:
;   472           0660  1 !
;   473           0661  1 !       None.
;   474           0662  1 !
;   475           0663  1 ! ROUTINE VALUE:
;   476           0664  1 !
;   477           0665  1 !       1.
;   478           0666  1 !
;   479           0667  1 ! SIDE EFFECTS:
;   480           0668  1 !
;   481           0669  1 !       None.
;   482           0670  1 !
;   483           0671  1 !--
;   484           0672  1
;   485           0673  2 BEGIN
;   486           0674  2
;   487           0675  2 EXTERNAL
;   488           0676  2       SCH$GL_CURPCB   : REF BLOCK [, BYTE] ADDRESSING_MODE (ABSOLUTE);
;   489           0677  2                                               ! System address of process PCB
;   490           0678  2 SCH$GL_CURPCB [PCB$L_UIC] = .UIC;               ! Set UIC
;   491           0679  2
;   492           0680  2 RETURN 1;
;   493           0681  2
;   494           0682  1 END;                                            ! End of routine SET_UIC



                                    0000 00000 SET_UIC:.WORD   Save nothing                              ; 0635
                  50 00000000G  9F  D0 00002          MOVL    @#SCH$GL_CURPCB, R0                       ; 0678
```

```
                        00BC   C0       04    AC  D0 00009         MOVL    UIC, 188(R0)                          ;
                               50             01  D0 0000F         MOVL    #1, R0                                ; 0680
                                                  04 00012         RET                                           ; 0682

; Routine Size:  19 bytes,    Routine Base:  $CODE$ + 00CD


;  495          0683  1
;  496          0684  1
```

```
1
1 %SBTTL 'CHECK_DEVICE     - Check if the mount request should be processed'
1 ROUTINE CHECK_DEVICE ( BUFFER ) =
1
1 !+
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine determines if the mount request received should
1 !       be processed. If the target device is already mounted, or is
1 !       not a cluster-wide device, then the request should be dismissed.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       CHECK_DEVICE (ARG1)
1 !
1 ! INPUTS:
1 !
1 !       ARG1    : Address of the mount item list
1 !
1 ! OUTPUTS:
1 !
1 !       None.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       None.
1 !
1 ! OUTPUT PARAMETERS:
1 !
1 !       None.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       None.
1 !
1 ! ROUTINE VALUES:
1 !
1 !       0       : If the mount request should be dismissed.
1 !       1       : If the mount request should be processed.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       None.
1 !
1 !-
1
1
2 BEGIN                                         ! Start of CHECK_DEVICE
2
2 MAP
2       BUFFER          : REF BLOCK [,BYTE];
2
2 LOCAL
2       STATUS,
2       LOCAL_EFN,                              ! Local event flag
2       ITEM            : REF BLOCK [,BYTE],    ! Pointer to item descriptor
```

```
  2           DEV_DESC          : BLOCK [DSC$K_S_BLN, BYTE], ! Target device descriptor
  2           DEVCHAR           : BLOCK [4, BYTE],       ! Device char word buffer
  2           DEVCHAR2          : BLOCK [4, BYTE],       ! 2nd device char word buffer
  2           ITMLST            : BLOCK [(2*12)+4, BYTE] INITIAL
  2                             !
  2                             ! 1st item - device charasteristic word
  2                             !
  2                             ( WORD (4),              ! Buffer length
  2                               WORD (DVI$_DEVCHAR),   ! 1st device char word
  2                               LONG (DEVCHAR),        ! Address of buffer
  2                               LONG (0),              ! No length
  2                             !
  2                             ! 2nd item - 2nd device characteristic word
  2                             !
  2                               WORD (4),              ! Buffer length
  2                               WORD (DVI$_DEVCHAR2),  ! 2nd device char word
  2                               LONG (DEVCHAR2),       ! Address of buffer
  2                               LONG (0),              ! No length
  2                               LONG (0));             ! Item list stopper
  2
  2 EXTERNAL ROUTINE
  2           LIB$GET_EF        : ADDRESSING_MODE (GENERAL), ! RTL routine to get an EF
  2           LIB$FREE_EF       : ADDRESSING_MODE (GENERAL); ! RTL routine to release the EF
  2
  2 MACRO ITEM_LENG  =  0,0,16,0%;                      ! Define buffer offsets
  2 MACRO ITEM_CODE  =  2,0,16,0%;
  2 MACRO ITEM_ADDR  =  4,0,32,0%;
  2 MACRO ITEM_NULL  =  8,0,32,0%;
  2 LITERAL ITEM_SIZE = 12;
  2
  2 STATUS = 0;                                         ! Assume failure
  2 ITEM = .BUFFER;                                     ! Point to the beginning of buffer
  2 LIB$GET_EF (LOCAL_EFN);                             ! Get a local event flag
  2
  2 !
  2 ! Scan the item list for device names. For each device name in item list,
  2 ! issue a $GETDVI system service to find out the status of the device.
  2 !
  2 WHILE ( .ITEM [ITEM_CODE] NEQ 0 ) DO                ! Examine each item
  3     BEGIN
  3     IF .ITEM [ITEM_CODE] EQL MNT$_DEVNAM
  3     THEN
  4         BEGIN                                       ! For device names only
  4         DEV_DESC [DSC$B_DTYPE] = 0;                 ! Set up device descritor
  4         DEV_DESC [DSC$B_CLASS] = 0;
  4         DEV_DESC [DSC$W_LENGTH]  = .ITEM [ITEM_LENG];
  4         DEV_DESC [DSC$A_POINTER] = .ITEM [ITEM_ADDR];
  4
P 4         STATUS = $GETDVIW ( DEVNAM = DEV_DESC,  ! Get device info
P 4                             ITMLST = ITMLST,
  4                             EFN    = .LOCAL_EFN );
  4
  5         IF ( NOT .STATUS )                          ! If $GETDVI failed
  5         OR ( .DEVCHAR [DEV$V_MNT] )                 ! or device already mounted
  5         OR ( NOT .DEVCHAR2 [DEV$V_CLU] )            ! or not cluster-wide device
  4         THEN
  5             BEGIN
```

```
;   612      0799  5              STATUS = 0;                         ! Return with failure
;   613      0800  5              EXITLOOP;
;   614      0801  4              END;
;   615      0802  3          END;                                    !
;   616      0803  3      ITEM = .ITEM + ITEM_SIZE;                   ! Bump item desciptor pointer
;   617      0804  2      END;                                        ! End of while loop
;   618      0805  2
;   619      0806  2  LIB$FREE_EF (LOCAL_EFN);                        ! Release the event flag
;   620      0807  2
;   621      0808  2  RETURN .STATUS;                                  ! Back to caller
;   622      0809  2
;   623      0810  1  END;                                            ! End of CHECK_DEVICE


                                                             .PSECT  $PLIT$,NOWRT,NOEXE,2

                                     0004  00000 P.AAA:     .WORD   4                                        ;
                                     0002  00002            .WORD   2                                        ;
                                 00000000  00004            .LONG   0                                        ;
                                 00000000  00008            .LONG   0                                        ;
                                     0004  0000C            .WORD   4                                        ;
                                     00E6  0000E            .WORD   230                                      ;
                                 00000000  00010            .LONG   0                                        ;
                                 00000000  00014            .LONG   0                                        ;
                                 00000000  00018            .LONG   0                                        ;

                                                            .EXTRN  LIB$GET_EF, LIB$FREE_EF
                                                            .EXTRN  SYS$GETDVIW

                                                            .PSECT  $CODE$,NOWRT,2

                                      003C 00000 CHECK_DEVICE:
                                                            .WORD   Save R2,R3,R4,R5                         ; 0687
                         5E            30  C2 00002         SUBL2   #48, SP                                  ;
   0C    AE     0000'    CF            1C  28 00005         MOVC3   #28, P.AAA, ITMLST                       ; 0760
                 10      AE            6E  9E 0000C         MOVAB   DEVCHAR, ITMLST+4                        ; 0733
                 1C      AE      04    AE  9E 00010         MOVAB   DEVCHAR2, ITMLST+16                      ;
                                       53  D4 00015         CLRL    STATUS                                   ; 0772
                         52      04    AC  D0 00017         MOVL    BUFFER, ITEM                             ; 0773
                                 08    AE  9F 0001B         PUSHAB  LOCAL_EFN                                ; 0774
         00000000G       00            01  FB 0001E         CALLS   #1, LIB$GET_EF                           ;
                                 02    A2  B5 00025 1$:     TSTW    2(ITEM)                                  ; 0780
                                       3D  13 00028         BEQL    4$                                       ;
                         01      02    A2  B1 0002A         CMPW    2(ITEM), #1                              ; 0782
                                       32  12 0002E         BNEQ    3$                                       ;
                 28      AE            62  3C 00030         MOVZWL  (ITEM), DEV_DESC                         ; 0787
                 2C      AE      04    A2  D0 00034         MOVL    4(ITEM), DEV_DESC+4                      ; 0788
                                       7E  7C 00039         CLRQ    -(SP)                                    ; 0792
                                       7E  7C 0003B         CLRQ    -(SP)                                    ;
                                 1C    AE  9F 0003D         PUSHAB  ITMLST                                   ;
                                 3C    AE  9F 00040         PUSHAB  DEV_DESC                                 ;
                                       7E  D4 00043         CLRL    -(SP)                                    ;
                                 24    AE  DD 00045         PUSHL   LOCAL_EFN                                ;
         00000000G       00            08  FB 00048         CALLS   #8, SYS$GETDVIW                          ;
                         53            50  D0 0004F         MOVL    R0, STATUS                               ;
                         09            53  E9 00052         BLBC    STATUS, 2$                               ; 0794
```

```
                     04        02   AE             03  E0 00055         BBS      #3, DEVCHAR+2, 2$                     ; 0795
                                    04        04   AE  E8 0005A         BLBS     DEVCHAR2, 3$                          ; 0796
                                                   53  D4 0005E  2$:    CLRL     STATUS                                ; 0799
                                                   05  11 00060         BRB      4$                                    ; 0798
                                    52             0C  C0 00062  3$:    ADDL2    #12, ITEM                             ; 0803
                                                   BE  11 00065         BRB      1$                                    ; 0780
                                              08   AE  9F 00067  4$:    PUSHAB   LOCAL_EFN                             ; 0806
                           00000000G  00           01  FB 0006A         CALLS    #1, LIB$FREE_EF                       ;
                                    50             53  D0 00071         MOVL     STATUS, R0                            ; 0808
                                                       04 00074         RET                                            ; 0810

; Routine Size:  117 bytes,    Routine Base:  $CODE$ + 00E0


;  624          0811  1
;  625          0812  1 END                                              ! End of CSPMOUNT
;  626          0813  0 ELUDOM




;                                       PSECT SUMMARY
;
;       Name                          Bytes                         Attributes
;
;  $CODE$                               341  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                                28  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)




;                               Library Statistics
;
;                                           -------- Symbols --------      Pages       Processing
;       File                                Total   Loaded   Percent       Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1            18619       18         0       1000          00:01.4





;                                        COMMAND QUALIFIERS

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

; Size:          341 code + 28 data bytes
; Run Time:         00:08.6
; Elapsed Time:     00:39.7
; Lines/CPU Min:     5645
; Lexemes/CPU-Min: 29986
; Memory Used:  109 pages
; Compilation Complete
```

0394 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CSPOPCOM
LIS

CSPWAIT
LIS

CSPRCPCAC
LIS

CSPCJFRES
LIS

CSPQUORUM
LIS

DISTRLKI
LIS

CSPMOUNT
LIS

CSPVECTOR
LIS

CSPCLIENT
LIS

DSTRDLCK
LIS

DSTRLOCK
LIS