G 3

```
EEEEEEEEEEEEEEE  XXX         XXX     CCCCCCCCCCC   HHH         HHH  NNN         NNN     GGGGGGGGGGG
EEEEEEEEEEEEEEE  XXX         XXX     CCCCCCCCCCC   HHH         HHH  NNN         NNN     GGGGGGGGGGG
EEEEEEEEEEEEEEE  XXX         XXX     CCCCCCCCCCC   HHH         HHH  NNN         NNN     GGGGGGGGGGG
EEE              XXX         XXX  CCC              HHH         HHH  NNN         NNN  GGG
EEE              XXX         XXX  CCC              HHH         HHH  NNN         NNN  GGG
EEE              XXX         XXX  CCC              HHH         HHH  NNN         NNN  GGG
EEE                 XXX   XXX     CCC              HHH         HHH  NNNNNN      NNN  GGG
EEE                 XXX   XXX     CCC              HHH         HHH  NNNNNN      NNN  GGG
EEE                 XXX   XXX     CCC              HHH         HHH  NNNNNN      NNN  GGG
EEEEEEEEEEEE           XXX        CCC              HHHHHHHHHHHHHHH  NNN   NNN   NNN  GGG
EEEEEEEEEEEE           XXX        CCC              HHHHHHHHHHHHHHH  NNN   NNN   NNN  GGG
EEEEEEEEEEEE           XXX        CCC              HHHHHHHHHHHHHHH  NNN   NNN   NNN  GGG
EEE                 XXX   XXX     CCC              HHH         HHH  NNN      NNNNNN  GGG   GGGGGGGGG
EEE                 XXX   XXX     CCC              HHH         HHH  NNN      NNNNNN  GGG   GGGGGGGGG
EEE                 XXX   XXX     CCC              HHH         HHH  NNN      NNNNNN  GGG   GGGGGGGGG
EEE              XXX         XXX  CCC              HHH         HHH  NNN         NNN  GGG         GGG
EEE              XXX         XXX  CCC              HHH         HHH  NNN         NNN  GGG         GGG
EEE              XXX         XXX  CCC              HHH         HHH  NNN         NNN  GGG         GGG
EEEEEEEEEEEEEEE  XXX         XXX     CCCCCCCCCCC   HHH         HHH  NNN         NNN     GGGGGGGGG
EEEEEEEEEEEEEEE  XXX         XXX     CCCCCCCCCCC   HHH         HHH  NNN         NNN     GGGGGGGGG
EEEEEEEEEEEEEEE  XXX         XXX     CCCCCCCCCCC   HHH         HHH  NNN         NNN     GGGGGGGGG
```

```
**FILE**ID**EXCUTIL                                  L 11
```

```
EEEEEEEEEE  XX      XX    CCCCCCCC  UU      UU  TTTTTTTTTT    IIIIII    LL
EEEEEEEEEE  XX      XX    CCCCCCCC  UU      UU  TTTTTTTTTT    IIIIII    LL
EE          XX      XX  CC          UU      UU      TT          II      LL
EE          XX      XX  CC          UU      UU      TT          II      LL
EE            XX  XX    CC          UU      UU      TT          II      LL
EE            XX  XX    CC          UU      UU      TT          II      LL
EEEEEEEE        XX      CC          UU      UU      TT          II      LL
EEEEEEEE        XX      CC          UU      UU      TT          II      LL
EE            XX  XX    CC          UU      UU      TT          II      LL
EE            XX  XX    CC          UU      UU      TT          II      LL
EE          XX      XX  CC          UU      UU      TT          II      LL              ....
EE          XX      XX  CC          UU      UU      TT          II      LL              ....
EEEEEEEEEE  XX      XX    CCCCCCCC  UUUUUUUUUU      TT        IIIIII    LLLLLLLLLL      ....
EEEEEEEEEE  XX      XX    CCCCCCCC  UUUUUUUUUU      TT        IIIIII    LLLLLLLLLL      ....

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
```

EX
VO

```
0001 0 MODULE  exch$util                                  %TITLE 'Facility-wide misc routines'
0002 0                  (
0003 0                  IDENT = 'V04-000',
0004 0                  ADDRESSING_MODE (EXTERNAL=LONG_RELATIVE, NONEXTERNAL=WORD_RELATIVE)
0005 0                  ) =
0006 1 BEGIN
0007 1 !
0008 1 !****************************************************************************
0009 1 !*                                                                          *
0010 1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0011 1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0012 1 !*  ALL RIGHTS RESERVED.                                                    *
0013 1 !*                                                                          *
0014 1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0015 1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0016 1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0017 1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0018 1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0019 1 !*  TRANSFERRED.                                                            *
0020 1 !*                                                                          *
0021 1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0022 1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0023 1 !*  CORPORATION.                                                            *
0024 1 !*                                                                          *
0025 1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0026 1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0027 1 !*                                                                          *
0028 1 !*                                                                          *
0029 1 !****************************************************************************
0030 1
0031 1 !++
0032 1 ! FACILITY:      EXCHANGE - Foreign volume interchange facility
0033 1 !
0034 1 ! ABSTRACT:      Miscellaneous utility routines
0035 1 !
0036 1 ! ENVIRONMENT:   VAX/VMS User mode
0037 1 !
0038 1 ! AUTHOR:        CW Hobbs                CREATION DATE: 8-July-1982
0039 1 !
0040 1 ! MODIFIED BY:
0041 1 !
0042 1 !       V03-002 CWH3002         CW Hobbs                12-Apr-1984
0043 1 !               Change the getdvi to use FULLDEVNAM.
0044 1 !
0045 1 !
0046 1 !--
0047 1
0048 1 ! Include files:
0049 1 !
0050 1 MACRO $module_name_string = 'exch$util' %;       ! The require file needs to know our module name
0051 1 REQUIRE 'SRC$:EXCREQ'
0052 1     ;
```

```
1 %SBTTL 'Module table of contents'
1
1 ! Module table of contents:
1 !
1 FORWARD ROUTINE
1       exch$util_block_check : jsb_r0r1r2 NOVALUE, ! Check the block type and size fields
1       exch$util_dos11ctx_allocate,                ! Allocate a DOS-11 file context block
1       exch$util_dos11ctx_release : NOVALUE,       ! Release it
1       exch$util_fao_buffer,                       ! Pass arguments through FAO service
1       exch$util_filb_allocate,                    ! Allocate a file block
1       exch$util_filb_release : NOVALUE,           ! Release a file block
1       exch$util_file_error,                       ! Signal an RMS error
1       exch$util_find_mounted_volb,                ! Locate a mounted volume block in the volb in-use queue
1       exch$util_namb_allocate,                    ! Allocate a name block
1       exch$util_namb_release : NOVALUE,           ! Release a name block
1       exch$util_radix50_from_ascii,               ! Convert an ascii string to radix50
1       exch$util_radix50_to_ascii,                 ! Convert a radix50 string to ascii
1       exch$util_rmsb_allocate,                    ! Allocate a file information block
1       exch$util_rmsb_release : NOVALUE,           ! Release a file information block
1       exch$util_rt11ctx_allocate,                 ! Allocate an RT-11 file context block
1       exch$util_rt11ctx_release : NOVALUE,        ! Release it
1       exch$util_vm_allocate,                      ! Call LIB$GET_VM and signal errors
1       exch$util_vm_allocate_zeroed,               ! Call LIB$GET_VM, clear memory and signal errors
1       exch$util_vm_release   : NOVALUE,           ! Call LIB$FREE_VM and signal errors
1       exch$util_vol_getdvi,                       ! Fill in the device characteristics fields in a volb
1       exch$util_volb_allocate,                    ! Allocate a volume block
1       exch$util_volb_release : NOVALUE,           ! Release a volume block
1       exch$util_up_case      : NOVALUE jsb_r1r2r3 ! Convert string to uppercase
1       ;
1
1 ! EXCHANGE facility routines
1 !
1 !EXTERNAL ROUTINE
1 !     ;
1
1 ! Equated symbols:
1 !
1 !LITERAL
1 !     ;
1
1 ! Bound declarations:
1 !
1 !BIND
1 !     ;
```

```
;    99           0193  1 GLOBAL ROUTINE exch$util_block_check (addr : $ref_bblock, code,          %SBTTL 'exch$util_block_check'
;   100           0194  1                                        size_type : VECTOR [2, WORD]) : jsb_r0r1r2 NOVALUE =
;   101           0195  2 BEGIN
;   102           0196  2 !++
;   103           0197  2 !
;   104           0198  2 ! FUNCTIONAL DESCRIPTION:
;   105           0199  2 !
;   106           0200  2 !       This routine checks a data structure for correct size and type fields
;   107           0201  2 !
;   108           0202  2 ! INPUTS:
;   109           0203  2 !
;   110           0204  2 !       addr      - address of the block
;   111           0205  2 !       code      - error code to display if the block doesn't pass
;   112           0206  2 !       size_type - size and type values, size is in high word, type in low word
;   113           0207  2 !
;   114           0208  2 ! IMPLICIT INPUTS:
;   115           0209  2 !
;   116           0210  2 !       none
;   117           0211  2 !
;   118           0212  2 ! OUTPUTS:
;   119           0213  2 !
;   120           0214  2 !       none
;   121           0215  2 !
;   122           0216  2 ! IMPLICIT OUTPUTS:
;   123           0217  2 !
;   124           0218  2 !       none
;   125           0219  2 !
;   126           0220  2 ! ROUTINE VALUE:
;   127           0221  2 !
;   128           0222  2 !       none
;   129           0223  2 !
;   130           0224  2 ! SIDE EFFECTS:
;   131           0225  2 !
;   132           0226  2 !       If the block does not pass, the image is terminated
;   133           0227  2 !--
;   134           0228  2
;   135           0229  2 BIND
;   136           0230  2     size = size_type [1] : WORD,
;   137           0231  2     type = size_type [0] : WORD;
;   138           0232  2
;   139           0233  2 IF .addr EQL 0              ! Add 1000 to the error code if the block address is zero, this lets
;   140           0234  2 THEN                        !  us distinguish missing from bad blocks without defining additional error codes
;   141           0235  2     $exch_signal_stop (exch$_blockcheck0, 1, (1000+.code));
;   142           0236  2
;   143           0237  2 IF   .addr [excg$w_size] NEQ .size
;   144           0238  2   OR
;   145           0239  2      .addr [excg$b_type] NEQ .type
;   146           0240  2 THEN
;   147         P 0241  2     $exch_signal_stop (exch$_blockcheck, 6, .code, .addr,
;   148           0242  2                  .addr [excg$w_size], .size, .addr [excg$b_type], .type);
;   149           0243  2
;   150           0244  2 RETURN;
;   151           0245  1 END;


                                                              .TITLE  EXCH$UTIL Facility-wide misc routines
                                                              .IDENT  \V04-000\
                                                                     .EXTRN   EXCH$_BLOCKCHECKO
                                                                     .EXTRN   LIB$STOP, EXCH$_BLOCKCHECK

                                                                     .PSECT   EXCH$UTIL_CODE,NOWRT,2

                                            52  DD 00000 EXCH$UTIL_BLOCK_CHECK::
                                                                     PUSHL    R2                                          ; 0193
                                            50  D5 00002             TSTL     ADDR                                        ; 0233
                                            15  12 00004             BNEQ     1$
                                   03E8     C1  9F 00006             PUSHAB   1000(CODE)                                  ; 0235
                                            01  DD 0000A             PUSHL    #1
                              00000000G     8F  DD 0000C             PUSHL    #EXCH$_BLOCKCHECKO
              00000000G  00                 03  FB 00012             CALLS    #3, LIB$STOP
                                            32  11 00019             BRB      3$
                     02  AE          08     A0  B1 0001B 1$:         CMPW     8(ADDR), SIZE                               ; 0237
                                            09  12 00020             BNEQ     2$
                         52          0A     A0  9A 00022             MOVZBL   10(ADDR), R2                                ; 0239
                         6E                 52  B1 00026             CMPW     R2, TYPE
                                            22  13 00029             BEQL     3$
                         7E                 6E  3C 0002B 2$:         MOVZWL   TYPE, -(SP)                                 ; 0242
                         7E          0A     A0  9A 0002E             MOVZBL   10(ADDR), -(SP)
                         7E          0A     AE  3C 00032             MOVZWL   SIZE, -(SP)
                         7E          08     A0  3C 00036             MOVZWL   8(ADDR), -(SP)
                                            50  DD 0003A             PUSHL    ADDR
                                            51  DD 0003C             PUSHL    CODE
                                            06  DD 0003E             PUSHL    #6
                              00000000G     8F  DD 00040             PUSHL    #EXCH$_BLOCKCHECK
              00000000G  00                 08  FB 00046             CALLS    #8, LIB$STOP
                         5E                 04  C0 0004D 3$:         ADDL2    #4, SP                                      ; 0245
                                                05 00050             RSB

; Routine Size:  81 bytes,    Routine Base:  EXCH$UTIL_CODE + 0000
```

```
1 GLOBAL ROUTINE exch$util_dos11ctx_allocate (volb, filb) =         %SBTTL 'exch$util_dos11ctx_allocate (volb, f
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       This routine allocates one DOS-11 file context block.  If one is available, it is moved from the ava
2 !       queue to the in-use queue.  If none are available, then a fresh block is created and placed on the i
2 !       queue.
2 !
2 ! INPUTS:
2 !
2 !       volb - pointer to the associated volb
2 !       filb - pointer to the associated filb
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       exch$a_gbl [excg$q_dos11ctx_all] - list of allocated file blocks
2 !       exch$a_gbl [excg$q_dos11ctx_avl] - queue of available file blocks
2 !       exch$a_gbl [excg$q_dos11ctx_use] - queue of file blocks in use
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       address of the allocated file block
2 !
2 ! SIDE EFFECTS:
2 !
2 !       All errors are fatal
2 !--
2
2 LOCAL
2     offset,                                       ! Local temporary
2     ptr         : $ref_bblock,                    ! A local pointer to the dos11ctx
2     status
2     ;
2
2
2 ! First, try to find one in the available queue
2 !
2 ptr = $queue_remove_head (exch$a_gbl [excg$q_dos11ctx_avl]);
2
2 ! If we didn't find one, then it will have to be created
2 !
2 IF .ptr EQL 0
2 THEN
3     BEGIN
3
3     ! Allocate a fresh dos11ctx from virtual memory.  The entire block has been cleared to nulls
3     !
```

```
;  210      0303 3          ptr = exch$util_vm_allocate_zeroed (exchblk$s_dos11ctx);
;  211      0304 3
;  212      0305 3          ! Place the dos11ctx at the head of the list of allocated blocks
;  213      0306 3          !
;  214      0307 3          ptr [dos11ctx$a_alloc] = .exch$a_gbl [excg$a_dos11ctx_alloc];
;  215      0308 3          exch$a_gbl [excg$a_dos11ctx_alloc] = .ptr;
;  216      0309 3
;  217      0310 3          ! Set the block identification fields
;  218      0311 3          !
;  219      0312 3          $block_init (.ptr, dos11ctx);
;  220      0313 3
;  221      0314 2          END;
;  222      0315 2
;  223      0316 2    ! Check our block type, fatal error if any problems
;  224      0317 2    !
;  225      0318 2    $block_check (2, .ptr, dos11ctx, 578);
;  226      0319 2
;  227      0320 2    ! Set the last part of the block to nulls
;  228      0321 2    !
;  229      0322 2    CH$FILL (0, dos11ctx$k_end_zero - dos11ctx$k_start_zero, .ptr + dos11ctx$k_start_zero);
;  230      0323 2
;  231      0324 2    ! Insert the block at the head of the in-use queue
;  232      0325 2    !
;  233      0326 2    $queue_insert_head (ptr [dos11ctx$q_header], exch$a_gbl [excg$q_dos11ctx_use]);
;  234      0327 2
;  235      0328 2    ! Set the two associated fields
;  236      0329 2    !
;  237      0330 2    ptr [dos11ctx$a_assoc_volb] = .volb;
;  238      0331 2    ptr [dos11ctx$a_assoc_filb] = .filb;
;  239      0332 2
;  240      0333 2    ! Return the address of the file block to the caller
;  241      0334 2    !
;  242      0335 2    RETURN .ptr;
;  243      0336 2
;  244      0337 1    END;


                                                              .EXTRN  EXCH$A_GBL

                                    00FC 00000                .ENTRY  EXCH$UTIL_DOS11CTX_ALLOCATE, Save R2,R3,R4,-; 0246
                                                                      R5,R6,R7                                    ;
                     57 00000000G   EF   9E 00002             MOVAB   EXCH$A_GBL, R7                              ;
         51          67 00000064    8F   C1 00009             ADDL3   #100, EXCH$A_GBL, R1                        ; 0293
                     50        00   B1   0F 00011             REMQUE  @0(R1), _T_                                 ;
                                    04   1C 00015             BVC     1$                                          ;
                                    56   D4 00017             CLRL    PTR                                         ;
                                    03   11 00019             BRB     2$                                          ;
                     56             50   D0 0001B   1$:       MOVL    _T_, PTR                                    ;
                                    21   12 0001E   2$:       BNEQ    3$                                          ; 0297
                     7E        8A   8F   9A 00020             MOVZBL  #138, -(SP)                                 ; 0303
              0000V  CF             01   FB 00024             CALLS   #1, EXCH$UTIL_VM_ALLOCATE_ZEROED            ;
                     56             50   D0 00029             MOVL    R0, PTR                                     ;
                     50             67   D0 0002C             MOVL    EXCH$A_GBL, R0                              ; 0307
              0C     A6        58   A0   D0 0002F             MOVL    88(R0), 12(PTR)                             ;
              58     A0             56   D0 00034             MOVL    PTR, 88(R0)                                 ; 0308
              08     A6        8A   8F   9B 00038             MOVZBW  #138, 8(PTR)                                ; 0312
```

```
                          0A   A6              04  8E 0003D          MNEGB   #4, 10(PTR)                            ;
                               52 008A00FC     8F  D0 00041  3$:     MOVL    #9044220, R2                           ; 0318
                               51     0242     8F  3C 00048          MOVZWL  #578, R1                               ;
                               50              56  D0 0004D          MOVL    PTR, R0                                ;
                                  00000000G    EF  16 00050          JSB     EXCH$UTIL_BLOCK_CHECK                  ;
   006E   8F           00      6E              00  2C 00056          MOVC5   #0, (SP), #0, #T10, 28(PTR)            ; 0322
                                        1C     A6     0005D                                                         ;
                       50      67 0000005C     8F  C1 0005F          ADDL3   #92, EXCH$A_GBL, R0                    ; 0326
                               60              66  0E 00067          INSQUE  (PTR), (R0)                            ;
                          14   A6       04     AC  D0 0006A          MOVL    VOLB, 20(PTR)                          ; 0330
                          10   A6       08     AC  D0 0006F          MOVL    FILB, 16(PTR)                          ; 0331
                               50              56  D0 00074          MOVL    PTR, R0                                ; 0335
                                               04     00077          RET                                            ; 0337

; Routine Size:  120 bytes,    Routine Base:  EXCH$UTIL_CODE + 0051
```

```
1 GLOBAL ROUTINE exch$util_dos11ctx_release (addr) : NOVALUE =    %SBTTL 'exch$util_dos11ctx_release (addr)'
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       This routine deallocates one dos11ctx.  The block is moved from the in-use queue to the available qu
2 !
2 ! INPUTS:
2 !
2 !       addr - address of the block to release
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       exch$a_gbl [excg$q_dos11ctx_avl] - queue of available file blocks
2 !       exch$a_gbl [excg$q_dos11ctx_use] - queue of file blocks in use
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       none
2 !
2 ! SIDE EFFECTS:
2 !
2 !       All errors are fatal
2 !--
2
2 LOCAL
2     ptr         : $ref_bblock,                    ! A local pointer to the dos11ctx
2     status
2     ;
2
2
2 ! First, move the pointer to a local variable
2 !
2 ptr = .addr;
2
2 ! Check our block type, fatal error if any problems
2 !
2 $block_check (2, .ptr, dos11ctx, 579);
2
2 ! If there is a buffer allocated, free it
2 !
2 IF .ptr [dos11ctx$a_buffer] NEQ 0
2 THEN
2     exch$util_vm_release (ctx$k_buffer_length, .ptr [dos11ctx$a_buffer]);
2
2 ! Clear the pointers in the part of the block before the automatic zero
2 !
2 ptr [dos11ctx$a_assoc_filb] = 0;
```

```
;   303         0395  2 ptr [dos11ctx$a_assoc_volb] = 0;
;   304         0396  2 ptr [dos11ctx$a_buffer]     = 0;
;   305         0397  2
;   306         0398  2 ! Remove the dos11ctx from where ever it is in the in-use queue
;   307         0399  2 !
;   308         0400  2 $queue_remove (ptr [dos11ctx$q_header]);
;   309         0401  2
;   310         0402  2 ! Place the dos11ctx at the end of the available queue and the head of the in-use queue
;   311         0403  2 !
;   312         0404  2 $queue_insert_tail (ptr [dos11ctx$q_header], exch$a_gbl [excg$q_dos11ctx_avl]);
;   313         0405  2
;   314         0406  2 RETURN;
;   315         0407  1 END;



                                               000C 00000           .ENTRY  EXCH$UTIL_DOS11CTX_RELEASE, Save R2,R3           ; 0338
                        53          04     AC    D0 00002           MOVL    ADDR, PTR                                        ; 0380
                        52   008A00FC      8F    D0 00006           MOVL    #9044220, R2                                     ; 0384
                        51        0243     8F    3C 0000D           MOVZWL  #579, R1                                         ;
                        50                 53    D0 00012           MOVL    PTR, R0                                          ;
                             00000000G     EF    16 00015           JSB     EXCH$UTIL_BLOCK_CHECK                            ;
                                    18     A3    D5 0001B           TSTL    24(PTR)                                          ; 0388
                                           0D    13 0001E           BEQL    1$                                               ;
                                    18     A3    DD 00020           PUSHL   24(PTR)                                          ; 0390
                        7E        1800     8F    3C 00023           MOVZWL  #6144, -(SP)                                     ;
                   0000V    CF             02    FB 00028           CALLS   #2, EXCH$UTIL_VM_RELEASE                         ;
                                    10     A3    7C 0002D 1$:       CLRQ    16(PTR)                                          ; 0394
                                    18     A3    D4 00030           CLRL    24(PTR)                                          ; 0396
                        50                 63    0F 00033           REMQUE  (PTR), _T                                        ; 0400
             50 00000000G    EF 00000064   8F    C1 00036           ADDL3   #100, EXCH$A_GBL, R0                             ; 0404
                   04    B0                63    0E 00042           INSQUE  (PTR), @4(R0)                                    ;
                                           04 00046                 RET                                                      ; 0407

; Routine Size:  71 bytes,     Routine Base:  EXCH$UTIL_CODE + 00C9
;  317      0408  1 GLOBAL ROUTINE exch$util_fao_buffer (ctrstr : REF VECTOR[2], args : VECTOR [4]) =        %SBTTL 'exch$util_fa
;  318      0409  2 BEGIN
;  319      0410  2 !++
;  320      0411  2 !
;  321      0412  2 ! FUNCTIONAL DESCRIPTION:
;  322      0413  2 !
;  323      0414  2 !       This routine passes an ascii string through the FAO system service with any number of specified para
;  324      0415  2 !
;  325      0416  2 ! INPUTS:
;  326      0417  2 !
;  327      0418  2 !       ctrstr  Address of FAO control string descriptor
;  328      0419  2 !       args    Any number of additional arguments
;  329      0420  2 !
;  330      0421  2 ! IMPLICIT INPUTS:
;  331      0422  2 !
;  332      0423  2 !       none
;  333      0424  2 !
;  334      0425  2 ! OUTPUTS:
;  335      0426  2 !
;  336      0427  2 !       none
;  337      0428  2 !
;  338      0429  2 ! IMPLICIT OUTPUTS:
;  339      0430  2 !
;  340      0431  2 !       none
;  341      0432  2 !
;  342      0433  2 ! ROUTINE VALUE:
;  343      0434  2 !
;  344      0435  2 !       Address of formatted descriptor
;  345      0436  2 !
;  346      0437  2 ! SIDE EFFECTS:
;  347      0438  2 !
;  348      0439  2 !       none
;  349      0440  2 !--
;  350      0441  2
;  351      0442  2 BIND
;  352      0443  2     desc = exch$a_gbl [excg$t_fao_buffer] : VECTOR [3]
;  353      0444  2     ;
;  354      0445  2
;  355      0446  2
;  356      0447  2 desc [0] = excg$s_fao_buffer-8;                     ! Set up result descriptor
;  357      0448  2 desc [1] = desc [2];
;  358      0449  2
;  359      0450  2 $faol (ctrstr=.ctrstr, outlen=desc, outbuf=desc, prmlst=args);
;  360      0451  2
;  361      0452  2 RETURN desc;
;  362      0453  1 END;



                                                                        .EXTRN  SYS$FAOL

                                               0004 00000              .ENTRY  EXCH$UTIL_FAO_BUFFER, Save R2                  ; 0408
              52 00000000G  EF 000000E4   8F    C1 00002              ADDL3   #228, EXCH$A_GBL, R2                           ; 0443
                        62           FA   8F    9A 0000E              MOVZBL  #250, (R2)                                     ; 0447
                 04     A2           08   A2    9E 00012              MOVAB   8(R2), 4(R2)                                   ; 0448
                                     08   AC    9F 00017              PUSHAB  ARGS                                           ; 0450
                                          52    DD 0001A              PUSHL   R2                                             ;
```

```
                                           52  DD 0001C           PUSHL   R2                                            ;
                                     04    AC  DD 0001E           PUSHL   CTRSTR                                        ;
                      00000000G  00        04  FB 00021           CALLS   #4, SYS$FAOL                                  ;
                                 50        52  D0 00028           MOVL    R2, R0                                        ; 0452
                                               04 0002B           RET                                                   ; 0453

; Routine Size:  44 bytes,    Routine Base:  EXCH$UTIL_CODE + 0110
```

```
1 GLOBAL ROUTINE exch$util_filb_allocate =          %SBTTL 'exch$util_filb_allocate'
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       This routine allocates one $FILB.  If $FILBs are available, one is moved from the available queue to
2 !       in-use queue.  If none are available, then a fresh $FILB is created and placed on the in-use queue.
2 !
2 ! INPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       exch$a_gbl [excg$a_filb_all] - list of allocated file blocks
2 !       exch$a_gbl [excg$q_filb_avl] - queue of available file blocks
2 !       exch$a_gbl [excg$q_filb_use] - queue of file blocks in use
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       address of the allocated file block
2 !
2 ! SIDE EFFECTS:
2 !
2 !       All errors are fatal
2 !--
2
2 LOCAL
2     ptr         : $ref_bblock,                  ! A local pointer to the filb
2     status
2     ;
2
2
2 ! First, try to find one in the available queue
2 !
2 ptr = $queue_remove_head (exch$a_gbl [excg$q_filb_avl]);
2
2 ! If we didn't find one, then it will have to be created
2 !
2 IF .ptr EQL 0
2 THEN
3     BEGIN
3
3     ! Allocate a fresh filb from virtual memory.
3     !
3     ptr = exch$util_vm_allocate (exchblk$s_filb);
3
3     ! Place the filb at the head of the list of allocated blocks
```

```
;   421         0511  3         !
;   422         0512  3         ptr [filb$a_alloc] = .exch$a_gbl [excg$a_filb_alloc];
;   423         0513  3         exch$a_gbl [excg$a_filb_alloc] = .ptr;
;   424         0514  3
;   425         0515  3         ! Init the dynamic strings
;   426         0516  3         !
;   427         0517  3         $dyn_str_desc_init (ptr [filb$q_name_string]);
;   428         0518  3
;   429         0519  3         ! Set the block identification fields
;   430         0520  3         !
;   431         0521  3         $block_init (.ptr, filb);
;   432         0522  3
;   433         0523  2         END;
;   434         0524  2
;   435         0525  2     ! Check our block type, fatal error if any problems
;   436         0526  2     !
;   437         0527  2     $block_check (2, .ptr, filb, 481);
;   438         0528  2
;   439         0529  2     ! Place the filb at the head of the in-use queue
;   440         0530  2     !
;   441         0531  2     $queue_insert_head (ptr [filb$q_header], exch$a_gbl [excg$q_filb_use]);
;   442         0532  2
;   443         0533  2     ! Set the last part of the block to nulls
;   444         0534  2     !
;   445         0535  2     CH$FILL (0, filb$k_end_zero - filb$k_start_zero, .ptr + filb$k_start_zero);
;   446         0536  2
;   447         0537  2     ! Return the address of the file block to the caller
;   448         0538  2     !
;   449         0539  2     RETURN .ptr;
;   450         0540  2
;   451         0541  1     END;


                                                                     .EXTRN  EXCH$GQ_DYN_STR_TEMPLATE

                                                   00FC 00000         .ENTRY  EXCH$UTIL_FILB_ALLOCATE, Save R2,R3,R4,R5,-  ; 0454
                                                                              R6,R7                                        ;
                                57 00000000G   EF   9E 00002         MOVAB   EXCH$A_GBL, R7                               ;
                51              67 00000078    8F   C1 00009         ADDL3   #120, EXCH$A_GBL, R1                         ; 0498
                                50        00   B1   0F 00011         REMQUE  @0(R1), _T_                                  ;
                                               04   1C 00015         BVC     1$                                           ;
                                               56   D4 00017         CLRL    PTR                                          ;
                                               03   11 00019         BRB     2$                                           ;
                                56             50   D0 0001B 1$:     MOVL    _T_, PTR                                     ;
                                               2E   12 0001E 2$:     BNEQ    3$                                           ; 0502
                                7E      035B   8F   3C 00020         MOVZWL  #859, -(SP)                                  ; 0508
                        0000V   CF             01   FB 00025         CALLS   #1, EXCH$UTIL_VM_ALLOCATE                    ;
                                56             50   D0 0002A         MOVL    R0, PTR                                      ;
                                50             67   D0 0002D         MOVL    EXCH$A_GBL, R0                               ; 0512
                        0C      A6        6C   A0   D0 00030         MOVL    108(R0), 12(PTR)                             ;
                        6C      A0             56   D0 00035         MOVL    PTR, 108(R0)                                 ; 0513
                                50        10   A6   9E 00039         MOVAB   16(PTR), R0                                  ; 0517
                                60 00000000G   EF   7D 0003D         MOVQ    TMPL, (R0)                                   ;
                        08      A6      035B   8F   B0 00044         MOVW    #859, 8(PTR)                                 ; 0521
                        0A      A6             06   8E 0004A         MNEGB   #6, 10(PTR)                                  ;
                                52 035B00FA    8F   D0 0004E 3$:     MOVL    #56295674, R2                                ; 0527
                                51      01E1    8F  3C 00055         MOVZWL  #481, R1                              ;
                                50              56  D0 0005A         MOVL    PTR, R0                               ;
                                   00000000G    EF  16 0005D         JSB     EXCH$UTIL_BLOCK_CHECK                 ;
                    50          67 00000070     8F  C1 00063         ADDL3   #112, EXCH$A_GB[, R0                  ; 0531
                                60              66  0E 0006B         INSQUE  (PTR), (R0)                           ;
   0042    8F       00          6E              00  2C 0006E         MOVC5   #0, (SP), #0, #66, 24(PTR)            ; 0535
                                        18      A6     00075                                                       ;
                                50              56  D0 00077         MOVL    PTR, R0                               ; 0539
                                                    04 0007A         RET                                           ; 0541

; Routine Size:  123 bytes,    Routine Base:  EXCH$UTIL_CODE + 013C
```

```
1 GLOBAL ROUTINE exch$util_filb_release (addr) : NOVALUE =          %SBTTL 'exch$util_filb_release (addr)'
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       This routine deallocates one $FILB.  The $FILB is moved from the in-use queue to the available queue
2 !
2 ! INPUTS:
2 !
2 !       addr - address of the block to release
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       exch$a_gbl [excg$q_filb_avl] - queue of available file blocks
2 !       exch$a_gbl [excg$q_filb_use] - queue of file blocks in use
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       none
2 !
2 ! SIDE EFFECTS:
2 !
2 !       All errors are fatal
2 !--
2
2 LOCAL
2     ptr         : $ref_bblock,                        ! A local pointer to the filb
2     status
2     ;
2
2
2 ! First, move the pointer to a local variable
2 !
2 ptr = .addr;
2
2 ! Check our block type, fatal error if any problems
2 !
2 $block_check (2, .ptr, filb, 482);
2
2 ! Remove the filb from where ever it is in the in-use queue
2 !
2 $queue_remove (ptr [filb$q_header]);
2
2 ! Place the filb at the end of the available queue.
2 !
2 $queue_insert_tail (ptr [filb$q_header], exch$a_gbl [excg$q_filb_avl]);
2
2 RETURN;
```

```
;  510          0599  1 END;



                                          000C 00000          .ENTRY  EXCH$UTIL_FILB_RELEASE, Save R2,R3              ; 0542
                  53          04   AC   D0 00002          MOVL    ADDR, PTR                                                ; 0584
                  52 035B00FA   8F   D0 00006          MOVL    #56295674, R2                                            ; 0588
                  51        01E2   8F   3C 0000D          MOVZWL  #482, R1                                                 ;
                  50               53   D0 00012          MOVL    PTR, R0                                                  ;
                       00000000G   EF   16 00015          JSB     EXCH$UTIL_BLOCK_CHECK                                    ;
                  50               63   0F 0001B          REMQUE  (PTR), T                                                 ; 0592
      50 00000000G   EF 00000078   8F   C1 0001E          ADDL3   #120, EXCH$A_GBL, R0                                     ; 0596
                  04   B0          63   0E 0002A          INSQUE  (PTR), @4(R0)                                            ;
                                        04 0002E          RET                                                              ; 0599

; Routine Size:  47 bytes,    Routine Base:  EXCH$UTIL_CODE + 01B7
```

```
GLOBAL ROUTINE exch$util_file_error (msg, rms_status, fabb : $ref_bblock, stv) =         %SBTTL 'exch$util_fi
BEGIN
!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine signals an RMS error.  The appropriate file name for the signal is found by
!       examining the contents of the nam block.
!
! INPUTS:
!
!       msg         Error message value, assumed to have one !AS FAO argument
!       rms_status  Error message from RMS call
!       fabb        Pointer to FAB, used to locate nam block
!       stv         The RMS STV error from the FAB or RAB
!
! IMPLICIT INPUTS:
!
!       RMS nam block attached to the FAB (fabb)
!
! OUTPUTS:
!
!       none
!
! IMPLICIT OUTPUTS:
!
!       none
!
! ROUTINE VALUE:
!
!       msg - with inhibit signal bit set
!
! SIDE EFFECTS:
!
!       An error will be signalled
!--

LOCAL
    tmp_desc : $desc_block,                            ! A descriptor for the file name
    nam_blk : $ref_bblock;                             ! Pointer to the name block

nam_blk = .fabb [fab$l_nam];                           ! Get pointer to the name block
tmp_desc [dsc$b_class] = dsc$k_class_s;                ! Static desc
tmp_desc [dsc$b_dtype] = dsc$k_dtype_t;                ! String desc

IF .nam_blk [nam$b_rsl] GTRU 0
THEN
    BEGIN
    tmp_desc [dsc$w_length]  = .nam_blk [nam$b_rsl];      ! Create file name desc
    tmp_desc [dsc$a_pointer] = .nam_blk [nam$l_rsa];      ! ...
    END
ELSE IF .nam_blk [nam$b_esl] GTRU 0
THEN
    BEGIN
    tmp_desc [dsc$w_length]  = .nam_blk [nam$b_esl];      ! Create file name desc
    tmp_desc [dsc$a_pointer] = .nam_blk [nam$l_esa];      ! ...
    END
```

```
;  569          0657 2 ELSE
;  570          0658 3     BEGIN
;  571          0659 3     tmp_desc [dsc$w_length]  = .fabb [fab$b_fns];         ! Create file name desc
;  572          0660 3     tmp_desc [dsc$a_pointer] = .fabb [fab$l_fna];         ! ...
;  573          0661 2     END;
;  574          0662 2
;  575          0663 2 SIGNAL (.msg, 1, tmp_desc, .rms_status, .stv);
;  576          0664 2
;  577          0665 2 RETURN .msg;
;  578          0666 2
;  579          0667 1 END;



                                         0000 00000          .ENTRY  EXCH$UTIL_FILE_ERROR, Save nothing                  : 0600
                 5E               08      C2 00002          SUBL2   #8, SP                                              :
                 51               0C  AC  D0 00005          MOVL    FABB, R1                                            : 0641
                 50               28  A1  D0 00009          MOVL    40(R1), NAM_BLK                                     :
          02     AE             010E  8F  B0 0000D          MOVW    #270, TMP_DESC+2                                    : 0643
                                  03  A0  95 00013          TSTB    3(NAM_BLK)                                          : 0645
                                      0B  13 00016          BEQL    1$                                                  :
                 6E               03  A0  9B 00018          MOVZBW  3(NAM_BLK), TMP_DESC                                : 0648
          04     AE               04  A0  D0 0001C          MOVL    4(NAM_BLK), TMP_DESC+4                              : 0649
                                      19  11 00021          BRB     3$                                                  : 0645
                                  0B  A0  95 00023 1$:      TSTB    11(NAM_BLK)                                         : 0651
                                      0B  13 00026          BEQL    2$                                                  :
                 6E               0B  A0  9B 00028          MOVZBW  11(NAM_BLK), TMP_DESC                               : 0654
          04     AE               0C  A0  D0 0002C          MOVL    12(NAM_BLK), TMP_DESC+4                             : 0655
                                      09  11 00031          BRB     3$                                                  : 0651
                 6E               34  A1  9B 00033 2$:      MOVZBW  52(R1), TMP_DESC                                    : 0659
          04     AE               2C  A1  D0 00037          MOVL    44(R1), TMP_DESC+4                                  : 0660
                                  10  AC  DD 0003C 3$:      PUSHL   STV                                                 : 0663
                                  08  AC  DD 0003F          PUSHL   RMS_STATUS                                          :
                                  08  AE  9F 00042          PUSHAB  TMP_DESC                                            :
                                      01  DD 00045          PUSHL   #1                                                  :
                                  04  AC  DD 00047          PUSHL   MSG                                                 :
       00000000G  00                  05  FB 0004A          CALLS   #5, LIB$SIGNAL                                      :
                 50               04  AC  D0 00051          MOVL    MSG, R0                                             : 0665
                                          04 00055          RET                                                         : 0667

; Routine Size:  86 bytes,    Routine Base:  EXCH$UTIL_CODE + 01E6
```

```
GLOBAL ROUTINE exch$util_find_mounted_volb (ident : $ref_bvector) =      %SBTTL 'exch$util_find_mounted_volb
BEGIN
!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine scans the queue of in-use volume blocks to see if any have the same name as the
!       input name.
!
! INPUTS:
!
!       ident - address of the first byte
!
! IMPLICIT INPUTS:
!
!       none
!
! OUTPUTS:
!
!       none
!
! IMPLICIT OUTPUTS:
!
!       none
!
! ROUTINE VALUE:
!
!       0 if name not found, address of volb if name is found
!
! SIDE EFFECTS:
!
!       none
!--

$dbgtrc_prefix ('util_find_mounted_volb> ');

LOCAL
    ptr : $ref_bblock,                          ! Pointer to scan along the queue
    status
    ;

! Get the first volb, and scan the list of file names
!
ptr = .exch$a_gbl [excg$a_volb_use_flink];

WHILE .ptr NEQA exch$a_gbl [excg$q_volb_use]
DO
    BEGIN

    $block_check (2, .ptr, volb, 483);

    IF CH$EQL (volb$s_vol_ident, .ident, volb$s_vol_ident, ptr [volb$t_vol_ident])
    THEN
        RETURN .ptr;

    ptr = .ptr [volb$a_flink];                  ! Advance to next volb in the in-use queue
```

```
;  638        0725  3
;  639        0726  2        END;
;  640        0727  2
;  641        0728  2  RETURN 0;
;  642        0729  1  END;



                                              003C 00000          .ENTRY  EXCH$UTIL_FIND_MOUNTED_VOLB, Save R2,R3,R4,-; 0668
                                                                          R5                                            ;
                          55 00000000G   EF   9E 00002            MOVAB   EXCH$A_GBL, R5                                ;
                          50             65   D0 00009            MOVL    EXCH$A_GBL, R0                                ; 0712
                          54       00C0  C0   D0 0000C            MOVL    192(R0), PTR                                  ;
              50          65 000000C0    8F   C1 00011 1$:        ADDL3   #192, EXCH$A_GBL, R0                          ; 0714
                          50             54   D1 00019            CMPL    PTR, R0                                       ;
                                         28   13 0001C            BEQL    3$                                            ;
                          52 041B00F3    8F   D0 0001E            MOVL    #68878579, R2                                 ; 0718
                          51       01E3  8F   3C 00025            MOVZWL  #483, R1                                      ;
                          50             54   D0 0002A            MOVL    PTR, R0                                       ;
                             00000000G   EF   16 0002D            JSB     EXCH$UTIL_BLOCK_CHECK                         ;
         69    A4      04    BC    0080  8F   29 00033            CMPC3   #128, @IDENT, 105(PTR)                        ; 0720
                                         04   12 0003B            BNEQ    2$                                            ;
                          50             54   D0 0003D            MOVL    PTR, R0                                       ; 0722
                                            04 00040              RET                                                   ;
                          54             64   D0 00041 2$:        MOVL    (PTR), PTR                                    ; 0724
                                         CB   11 00044            BRB     1$                                            ; 0714
                                         50   D4 00046 3$:        CLRL    R0                                            ; 0728
                                            04 00048              RET                                                   ; 0729

; Routine Size:  73 bytes,    Routine Base:  EXCH$UTIL_CODE + 023C
```

```
1 GLOBAL ROUTINE exch$util_namb_allocate =          %SBTTL 'exch$util_namb_allocate'
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       This routine allocates one $NAMB.  If $NAMBs are available, one is moved from the available queue to
2 !       in-use queue.  If none are available, then a fresh $NAMB is created and placed on the in-use queue.
2 !
2 ! INPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       exch$a_gbl [excg$q_namb_all] - list of allocated name blocks
2 !       exch$a_gbl [excg$q_namb_avl] - queue of available name blocks
2 !       exch$a_gbl [excg$q_namb_use] - queue of name blocks in use
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       address of the allocated name block
2 !
2 ! SIDE EFFECTS:
2 !
2 !       All errors are fatal
2 !--
2
2 LOCAL
2     offset,                                      ! Local temporary
2     ptr          : $ref_bblock,                  ! A local pointer to the namb
2     status
2     ;
2
2
2 ! First, try to find one in the available queue
2 !
2 ptr = $queue_remove_head (exch$a_gbl [excg$q_namb_avl]);
2
2 ! If we didn't find one, then it will have to be created
2 !
2 IF .ptr EQL 0
2 THEN
3     BEGIN
3
3     ! Allocate a fresh namb from virtual memory.  The entire block has been cleared to nulls
3     !
3     ptr = exch$util_vm_allocate_zeroed (exchblk$s_namb);
3
```

```
;  701          0787 3        ! Place the namb at the head of the list of allocated blocks
;  702          0788 3        !
;  703          0789 3        ptr [namb$a_alloc] = .exch$a_gbl [excg$a_namb_alloc];
;  704          0790 3        exch$a_gbl [excg$a_namb_alloc] = .ptr;
;  705          0791 3
;  706          0792 3        ! Set the block identification fields
;  707          0793 3
;  708          0794 3        $block_init (.ptr, namb);
;  709          0795 3
;  710          0796 3        ! Initialize the dynamic strings
;  711          0797 3
;  712          0798 3        $dyn_str_desc_init (ptr [namb$q_input]);
;  713          0799 3        $dyn_str_desc_init (ptr [namb$q_fullname]);
;  714          0800 3        $dyn_str_desc_init (ptr [namb$q_expanded]);
;  715          0801 3        $dyn_str_desc_init (ptr [namb$q_result]);
;  716          0802 3        $dyn_str_desc_init (ptr [namb$q_device_dvi]);
;  717          0803 3
;  718          0804 2        END;
;  719          0805 2
;  720          0806 2    ! Check our block type, fatal error if any problems
;  721          0807 2    !
;  722          0808 2    $block_check (2, .ptr, namb, 484);
;  723          0809 2
;  724          0810 2    ! Place the namb at the head of the in-use queue
;  725          0811 2    !
;  726          0812 2    $queue_insert_head (ptr [namb$q_header], exch$a_gbl [excg$q_namb_use]);
;  727          0813 2
;  728          0814 2    ! Set the last part of the block to nulls
;  729          0815 2    !
;  730          0816 2    CH$FILL (0, exchblk$s_namb - namb$k_start_zero, .ptr + namb$k_start_zero);
;  731          0817 2
;  732          0818 2    ! Return the address of the name block to the caller
;  733          0819 2    !
;  734          0820 2    RETURN .ptr;
;  735          0821 2
;  736          0822 1    END;



                                           00FC 00000          .ENTRY   EXCH$UTIL_NAMB_ALLOCATE, Save R2,R3,R4,R5,-  ; 0730
                                                                        R6,R7                                        ;
                           57 00000000G  EF   9E 00002          MOVAB    EXCH$A_GBL, R7                               ;
           51              67 0000008C   8F   C1 00009          ADDL3    #140, EXCH$A_GBL, R1                         ; 0775
                           50       00   B1   0F 00011          REMQUE   @0(R1), _T_                                  ;
                                         04   1C 00015          BVC      1$                                           ;
                                         56   D4 00017          CLRL     PTR                                          ;
                                         03   11 00019          BRB      2$                                           ;
                           56            50   D0 0001B 1$:      MOVL     _T_, PTR                                     ;
                                         6A   12 0001E 2$:      BNEQ     3$                                           ; 0779
                           7E     010A   8F   3C 00020          MOVZWL   #266, -(SP)                                  ; 0785
                   0000V   CF            01   FB 00025          CALLS    #1, EXCH$UTIL_VM_ALLOCATE_ZEROED             ;
                           56            50   D0 0002A          MOVL     R0, PTR                                      ;
                           50            67   D0 0002D          MOVL     EXCH$A_GBL, R0                               ; 0789
                     0C    A6     0080   C0   D0 00030          MOVL     128(R0), 12(PTR)                             ;
                   0080    C0            56   D0 00036          MOVL     PTR, 128(R0)                                 ; 0790
```

```
                    08   A6       010A   8F  B0 0003B         MOVW    #266, 8(PTR)                              ; 0794
                    0A   A6              09  8E 00041         MNEGB   #9, 10(PTR)                               ;
                         50         10   A6  9E 00045         MOVAB   16(PTR), R0                               ; 0798
                         52  00000000G   EF  D0 00049         MOVL    TMPL, R2                                  ;
                         60              52  D0 00050         MOVL    R2, (R0)                                  ;
                         51  00000000G   EF  D0 00053         MOVL    TMPL+4, R1                                ;
                    04   A0              51  D0 0005A         MOVL    R1, 4(R0)                                 ;
                         50         18   A6  9E 0005E         MOVAB   24(PTR), R0                               ; 0799
                         60              52  D0 00062         MOVL    R2, (R0)                                  ;
                    04   A0              51  D0 00065         MOVL    R1, 4(R0)                                 ;
                         50         20   A6  9E 00069         MOVAB   32(PTR), R0                               ; 0800
                         60              52  D0 0006D         MOVL    R2, (R0)                                  ;
                    04   A0              51  D0 00070         MOVL    R1, 4(R0)                                 ;
                         50         28   A6  9E 00074         MOVAB   40(PTR), R0                               ; 0801
                         60              52  D0 00078         MOVL    R2, (R0)                                  ;
                    04   A0              51  D0 0007B         MOVL    R1, 4(R0)                                 ;
                         50         30   A6  9E 0007F         MOVAB   48(PTR), R0                               ; 0802
                         60              52  D0 00083         MOVL    R2, (R0)                                  ;
                    04   A0              51  D0 00086         MOVL    R1, 4(R0)                                 ;
                         52   010A00F7   8F  D0 0008A  3$:    MOVL    #17432823, R2                             ; 0808
                         51       01E4   8F  3C 00091         MOVZWL  #484, R1                                  ;
                         50              56  D0 00096         MOVL    PTR, R0                                   ;
                             00000000G   EF  16 00099         JSB     EXCH$UTIL_BLOCK_CHECK                     ;
              50         67   00000084   8F  C1 0009F         ADDL3   #132, EXCH$A_GBL, R0                      ; 0812
                         60              66  0E 000A7         INSQUE  (PTR), (R0)                               ;
  00A2    8F  00         6E              00  2C 000AA         MOVC5   #0, (SP), #0, #162, 104(PTR)              ; 0816
                                    68   A6     000B1                                                           ;
                         50              56  D0 000B3         MOVL    PTR, R0                                   ; 0820
                                             04 000B6         RET                                               ; 0822

; Routine Size:  183 bytes,    Routine Base:  EXCH$UTIL_CODE + 0285
```

```
1 GLOBAL ROUTINE exch$util_namb_release (addr) : NOVALUE =         %SBTTL 'exch$util_namb_release (addr)'
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       This routine deallocates one $NAMB.  The $NAMB is moved from the in-use queue to the available queue
2 !
2 ! INPUTS:
2 !
2 !       addr - address of the block to release
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       exch$a_gbl [excg$q_namb_avl] - queue of available name blocks
2 !       exch$a_gbl [excg$q_namb_use] - queue of name blocks in use
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       none
2 !
2 ! SIDE EFFECTS:
2 !
2 !       All errors are fatal
2 !--
2
2 LOCAL
2     ptr         : $ref_bblock,                    ! A local pointer to the namb
2     status
2     ;
2
2
2 ! First, move the pointer to a local variable
2 !
2 ptr = .addr;
2
2 ! Check our block type, fatal error if any problems
2 !
2 $block_check (2, .ptr, namb, 485);
2
2 ! Remove the namb from where ever it is in the in-use queue
2 !
2 $queue_remove (ptr [namb$q_header]);
2
2 ! Place the namb at the end of the available queue.
2 !
2 $queue_insert_tail (ptr [namb$q_header], exch$a_gbl [excg$q_namb_avl]);
2
2 RETURN;
```

```
;  795          0880  1 END;



                                          000C 00000        .ENTRY  EXCH$UTIL_NAMB_RELEASE, Save R2,R3                  ; 0823
                        53        04  AC  D0 00002        MOVL    ADDR, PTR                                           ; 0865
                        52 010A00F7   8F  D0 00006        MOVL    #17432823, R2                                       ; 0869
                        51      01E5  8F  3C 0000D        MOVZWL  #485, R1                                            ;
                        50            53  D0 00012        MOVL    PTR, R0                                             ;
                           00000000G  EF  16 00015        JSB     EXCH$UTIL_BLOCK_CHECK                               ;
                        50            63  0F 0001B        REMQUE  (PTR), T                                            ; 0873
            50 00000000G  EF 0000008C  8F  C1 0001E        ADDL3   #140, EXCH$A_GBL, R0                                ; 0877
                     04  B0            63  0E 0002A        INSQUE  (PTR), @4(R0)                                       ;
                                          04 0002E        RET                                                         ; 0880

; Routine Size:  47 bytes,    Routine Base:  EXCH$UTIL_CODE + 033C
```

```
0881  1 GLOBAL ROUTINE exch$util_radix50_from_ascii (asc_cnt, asc, r50_cnt, r50) =        %SBTTL 'exch$util_radix50_fr
0882  2 BEGIN
0883  2 !++
0884  2 !
0885  2 ! FUNCTIONAL DESCRIPTION:
0886  2 !
0887  2 !       This converts ascii strings to Radix-50.
0888  2 !
0889  2 ! INPUTS:
0890  2 !
0891  2 !       asc_cnt - count of ascii characters to output
0892  2 !       asc     - address of buffer of ascii characters
0893  2 !       r50_cnt - count of radix-50 characters
0894  2 !
0895  2 ! IMPLICIT INPUTS:
0896  2 !
0897  2 !       none
0898  2 !
0899  2 ! OUTPUTS:
0900  2 !
0901  2 !       r50     - address of Radix-50 string
0902  2 !
0903  2 ! IMPLICIT OUTPUTS:
0904  2 !
0905  2 !       none
0906  2 !
0907  2 ! ROUTINE VALUE:
0908  2 !
0909  2 !       true if conversion went smoothly, false if anything unusual
0910  2 !
0911  2 ! SIDE EFFECTS:
0912  2 !
0913  2 !       none
0914  2 !--
0915  2
0916  2 LOCAL
0917  2     buf : $bvector [6]
0918  2     ;
0919  2
0920  2 EXTERNAL ROUTINE irad50 : ADDRESSING_MODE (GENERAL);     ! F4P compatibility routine
0921  2
0922  2 $logic_check (2, (.asc_cnt LEQU 6), 165);
0923  2 CH$COPY (.asc_cnt, .asc, 32, 6, buf);
0924  2
0925  2 irad50 (r50_cnt, buf, .r50);
0926  2
0927  2 RETURN true;
0928  2
0929  1 END;


                                                        .EXTRN  IRAD50, EXCH$_BADLOGIC

                                  003C 00000            .ENTRY  EXCH$UTIL_RADIX50_FROM_ASCII, Save R2,R3,-   ; 0881
                                                                R4,R5                                        ;
                  5E           08  C2 00002            SUBL2   #8, SP                                       ;
```

```
                                06          04   AC  D1 00005          CMPL    ASC_CNT, #6                                    ; 0922
                                                 13  1B 00009          BLEQU   1$                                             ;
                                7E          A5   8F  9A 0000B          MOVZBL  #165, -(SP)                                    ;
                                                 01  DD 0000F          PUSHL   #1                                             ;
                                      00000000G  8F  DD 00011          PUSHL   #EXCH$_BADLOGIC                                ;
                       00000000G  00             03  FB 00017          CALLS   #3, LIB$STOP                                   ;
        06          20      08    BC        04   AC  2C 0001E  1$:     MOVC5   ASC_CNT, @ASC, #32, #6, BUF                    ; 0923
                                                 6E     00025                                                                 ;
                                            10   AC  DD 00026          PUSHL   R50                                            ; 0925
                                            04   AE  9F 00029          PUSHAB  BUF                                            ;
                                            0C   AC  9F 0002C          PUSHAB  R50_CNT                                        ;
                       00000000G  00             03  FB 0002F          CALLS   #3, IRAD50                                     ;
                                  50             01  D0 00036          MOVL    #1, R0                                         ; 0927
                                                     04 00039          RET                                                    ; 0929

; Routine Size:  58 bytes,    Routine Base:  EXCH$UTIL_CODE + 036B
;  847          0930   1 GLOBAL ROUTINE exch$util_radix50_to_ascii (asc_cnt, r50, asc) = %SBTTL 'exch$util_radix50_to_ascii (asc_cnt,
;  848          0931   2 BEGIN
;  849          0932   2 !++
;  850          0933   2 !
;  851          0934   2 ! FUNCTIONAL DESCRIPTION:
;  852          0935   2 !
;  853          0936   2 !       This converts Radix-50 strings to ascii.
;  854          0937   2 !
;  855          0938   2 ! INPUTS:
;  856          0939   2 !
;  857          0940   2 !       asc_cnt - count of ascii characters to output
;  858          0941   2 !       r50     - address of Radix-50 string.  Asc_cnt implies the length of this string.
;  859          0942   2 !
;  860          0943   2 ! IMPLICIT INPUTS:
;  861          0944   2 !
;  862          0945   2 !       none
;  863          0946   2 !
;  864          0947   2 ! OUTPUTS:
;  865          0948   2 !
;  866          0949   2 !       asc     - address of buffer to receive ascii characters
;  867          0950   2 !
;  868          0951   2 ! IMPLICIT OUTPUTS:
;  869          0952   2 !
;  870          0953   2 !       none
;  871          0954   2 !
;  872          0955   2 ! ROUTINE VALUE:
;  873          0956   2 !
;  874          0957   2 !       true if conversion went smoothly, false if anything unusual
;  875          0958   2 !
;  876          0959   2 ! SIDE EFFECTS:
;  877          0960   2 !
;  878          0961   2 !       none
;  879          0962   2 !--
;  880          0963   2
;  881          0964   2
;  882          0965   2 EXTERNAL ROUTINE r50asc : ADDRESSING_MODE (GENERAL);     ! F4P compatibility routine
;  883          0966   2 r50asc (asc_cnt, .r50, .asc);
;  884          0967   2
;  885          0968   2 RETURN true;
;  886          0969   2
;  887          0970   1 END;


                                                                .EXTRN   R50ASC

                                              0000 00000        .ENTRY   EXCH$UTIL_RADIX50_TO_ASCII, Save nothing     ; 0930
                          7E        08    AC  7D 00002          MOVQ     R50, -(SP)                                    ; 0966
                                    04    AC  9F 00006          PUSHAB   ASC_CNT                                       ;
                  00000000G  00           03  FB 00009          CALLS    #3, R50ASC                                    ;
                          50              01  D0 00010          MOVL     #1, R0                                        ; 0968
                                              04 00013          RET                                                    ; 0970

; Routine Size:  20 bytes,    Routine Base:  EXCH$UTIL_CODE + 03A5
```

```
GLOBAL ROUTINE exch$util_rmsb_allocate =        %SBTTL 'exch$util_rmsb_allocate'
BEGIN
!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine allocates one $RMSB.  If $RMSBs are available, one is moved from the available queue to
!       in-use queue.  If none are available, then a fresh $RMSB is created and placed on the in-use queue.
!
! INPUTS:
!
!       none
!
! IMPLICIT INPUTS:
!
!       exch$a_gbl [excg$q_rmsb_all] - list of allocated file blocks
!       exch$a_gbl [excg$q_rmsb_avl] - queue of available file blocks
!       exch$a_gbl [excg$q_rmsb_use] - queue of file blocks in use
!
! OUTPUTS:
!
!       none
!
! IMPLICIT OUTPUTS:
!
!       none
!
! ROUTINE VALUE:
!
!       address of the allocated file block
!
! SIDE EFFECTS:
!
!       All errors are fatal
!--

LOCAL
    offset,                                         ! Local temporary
    ptr         : $ref_bblock,                      ! A local pointer to the rmsb
    status
    ;


! First, try to find one in the available queue
!
ptr = $queue_remove_head (exch$a_gbl [excg$q_rmsb_avl]);

! If we didn't find one, then it will have to be created
!
IF .ptr EQL 0
THEN
    BEGIN

    ! Allocate a fresh rmsb from virtual memory.  The entire block has been cleared to nulls
    !
    ptr = exch$util_vm_allocate_zeroed (exchblk$s_rmsb);

```

```
;   946        1028  3        ! Place the rmsb at the head of the list of allocated blocks
;   947        1029  3        !
;   948        1030  3        ptr [rmsb$a_alloc] = .exch$a_gbl [excg$a_rmsb_alloc];
;   949        1031  3        exch$a_gbl [excg$a_rmsb_alloc] = .ptr;
;   950        1032  3
;   951        1033  3        ! Set the block identification fields
;   952        1034  3        !
;   953        1035  3        $block_init (.ptr, rmsb);
;   954        1036  3
;   955        1037  3        ! Several items are located at the end of the $RMSB, fill in the pointers
;   956        1038  3        !
;   957        1039  3        ptr [rmsb$a_fab]    = .ptr + rmsb$k_length;                      ! Fab is at end of block
;   958        1040  3        ptr [rmsb$a_rab]    = .ptr [rmsb$a_fab] + fab$k_bln;             ! Rab right after Fab
;   959        1041  3        ptr [rmsb$a_nam]    = .ptr [rmsb$a_rab] + rab$k_bln;             ! Nam after Rab
;   960        1042  3        ptr [rmsb$a_esbuf] = .ptr [rmsb$a_nam] + nam$k_bln;              ! Expanded string after Nam
;   961        1043  3        ptr [rmsb$a_rsbuf] = .ptr [rmsb$a_esbuf] + nam$c_maxrss;         ! Result string after Ebuf
;   962        1044  3
;   963        1045  2        END;
;   964        1046  2
;   965        1047  2   ! Check our block type, fatal error if any problems
;   966        1048  2   !
;   967        1049  2   $block_check (2, .ptr, rmsb, 407);
;   968        1050  2
;   969        1051  2   ! Set the last part of the block to nulls
;   970        1052  2   !
;   971        1053  2   CH$FILL (0, exchblk$s_rmsb - rmsb$k_start_zero, .ptr + rmsb$k_start_zero);
;   972        1054  2
;   973        1055  2   ! Insert the block at the head of the in-use queue
;   974        1056  2   !
;   975        1057  2   $queue_insert_head (ptr [rmsb$q_header], exch$a_gbl [excg$q_rmsb_use]);
;   976        1058  2
;   977        1059  2   ! Return the address of the file block to the caller
;   978        1060  2   !
;   979        1061  2   RETURN .ptr;
;   980        1062  2
;   981        1063  1   END;



                                       00FC 00000           .ENTRY  EXCH$UTIL_RMSB_ALLOCATE, Save R2,R3,R4,R5,-  ; 0971
                                                                    R6,R7                                        ;
                57 00000000G    EF   9E 00002           MOVAB   EXCH$A_GBL, R7                               ;
      51        67 000000A0     8F   C1 00009           ADDL3   #160, EXCH$A_GBL, R1                         ; 1016
                50        00    B1   0F 00011           REMQUE  @0(R1), _T_                                  ;
                                04   1C 00015           BVC     1$                                           ;
                                56   D4 00017           CLRL    PTR                                          ;
                                03   11 00019           BRB     2$                                           ;
                56              50   D0 0001B 1$:       MOVL    _T_, PTR                                     ;
                                52   12 0001E 2$:       BNEQ    3$                                           ; 1020
                7E      0316    8F   3C 00020           MOVZWL  #790, -(SP)                                  ; 1026
          0000V CF              01   FB 00025           CALLS   #1, EXCH$UTIL_VM_ALLOCATE_ZEROED             ;
                56              50   D0 0002A           MOVL    R0, PTR                                      ;
                50              67   D0 0002D           MOVL    EXCH$A_GBL, R0                               ; 1030
             0C A6      0094    C0   D0 00030           MOVL    148(R0), 12(PTR)                             ;
           0094 C0              56   D0 00036           MOVL    PTR, 148(R0)                                 ; 1031
```

```
                               08   A6       0316    8F  B0 0003B          MOVW     #790, 8(PTR)                                ; 1035
                               0A   A6               0A  8E 00041          MNEGB    #10, 10(PTR)                                ;
                               10   A6         24    A6  9E 00045          MOVAB    36(R6), 16(PTR)                             ; 1039
                14     A6      10   A6 00000050      8F  C1 0004A          ADDL3    #80, 16(PTR), 20(PTR)                       ; 1040
                18     A6      14   A6 00000044      8F  C1 00054          ADDL3    #68, 20(PTR), 24(PTR)                       ; 1041
                1C     A6      18   A6 00000060      8F  C1 0005E          ADDL3    #96, 24(PTR), 28(PTR)                       ; 1042
                20     A6      1C   A6 000000FF      8F  C1 00068          ADDL3    #255, 28(PTR), 32(PTR)                      ; 1043
                               52 031600F6           8F  D0 00072  3$:     MOVL     #51773686, R2                               ; 1049
                               51        0197        8F  3C 00079          MOVZWL   #407, R1                                    ;
                               50                    56  D0 0007E          MOVL     PTR, R0                                     ;
                                    00000000G        EF  16 00081          JSB      EXCH$UTIL_BLOCK_CHECK                       ;
     02F2   8F            00        6E               00  2C 00087          MOVC5    #0, (SP), #0, #754, 36(PTR)                 ; 1053
                                               24    A6     0008E                                                               ;
                          50        67 00000098      8F  C1 00090          ADDL3    #152, EXCH$A_GBL, R0                        ; 1057
                                    60               66  0E 00098          INSQUE   (PTR), (R0)                                 ;
                                    50               56  D0 0009B          MOVL     PTR, R0                                     ; 1061
                                                     04     0009E          RET                                                  ; 1063

; Routine Size:  159 bytes,     Routine Base:  EXCH$UTIL_CODE + 03B9
;  983     1 GLOBAL ROUTINE exch$util_rmsb_release (addr) : NOVALUE =          %SBTTL 'exch$util_rmsb_release (addr)'
;  984     2 BEGIN
;  985     2 !++
;  986     2 !
;  987     2 ! FUNCTIONAL DESCRIPTION:
;  988     2 !
;  989     2 !       This routine deallocates one $RMSB.  The $RMSB is moved from the in-use queue to the available queue
;  990     2 !
;  991     2 ! INPUTS:
;  992     2 !
;  993     2 !       addr - address of the block to release
;  994     2 !
;  995     2 ! IMPLICIT INPUTS:
;  996     2 !
;  997     2 !       exch$a_gbl [excg$q_rmsb_avl] - queue of available file blocks
;  998     2 !       exch$a_gbl [excg$q_rmsb_use] - queue of file blocks in use
;  999     2 !
; 1000     2 ! OUTPUTS:
; 1001     2 !
; 1002     2 !       none
; 1003     2 !
; 1004     2 ! IMPLICIT OUTPUTS:
; 1005     2 !
; 1006     2 !       none
; 1007     2 !
; 1008     2 ! ROUTINE VALUE:
; 1009     2 !
; 1010     2 !       none
; 1011     2 !
; 1012     2 ! SIDE EFFECTS:
; 1013     2 !
; 1014     2 !       All errors are fatal
; 1015     2 !--
; 1016     2
; 1017     2 LOCAL
; 1018     2     ptr          : $ref_bblock,                  ! A local pointer to the rmsb
; 1019     2     status
; 1020     2     ;
; 1021     2
; 1022     2
; 1023     2 ! First, move the pointer to a local variable
; 1024     2 !
; 1025     2 ptr = .addr;
; 1026     2
; 1027     2 ! Check our block type, fatal error if any problems
; 1028     2 !
; 1029     2 $block_check (2, .ptr, rmsb, 519);
; 1030     2
; 1031     2 ! Remove the rmsb from where ever it is in the in-use queue
; 1032     2 !
; 1033     2 $queue_remove (ptr [rmsb$q_header]);
; 1034     2
; 1035     2 ! Place the rmsb at the end of the available queue and the head of the in-use queue
; 1036     2 !
; 1037     2 $queue_insert_tail (ptr [rmsb$q_header], exch$a_gbl [excg$q_rmsb_avl]);
; 1038     2
; 1039     2 RETURN;
; 1040         1121  1 END;



                                             000C 00000          .ENTRY  EXCH$UTIL_RMSB_RELEASE, Save R2,R3                 ; 1064
                         53        04    AC   D0 00002          MOVL    ADDR, PTR                                          ; 1106
                         52 031600F6    8F   D0 00006          MOVL    #51773686, R2                                      ; 1110
                         51      0207    8F   3C 0000D          MOVZWL  #519, R1                                           ;
                         50               53   D0 00012          MOVL    PTR, R0                                            ;
                            00000000G    EF   16 00015          JSB     EXCH$UTIL_BLOCK_CHECK                              ;
                         50               63   0F 0001B          REMQUE  (PTR), T                                           ; 1114
        50 00000000G  EF 000000A0    8F   C1 0001E          ADDL3   #160, EXCH$A_GBL, R0                               ; 1118
                 04   B0               63   0E 0002A          INSQUE  (PTR), @4(R0)                                      ;
                                             04 0002E          RET                                                        ; 1121

; Routine Size:  47 bytes,    Routine Base:  EXCH$UTIL_CODE + 0458
```

```
1 GLOBAL ROUTINE exch$util_rt11ctx_allocate (volb, filb) =          %SBTTL 'exch$util_rt11ctx_allocate (volb, fi
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       This routine allocates one RT-11 file context block.  If one is available, it is moved from the avai
2 !       queue to the in-use queue.  If none are available, then a fresh block is created and placed on the i
2 !       queue.
2 !
2 ! INPUTS:
2 !
2 !       volb - pointer to the associated volb
2 !       filb - pointer to the associated filb
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       exch$a_gbl [excg$q_rt11ctx_all] - list of allocated file blocks
2 !       exch$a_gbl [excg$q_rt11ctx_avl] - queue of available file blocks
2 !       exch$a_gbl [excg$q_rt11ctx_use] - queue of file blocks in use
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       address of the allocated file block
2 !
2 ! SIDE EFFECTS:
2 !
2 !       All errors are fatal
2 !--
2
2 LOCAL
2     offset,                                           ! Local temporary
2     ptr           : $ref_bblock,                      ! A local pointer to the rt11ctx
2     status
2     ;
2
2
2 ! First, try to find one in the available queue
2 !
2 ptr = $queue_remove_head (exch$a_gbl [excg$q_rt11ctx_avl]);
2
2 ! If we didn't find one, then it will have to be created
2 !
2 IF .ptr EQL 0
2 THEN
3     BEGIN
3
3     ! Allocate a fresh rt11ctx from virtual memory.  The entire block has been cleared to nulls
3     !
```

```
; 1099       1179 3      ptr = exch$util_vm_allocate_zeroed (exchblk$s_rt11ctx);
; 1100       1180 3
; 1101       1181 3      ! Place the rt11ctx at the head of the list of allocated blocks
; 1102       1182 3      !
; 1103       1183 3      ptr [rt11ctx$a_alloc] = .exch$a_gbl [excg$a_rt11ctx_alloc];
; 1104       1184 3      exch$a_gbl [excg$a_rt11ctx_alloc] = .ptr;
; 1105       1185 3
; 1106       1186 3      ! Set the block identification fields
; 1107       1187 3      !
; 1108       1188 3      $block_init (.ptr, rt11ctx);
; 1109       1189 3
; 1110       1190 2      END;
; 1111       1191 2
; 1112       1192 2  ! Check our block type, fatal error if any problems
; 1113       1193 2  !
; 1114       1194 2  $block_check (2, .ptr, rt11ctx, 486);
; 1115       1195 2
; 1116       1196 2  ! Set the last part of the block to nulls
; 1117       1197 2  !
; 1118       1198 2  CH$FILL (0, rt11ctx$k_end_zero - rt11ctx$k_start_zero, .ptr + rt11ctx$k_start_zero);
; 1119       1199 2
; 1120       1200 2  ! Insert the block at the head of the in-use queue
; 1121       1201 2  !
; 1122       1202 2  $queue_insert_head (ptr [rt11ctx$q_header], exch$a_gbl [excg$q_rt11ctx_use]);
; 1123       1203 2
; 1124       1204 2  ! Set the two associated fields
; 1125       1205 2  !
; 1126       1206 2  ptr [rt11ctx$a_assoc_volb] = .volb;
; 1127       1207 2  ptr [rt11ctx$a_assoc_filb] = .filb;
; 1128       1208 2
; 1129       1209 2  ! Return the address of the file block to the caller
; 1130       1210 2  !
; 1131       1211 2  RETURN .ptr;
; 1132       1212 2
; 1133       1213 1  END;



                                         00FC 00000          .ENTRY  EXCH$UTIL_RT11CTX_ALLOCATE, Save R2,R3,R4,- ; 1122
                                                                     R5,R6,R7                                    ;
                      57 00000000G   EF   9E 00002           MOVAB   EXCH$A_GBL, R7                              ;
         51           67 000000B4    8F   C1 00009           ADDL3   #180, EXCH$A_GBL, R1                        ; 1169
                      50       00    B1   0F 00011           REMQUE  @0(R1), _T_                                 ;
                                     04   1C 00015           BVC     1$                                          ;
                                     56   D4 00017           CLRL    PTR                                         ;
                                     03   11 00019           BRB     2$                                          ;
                      56             50   D0 0001B 1$:       MOVL    _T_, PTR                                    ;
                                     23   12 0001E 2$:       BNEQ    3$                                          ; 1173
                      7E       82    8F   9A 00020           MOVZBL  #130, -(SP)                                 ; 1179
               0000V  CF             01   FB 00024           CALLS   #1, EXCH$UTIL_VM_ALLOCATE_ZEROED            ;
                      56             50   D0 00029           MOVL    R0, PTR                                     ;
                      50             67   D0 0002C           MOVL    EXCH$A_GBL, R0                              ; 1183
                  0C  A6     00A8    C0   D0 0002F           MOVL    168(R0), 12(PTR)                            ;
                00A8  C0             56   D0 00035           MOVL    PTR, 168(R0)                                ; 1184
                  08  A6       82    8F   9B 0003A           MOVZBW  #130, 8(PTR)                                ; 1188
```

```
                        0A   A6              0C  8E 0003F         MNEGB   #12, 10(PTR)                          ;
                             52 008200F4     8F  D0 00043 3$:     MOVL    #8519924, R2                          ; 1194
                             51     01E6     8F  3C 0004A         MOVZWL  #486, R1                              ;
                             50              56  D0 0004F         MOVL    PTR, R0                               ;
                                0000000G     EF  16 00052         JSB     EXCH$UTIL_BLOCK_CHECK                 ;
     0066   8F          00   6E              00  2C 00058         MOVC5   #0, (SP), #0, #T02, 28(PTR)           ; 1198
                                      1C     A6     0005F                                                       ;
                        50   67 000000AC     8F  C1 00061         ADDL3   #172, EXCH$A_GBL, R0                  ; 1202
                             60              66  0E 00069         INSQUE  (PTR), (R0)                           ;
                        14   A6       04     AC  D0 0006C         MOVL    VOLB, 20(PTR)                         ; 1206
                        10   A6       08     AC  D0 00071         MOVL    FILB, 16(PTR)                         ; 1207
                             50              56  D0 00076         MOVL    PTR, R0                               ; 1211
                                                 04 00079         RET                                           ; 1213

; Routine Size:  122 bytes,    Routine Base:  EXCH$UTIL_CODE + 0487
```

```
1 GLOBAL ROUTINE exch$util_rt11ctx_release (addr) : NOVALUE =      %SBTTL 'exch$util_rt11ctx_release (addr)'
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       This routine deallocates one rt11ctx.  The block is moved from the in-use queue to the available que
2 !
2 ! INPUTS:
2 !
2 !       addr - address of the block to release
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       exch$a_gbl [excg$q_rt11ctx_avl] - queue of available file blocks
2 !       exch$a_gbl [excg$q_rt11ctx_use] - queue of file blocks in use
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       none
2 !
2 ! SIDE EFFECTS:
2 !
2 !       All errors are fatal
2 !--
2
2 LOCAL
2     ptr         : $ref_bblock,                     ! A local pointer to the rt11ctx
2     status
2     ;
2
2 ! First, move the pointer to a local variable
2 !
2 ptr = .addr;
2
2 ! Check our block type, fatal error if any problems
2 !
2 $block_check (2, .ptr, rt11ctx, 487);
2
2 ! If there is a buffer allocated, free it
2 !
2 IF .ptr [rt11ctx$a_buffer] NEQ 0
2 THEN
2     exch$util_vm_release (ctx$k_buffer_length, .ptr [rt11ctx$a_buffer]);
2
2 ! Clear the pointers in the part of the block before the automatic zero
2 !
2 ptr [rt11ctx$a_assoc_filb] = 0;
```

```
; 1192          1271  2 ptr [rt11ctx$a_assoc_volb] = 0;
; 1193          1272  2 ptr [rt11ctx$a_buffer]     = 0;
; 1194          1273  2
; 1195          1274  2 ! Remove the rt11ctx from where ever it is in the in-use queue
; 1196          1275  2 !
; 1197          1276  2 $queue_remove (ptr [rt11ctx$q_header]);
; 1198          1277  2
; 1199          1278  2 ! Place the rt11ctx at the end of the available queue and the head of the in-use queue
; 1200          1279  2 !
; 1201          1280  2 $queue_insert_tail (ptr [rt11ctx$q_header], exch$a_gbl [excg$q_rt11ctx_avl]);
; 1202          1281  2
; 1203          1282  2 RETURN;
; 1204          1283  1 END;



                                             000C 00000          .ENTRY  EXCH$UTIL_RT11CTX_RELEASE, Save R2,R3         ; 1214
                        53          04    AC   D0 00002          MOVL    ADDR, PTR                                      ; 1256
                        52  008200F4    8F   D0 00006          MOVL    #8519924, R2                                   ; 1260
                        51        01E7    8F   3C 0000D          MOVZWL  #487, R1                                       ;
                        50                     53 D0 00012       MOVL    PTR, R0                                        ;
                           00000000G    EF   16 00015          JSB     EXCH$UTIL_BLOCK_CHECK                          ;
                                  18    A3   D5 0001B          TSTL    24(PTR)                                        ; 1264
                                        0D   13 0001E          BEQL    1$                                             ;
                                  18    A3   DD 00020          PUSHL   24(PTR)                                        ; 1266
                        7E      1800    8F   3C 00023          MOVZWL  #6144, -(SP)                                   ;
                0000V   CF              02   FB 00028          CALLS   #2, EXCH$UTIL_VM_RELEASE                       ;
                                  10    A3   7C 0002D 1$:      CLRQ    16(PTR)                                        ; 1270
                                  18    A3   D4 00030          CLRL    24(PTR)                                        ; 1272
                        50                     63 0F 00033     REMQUE  (PTR), T                                       ; 1276
             50 00000000G  EF  000000B4    8F   C1 00036       ADDL3   #180, EXCH$A_GBL, R0                           ; 1280
                        04  B0                 63 0E 00042     INSQUE  (PTR), @4(R0)                                  ;
                                               04 00046        RET                                                    ; 1283

; Routine Size:  71 bytes,    Routine Base:  EXCH$UTIL_CODE + 0501
; 1206      1284  1 GLOBAL ROUTINE exch$util_vm_allocate (size) =    %SBTTL 'exch$util_vm_allocate (size)'
; 1207      1285  2 BEGIN
; 1208      1286  2 !++
; 1209      1287  2 !
; 1210      1288  2 ! FUNCTIONAL DESCRIPTION:
; 1211      1289  2 !
; 1212      1290  2 !       This routine calls the LIB$GET_VM service to allocate dynamic memory.
; 1213      1291  2 !
; 1214      1292  2 ! INPUTS:
; 1215      1293  2 !
; 1216      1294  2 !       size    Number of bytes to allocate (by value)
; 1217      1295  2 !
; 1218      1296  2 ! IMPLICIT INPUTS:
; 1219      1297  2 !
; 1220      1298  2 !       none
; 1221      1299  2 !
; 1222      1300  2 ! OUTPUTS:
; 1223      1301  2 !
; 1224      1302  2 !       none
; 1225      1303  2 !
; 1226      1304  2 ! IMPLICIT OUTPUTS:
; 1227      1305  2 !
; 1228      1306  2 !       none
; 1229      1307  2 !
; 1230      1308  2 ! ROUTINE VALUE:
; 1231      1309  2 !
; 1232      1310  2 !       address of the allocated memory
; 1233      1311  2 !
; 1234      1312  2 ! SIDE EFFECTS:
; 1235      1313  2 !
; 1236      1314  2 !       All errors are fatal
; 1237      1315  2 !--
; 1238      1316  2
; 1239      1317  2 LOCAL
; 1240      1318  2     addr,
; 1241      1319  2     status
; 1242      1320  2     ;
; 1243      1321  2
; 1244      1322  3 IF NOT (status = lib$get_vm (size, addr))        ! Pass the call through
; 1245      1323  2 THEN
; 1246      1324  2     $exch_signal_stop (.status);
; 1247      1325  2
; 1248      1326  2 RETURN .addr;
; 1249      1327  1 END;


                                                                  .EXTRN  LIB$GET_VM

                                                 0000 00000       .ENTRY  EXCH$UTIL_VM_ALLOCATE, Save nothing       ; 1284
                     5E                       04  C2 00002        SUBL2   #4, SP
                                              5E  DD 00005        PUSHL   SP                                        ; 1322
                                       04     AC  9F 00007        PUSHAB  SIZE                                      ;
               00000000G  00                  02  FB 0000A        CALLS   #2, LIB$GET_VM                            ;
                          0A                  50  E8 00011        BLBS    STATUS, 1$                                ;
                                              50  DD 00014        PUSHL   STATUS                                    ; 1324
               00000000G  00                  01  FB 00016        CALLS   #1, LIB$STOP                              ;
```

```
                                              M 14
EXCH$UTIL        Facility-wide misc routines  16-Sep-1984 01:25:39    VAX-11 Bliss-32 V4.0-742          Page  40
V04-000          exch$util_vm_allocate (size) 14-Sep-1984 12:29:09    [EXCHNG.SRC]EXCUTIL.B32;1              (19)

                                          04 0001D       RET                                          ;
                          50          6E  D0 0001E 1$:   MOVL    ADDR, R0                             ; 1326
                                          04 00021       RET                                          ; 1327

; Routine Size:  34 bytes,    Routine Base:  EXCH$UTIL_CODE + 0548
```

```
GLOBAL ROUTINE exch$util_vm_allocate_zeroed (size) =     %SBTTL 'exch$util_vm_allocate_zeroed (size)'
BEGIN
!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine allocates dynamic memory.  The memory contents are set to nulls.
!
! INPUTS:
!
!       size    Number of bytes to allocate (by value)
!
! IMPLICIT INPUTS:
!
!       none
!
! OUTPUTS:
!
!       none
!
! IMPLICIT OUTPUTS:
!
!       none
!
! ROUTINE VALUE:
!
!       address of the allocated memory
!
! SIDE EFFECTS:
!
!       All errors are fatal
!--

REGISTER
    addr,                                              ! address of new memory
    chunk : INITIAL (65535),                           ! used to force a large constant into a register
    tmp_adr,                                           ! temp pointer and size
    tmp_siz
    ;


! Allocate the memory
!
addr = exch$util_vm_allocate (.size);

! Zap the entire piece of memory to nulls.  Since the VAX architecture only supports short strings, we must
! it into 64K chunks
!
tmp_adr = .addr;
tmp_siz = .size;
WHILE .tmp_siz GTRU .chunk
DO
    BEGIN
    CH$FILL (0, .chunk, .tmp_adr);
    tmp_adr = .tmp_adr + .chunk;
    tmp_siz = .tmp_siz - .chunk;
    END;
```

```
; 1308          1385  2
; 1309          1386  2  ! Do the last (usually only) piece of memory
; 1310          1387  2  !
; 1311          1388  2  CH$FILL (0, .tmp_siz, .tmp_adr);
; 1312          1389  2
; 1313          1390  2  RETURN .addr;
; 1314          1391  1  END;




                                              03FC 00000           .ENTRY  EXCH$UTIL_VM_ALLOCATE_ZEROED, Save R2,R3,-   ; 1328
                                                                           R4,R5,R6,R7,R8,R9                            ;
                          57      FFFF     8F   3C 00002           MOVZWL  #65535, CHUNK                                ; 1329
                                    04     AC   DD 00007           PUSHL   SIZE                                         ; 1371
                    D0    AF               01   FB 0000A           CALLS   #1, EXCH$UTIL_VM_ALLOCATE                    ;
                          56               50   D0 0000E           MOVL    R0, ADDR                                     ;
                          58               56   D0 00011           MOVL    ADDR, TMP_ADR                                ; 1376
                          59        04     AC   D0 00014           MOVL    SIZE, TMP_SIZ                                ; 1377
                          57               59   D1 00018 1$:       CMPL    TMP_SIZ, CHUNK                               ; 1378
                                           0E   1B 0001B           BLEQU   2$                                           ;
      57         00       6E               00   2C 0001D           MOVC5   #0, (SP), #0, CHUNK, (TMP_ADR)               ; 1381
                                           68      00022                                                                ;
                          58               57   C0 00023           ADDL2   CHUNK, TMP_ADR                               ; 1382
                          59               57   C2 00026           SUBL2   CHUNK, TMP_SIZ                               ; 1383
                                           ED   11 00029           BRB     1$                                           ; 1378
      59         00       6E               00   2C 0002B 2$:       MOVC5   #0, (SP), #0, TMP_SIZ, (TMP_ADR)             ; 1388
                                           68      00030                                                                ;
                          50               56   D0 00031           MOVL    ADDR, R0                                     ; 1390
                                                04 00034           RET                                                  ; 1391

; Routine Size:  53 bytes,    Routine Base:  EXCH$UTIL_CODE + 056A
; 1316        1392  1 GLOBAL ROUTINE exch$util_vm_release (size, addr) : NOVALUE =     %SBTTL 'exch$util_vm_release (size, addr)'
; 1317        1393  2 BEGIN
; 1318        1394  2 !++
; 1319        1395  2 !
; 1320        1396  2 ! FUNCTIONAL DESCRIPTION:
; 1321        1397  2 !
; 1322        1398  2 !       This routine calls the LIB$FREE_VM service to release dynamic memory.
; 1323        1399  2 !
; 1324        1400  2 ! INPUTS:
; 1325        1401  2 !
; 1326        1402  2 !       size    Number of bytes to release (by value)
; 1327        1403  2 !       addr    Address of longword containing address of memory to release
; 1328        1404  2 !
; 1329        1405  2 ! IMPLICIT INPUTS:
; 1330        1406  2 !
; 1331        1407  2 !       none
; 1332        1408  2 !
; 1333        1409  2 ! OUTPUTS:
; 1334        1410  2 !
; 1335        1411  2 !       none
; 1336        1412  2 !
; 1337        1413  2 ! IMPLICIT OUTPUTS:
; 1338        1414  2 !
; 1339        1415  2 !       none
; 1340        1416  2 !
; 1341        1417  2 ! ROUTINE VALUE:
; 1342        1418  2 !
; 1343        1419  2 !       Success, or status code of error converted to warning severity
; 1344        1420  2 !
; 1345        1421  2 ! SIDE EFFECTS:
; 1346        1422  2 !
; 1347        1423  2 !       Errors are signalled
; 1348        1424  2 !--
; 1349        1425  2
; 1350        1426  2 LOCAL
; 1351        1427  2     status
; 1352        1428  2     ;
; 1353        1429  2
; 1354        1430  3 IF NOT (status = lib$free_vm (size, addr))       ! Pass the call through, no dots tho
; 1355        1431  2 THEN
; 1356        1432  2     $exch_signal_stop (.status);
; 1357        1433  2
; 1358        1434  2 RETURN;
; 1359        1435  1 END;


                                                               .EXTRN   LIB$FREE_VM

                                         0000 00000            .ENTRY   EXCH$UTIL_VM_RELEASE, Save nothing          ; 1392
                             08    AC    9F 00002              PUSHAB   ADDR                                        ; 1430
                             04    AC    9F 00005              PUSHAB   SIZE                                        ;
             00000000G  00         02    FB 00008              CALLS    #2, LIB$FREE_VM                             ;
                        09         50    E8 0000F              BLBS     STATUS, 1$                                  ;
                                   50    DD 00012              PUSHL    STATUS                                      ; 1432
             00000000G  00         01    FB 00014              CALLS    #1, LIB$STOP                                ;
                                         04 0001B 1$:          RET                                                  ; 1435
```

```
; Routine Size:  28 bytes,    Routine Base:  EXCH$UTIL_CODE + 059F
```

```
1 GLOBAL ROUTINE exch$util_vol_getdvi (devname : REF $desc_block,            %SBTTL 'exch$util_vol_getdvi (devnam
1                                      volb : $ref_bblock) =
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       Get standard device information for a volb
2 !
2 ! INPUTS:
2 !
2 !       devname - address of descriptor for device name
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       none
2 !
2 ! OUTPUTS:
2 !
2 !       volb - several characteristics fields in the volb are filled in
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       Success or worst error encountered.
2 !
2 ! SIDE EFFECTS:
2 !
2 !       none
2 !--
2
2 $dbgtrc_prefix ('util_vol_getdvi> ');
2
2 LOCAL
2     status,
2     dev_item : VECTOR [22, LONG]
2     ;
2
2 $block_check (2, .volb, volb, 488);
2
2 ! Initialize the item list for the $GETDVI
2 !
2 dev_item  [0] = (dvi$_devbufsiz^16 OR 4);      ! Device buffer size, output length 4
2 dev_item  [1] = volb [volb$l_devbufsiz];       ! Address of output buffer
2 dev_item  [2] = 0;                             ! No returned length
2 dev_item  [3] = (dvi$_devchar^16 OR 4);
2 dev_item  [4] = volb [volb$l_devchar];
2 dev_item  [5] = 0;
2 dev_item  [6] = (dvi$_devclass^16 OR 4);
2 dev_item  [7] = volb [volb$l_devclass];
2 dev_item  [8] = 0;
2 dev_item  [9] = (dvi$_devdepend^16 OR 4);
2 dev_item [10] = volb [volb$l_devdepend];
2 dev_item [11] = 0;
```

```
; 1418          1493  2 dev_item [12] = (dvi$_fulldevnam^16 OR 16);
; 1419          1494  2 dev_item [13] = volb [volb$t_devnam];
; 1420          1495  2 dev_item [14] = volb [volb$l_devnamlen];
; 1421          1496  2 dev_item [15] = (dvi$_devtype^16 OR 4);
; 1422          1497  2 dev_item [16] = volb [volb$l_devtype];
; 1423          1498  2 dev_item [17] = 0;
; 1424          1499  2 dev_item [18] = (dvi$_maxblock^16 OR 4);
; 1425          1500  2 dev_item [19] = volb [volb$l_devmaxblock];
; 1426          1501  2 dev_item [20] = 0;
; 1427          1502  2 dev_item [21] = 0;                                  ! End of GETDVI item list
; 1428          1503  2
; 1429          1504  2 ! Get the device information
; 1430          1505  2 !
; 1431          1506  3 IF NOT (status = $getdviw (efn=0, devnam=.devname, itmlst=dev_item))
; 1432          1507  2 THEN
; 1433          1508  2     RETURN .status;
; 1434          1509  2
; 1435          1510  2 ! Do any manipulations necessary with the raw device info
; 1436          1511  2 !
; 1437          1512  2 volb [volb$l_volmaxblock] = .volb [volb$l_devmaxblock]; ! Assume device and volume same size
; 1438          1513  2
; 1439          1514  2 ! Debugging trace code
; 1440          1515  2 !
; 1441        L 1516  2 %IF switch_trace
; 1442        U 1517  2 %THEN
; 1443        U 1518  2     BEGIN
; 1444        U 1519  2     EXTERNAL ROUTINE
; 1445        U 1520  2         exch$dbg_utl_print_devchar;
; 1446        U 1521  2     LOCAL
; 1447        U 1522  2         tmp_desc : $desc_block;
; 1448        U 1523  2     BIND
; 1449        U 1524  2         dep = volb [volb$l_devdepend] : $bblock;
; 1450        U 1525  2
; 1451        U 1526  2     $stat_str_desc_init (tmp_desc, .volb [volb$l_devnamlen], volb [volb$t_devnam]);
; 1452        U 1527  2     $trace_print_fao ('Getdvi for name "!AS" resolved to device "!AS"', .devname, tmp_desc);
; 1453        U 1528  2     $trace_print_fao ('Bufsiz = !UL,  Maxblocks = !UL,  Class = !XB,  Type = !XB', .volb [volb$l_devbufsiz],
; 1454        U 1529  2             .volb [volb$l_devmaxblock], .volb [volb$l_devclass], .volb [volb$l_devtype]);
; 1455        U 1530  2     $trace_print_fao ('Cylinders = !UL, Tracks = !UL, Sectors = !UL, DevChar = !XL',
; 1456        U 1531  2             .dep[0,16,16,0], .dep[0,8,8,0], .dep[0,0,8,0], .volb [volb$l_devchar]);
; 1457        U 1532  2     exch$dbg_utl_print_devchar  (.volb [volb$l_devchar]);
; 1458        U 1533  2     END;
; 1459          1534  2 %FI
; 1460          1535  2
; 1461          1536  2 RETURN .status;
; 1462          1537  1 END;


                                                              .EXTRN  SYS$GETDVIW

                                        000C 00000          .ENTRY  EXCH$UTIL_VOL_GETDVI, Save R2,R3                      ; 1436
                   5E          A8   AE   9E 00002          MOVAB   -88(SP), SP
                   53          08   AC   D0 00006          MOVL    VOLB, R3                                              ; 1477
                   52 041B00F3      8F   D0 0000A          MOVL    #68878579, R2                                         ;
                   51     01E8      8F   3C 00011          MOVZWL  #488, R1                                              ;
                   50               53   D0 00016          MOVL    R3, R0                                                ;
                      00000000G     EF   16 00019          JSB     EXCH$UTIL_BLOCK_CHECK                                 ;
```

```
                                                                G 15
EXCH$UTIL       Facility-wide misc routines                     16-Sep-1984 01:25:39    VAX-11 Bliss-32 V4.0-742          Page  47
V04-000         exch$util_vol_getdvi (devname, volb)            14-Sep-1984 12:29:09    [EXCHNG.SRC]EXCUTIL.B32;1              (22)

                6E 00080004     8F  D0 0001F            MOVL    #524292, DEV_ITEM                               ; 1481
        04      AE       28     A3  9E 00026            MOVAB   40(R3), DEV_ITEM+4                              ; 1482
                         08     AE  D4 0002B            CLRL    DEV_ITEM+8                                      ; 1483
        0C      AE 00020004     8F  D0 0002E            MOVL    #131076, DEV_ITEM+12                            ; 1484
        10      AE       2C     A3  9E 00036            MOVAB   44(R3), DEV_ITEM+16                             ; 1485
                         14     AE  D4 0003B            CLRL    DEV_ITEM+20                                     ; 1486
        18      AE 00040004     8F  D0 0003E            MOVL    #262148, DEV_ITEM+24                            ; 1487
        1C      AE       30     A3  9E 00046            MOVAB   48(R3), DEV_ITEM+28                             ; 1488
                         20     AE  D4 0004B            CLRL    DEV_ITEM+32                                     ; 1489
        24      AE 000A0004     8F  D0 0004E            MOVL    #655364, DEV_ITEM+36                            ; 1490
        28      AE       34     A3  9E 00056            MOVAB   52(R3), DEV_ITEM+40                             ; 1491
                         2C     AE  D4 0005B            CLRL    DEV_ITEM+44                                     ; 1492
        30      AE 00E80010     8F  D0 0005E            MOVL    #15204368, DEV_ITEM+48                          ; 1493
        34      AE     00E9     C3  9E 00066            MOVAB   233(R3), DEV_ITEM+52                            ; 1494
        38      AE       38     A3  9E 0006C            MOVAB   56(R3), DEV_ITEM+56                             ; 1495
        3C      AE 00060004     8F  D0 00071            MOVL    #393220, DEV_ITEM+60                            ; 1496
        40      AE       3C     A3  9E 00079            MOVAB   60(R3), DEV_ITEM+64                             ; 1497
                         44     AE  D4 0007E            CLRL    DEV_ITEM+68                                     ; 1498
        48      AE 001A0004     8F  D0 00081            MOVL    #1703940, DEV_ITEM+72                           ; 1499
        4C      AE       40     A3  9E 00089            MOVAB   64(R3), DEV_ITEM+76                             ; 1500
                         50     AE  7C 0008E            CLRQ    DEV_ITEM+80                                     ; 1501
                                7E  7C 00091            CLRQ    -(SP)                                           ; 1506
                                7E  7C 00093            CLRQ    -(SP)                                           ;
                         10     AE  9F 00095            PUSHAB  DEV_ITEM                                        ;
                         04     AC  DD 00098            PUSHL   DEVNAME                                         ;
                                7E  7C 0009B            CLRQ    -(SP)                                           ;
   00000000G    00               08  FB 0009D           CALLS   #8, SYS$GETDVIW                                 ;
                05               50  E9 000A4           BLBC    STATUS, 1$                                      ;
        44      A3       40     A3  D0 000A7            MOVL    64(R3), 68(R3)                                  ; 1512
                                    04 000AC 1$:        RET                                                     ; 1537

; Routine Size:  173 bytes,    Routine Base:  EXCH$UTIL_CODE + 05BB
```

```
EX

Mo
--
EX
DE
LO
DE
TR
EX
EX
MO
SN
IN
MA
SY
LI
SY
SY
```

```
GLOBAL ROUTINE exch$util_volb_allocate =        %SBTTL 'exch$util_volb_allocate'
BEGIN
!++
!
! FUNCTIONAL DESCRIPTION:
!
!       This routine allocates one $VOLB.  If $VOLBs are available, one is moved from the available queue to
!       in-use queue.  If none are available, then a fresh $VOLB is created and placed on the in-use queue.
!
! INPUTS:
!
!       none
!
! IMPLICIT INPUTS:
!
!       exch$a_gbl [excg$a_volb_alloc] - list of allocated volume blocks
!       exch$a_gbl [excg$q_volb_avl]   - queue of available volume blocks
!       exch$a_gbl [excg$q_volb_use]   - queue of volume blocks in use
!
! OUTPUTS:
!
!       none
!
! IMPLICIT OUTPUTS:
!
!       none
!
! ROUTINE VALUE:
!
!       address of the allocated volume block
!
! SIDE EFFECTS:
!
!       All errors are fatal
!--

LOCAL
    offset,                                     ! Local temporary
    ptr         : $ref_bblock,                  ! A local pointer to the volb
    status
    ;


! First, try to find one in the available queue
!
ptr = $queue_remove_head (exch$a_gbl [excg$q_volb_avl]);

! If we didn't find one, then it will have to be created
!
IF .ptr EQL 0
THEN
    BEGIN

    ! Allocate a fresh volb from virtual memory.  The entire block has been cleared to nulls
    !
    ptr = exch$util_vm_allocate_zeroed (exchblk$s_volb);

```

```
                                                            I 15
EXCH$UTIL          Facility-wide misc routines           16-Sep-1984 01:25:39    VAX-11 Bliss-32 V4.0-742              Page  49
V04-000            exch$util_volb_allocate               14-Sep-1984 12:29:09    [EXCHNG.SRC]EXCUTIL.B32;1                  (23)

; 1521      1595 3      ! Place the volb at the head of the list of allocated blocks
; 1522      1596 3      !
; 1523      1597 3      ptr [volb$a_alloc] = .exch$a_gbl [excg$a_volb_alloc];
; 1524      1598 3      exch$a_gbl [excg$a_volb_alloc] = .ptr;
; 1525      1599 3
; 1526      1600 3      ! Set the block identification fields
; 1527      1601 3      !
; 1528      1602 3      $block_init (.ptr, volb);
; 1529      1603 3
; 1530      1604 3      ! Several items are located at the end of the $VOLB, fill in the pointers
; 1531      1605 3      !
; 1532      1606 3      ptr [volb$a_fab]   = .ptr + volb$k_length;                  ! Fab is at end of block
; 1533      1607 3      ptr [volb$a_rab]   = .ptr [volb$a_fab] + fab$k_bln;         ! Rab right after Fab
; 1534      1608 3      ptr [volb$a_nam]   = .ptr [volb$a_rab] + rab$k_bln;         ! Nam after Rab
; 1535      1609 3      ptr [volb$a_esbuf] = .ptr [volb$a_nam] + nam$k_bln;         ! Expanded string after Nam
; 1536      1610 3      ptr [volb$a_rsbuf] = .ptr [volb$a_esbuf] + nam$c_maxrss;    ! Result string after Ebuf
; 1537      1611 3
; 1538      1612 2      END;
; 1539      1613 2
; 1540      1614 2  ! Check our block type, fatal error if any problems
; 1541      1615 2  !
; 1542      1616 2  $block_check (2, .ptr, volb, 489);
; 1543      1617 2
; 1544      1618 2  ! Set the last part of the block to nulls
; 1545      1619 2  !
; 1546      1620 2  CH$FILL (0, exchblk$s_volb - volb$k_start_zero, .ptr + volb$k_start_zero);
; 1547      1621 2
; 1548      1622 2  ! Place the volb at the head of the in-use queue
; 1549      1623 2  !
; 1550      1624 2  $queue_insert_head (ptr [volb$q_header], exch$a_gbl [excg$q_volb_use]);
; 1551      1625 2
; 1552      1626 2  ! Return the address of the volume block to the caller
; 1553      1627 2  !
; 1554      1628 2  RETURN .ptr;
; 1555      1629 2
; 1556      1630 1  END;


                                         00FC 00000           .ENTRY   EXCH$UTIL_VOLB_ALLOCATE, Save R2,R3,R4,R5,- ; 1538
                                                                       R6,R7                                        ;
                          57 00000000G   EF  9E 00002         MOVAB    EXCH$A_GBL, R7                               ;
          51              67 000000C8    8F  C1 00009         ADDL3    #200, EXCH$A_GBL, R1                         ; 1583
                          50       00    B1  0F 00011         REMQUE   @0(R1), _T_                                  ;
                                         04  1C 00015         BVC      1$                                           ;
                                         56  D4 00017         CLRL     PTR                                          ;
                                         03  11 00019         BRB      2$                                           ;
                          56             50  D0 0001B 1$:     MOVL     _T_, PTR                                     ;
                                         53  12 0001E 2$:     BNEQ     3$                                           ; 1587
                          7E     041B    8F  3C 00020         MOVZWL   #1051, -(SP)                                 ; 1593
                  FED8    CF             01  FB 00025         CALLS    #1, EXCH$UTIL_VM_ALLOCATE_ZEROED             ;
                          56             50  D0 0002A         MOVL     R0, PTR                                      ;
                          50             67  D0 0002D         MOVL     EXCH$A_GBL, R0                               ; 1597
                    0C    A6     00BC    C0  D0 00030         MOVL     188(R0), 12(PTR)                             ;
                  00BC    C0             56  D0 00036         MOVL     PTR, 188(R0)                                 ; 1598
```

```
                              08   A6      041B   8F  B0 0003B          MOVW     #1051, 8(PTR)                              ; 1602
                              0A   A6             0D  8E 00041          MNEGB    #13, 10(PTR)                               ;
                              10   A6      0129   C6  9E 00045          MOVAB    297(R6), 16(PTR)                           ; 1606
            14    A6          10   A6 00000050    8F  C1 0004B          ADDL3    #80, 16(PTR), 20(PTR)                      ; 1607
            18    A6          14   A6 00000044    8F  C1 00055          ADDL3    #68, 20(PTR), 24(PTR)                      ; 1608
            1C    A6          18   A6 00000060    8F  C1 0005F          ADDL3    #96, 24(PTR), 28(PTR)                      ; 1609
            20    A6          1C   A6 000000FF    8F  C1 00069          ADDL3    #255, 28(PTR), 32(PTR)                     ; 1610
                                   52 041B00F3    8F  D0 00073 3$:      MOVL     #68878579, R2                              ; 1616
                                   51     01E9    8F  3C 0007A          MOVZWL   #489, R1                                   ;
                                   50             56  D0 0007F          MOVL     PTR, R0                                    ;
                                      00000000G   EF  16 00082          JSB      EXCH$UTIL_BLOCK_CHECK                      ;
    03F3    8F            00       6E             00  2C 00088          MOVC5    #0, (SP), #0, #T011, 40(PTR)               ; 1620
                                            28    A6     0008F                                                              ;
                          50       67 000000C0    8F  C1 00091          ADDL3    #192, EXCH$A_GBL, R0                       ; 1624
                                   60             66  0E 00099          INSQUE   (PTR), (R0)                                ;
                                   50             56  D0 0009C          MOVL     PTR, R0                                    ; 1628
                                                  04     0009F          RET                                                 ; 1630

; Routine Size:  160 bytes,    Routine Base:  EXCH$UTIL_CODE + 0668
```

```
1 GLOBAL ROUTINE exch$util_volb_release (addr) : NOVALUE =          %SBTTL 'exch$util_volb_release (addr)'
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       This routine deallocates one $VOLB.  The $VOLBs is moved from the in-use queue to the available queu
2 !
2 ! INPUTS:
2 !
2 !       addr - address of the block to release
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       exch$a_gbl [excg$q_volb_avl] - queue of available volume blocks
2 !       exch$a_gbl [excg$q_volb_use] - queue of volume blocks in use
2 !
2 ! OUTPUTS:
2 !
2 !       none
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       none
2 !
2 ! SIDE EFFECTS:
2 !
2 !       All errors are fatal
2 !--
2
2 LOCAL
2     ptr         : $ref_bblock,                  ! A local pointer to the volb
2     spc         : $ref_bblock,                  ! Pointer to volume specific structure
2     status
2     ;
2
2 ! First, move the pointer to a local variable
2 !
2 ptr = .addr;
2
2 ! Check our block type, fatal error if any problems
2 !
2 $block_check (2, .ptr, volb, 490);
2
2 ! Perform some volume specific actions on the specific pointer
2 !
2 IF (spc = .ptr [volb$a_vfmt_specific]) NEQ 0
2 THEN
3     BEGIN
3     LOCAL
3         block_size
3         ;
```

```
; 1615          1688  3
; 1616          1689  3      CASE .ptr [volb$b_vol_format] FROM volb$k_vfmt_lobound TO volb$k_vfmt_hibound OF
; 1617          1690  3      SET
; 1618          1691  4          [volb$k_vfmt_dos11] :      BEGIN
; 1619          1692  4                                     LOCAL
; 1620          1693  4                                         ent : $ref_bblock;
; 1621          1694  4
; 1622          1695  4                                     ! Follow the chain of entries and release them
; 1623          1696  4                                     !
; 1624          1697  5                                     WHILE (ent = $queue_remove_head (spc [dos11$q_entry_header]))
; 1625          1698  4                                     DO
; 1626          1699  4                                         exch$util_vm_release (dos11ent$k_length, .ent);     ! Release the entry
; 1627          1700  4
; 1628          1701  4                                     block_size = exchblk$s_dos11;
; 1629          1702  3                                     END;
; 1630          1703  3
; 1631          1704  3          [volb$k_vfmt_rt11] :       block_size = exchblk$s_rt11;
; 1632          1705  3
; 1633          1706  3          [INRANGE, OUTRANGE] :      $logic_check (0, (false), 250);
; 1634          1707  3
; 1635          1708  3      TES;
; 1636          1709  3
; 1637          1710  3      exch$util_vm_release (.block_size, .spc);           ! Release the extension
; 1638          1711  2      END;
; 1639          1712  2
; 1640          1713  2  ! Remove the volb from where ever it is in the in-use queue
; 1641          1714  2  !
; 1642          1715  2  $queue_remove (ptr [volb$q_header]);
; 1643          1716  2
; 1644          1717  2  ! Place the volb at the end of the available queue
; 1645          1718  2  !
; 1646          1719  2  $queue_insert_tail (ptr [volb$q_header], exch$a_gbl [excg$q_volb_avl]);
; 1647          1720  2
; 1648          1721  2  RETURN;
; 1649          1722  1  END;




                                               001C 00000          .ENTRY  EXCH$UTIL_VOLB_RELEASE, Save R2,R3,R4          ; 1631
                        54         04     AC    D0 00002          MOVL    ADDR, PTR                                      ; 1674
                        52   041B00F3     8F    D0 00006          MOVL    #68878579, R2                                  ; 1678
                        51       01EA     8F    3C 0000D          MOVZWL  #490, R1                                       ;
                        50                54    D0 00012          MOVL    PTR, R0                                        ;
                             00000000G    EF    16 00015          JSB     EXCH$UTIL_BLOCK_CHECK                          ;
                        53         54     A4    D0 0001B          MOVL    84(PTR), SPC                                   ; 1682
                                          4E    13 0001F          BEQL    9$                                             ;
            03          00         58     A4    8F 00021          CASEB   88(PTR), #0, #3                                ; 1689
   003D   0008        001D              0008       00026 1$:      .WORD   2$-1$,-
                                                                          3$-1$,-
                                                                          2$-1$,-
                                                                          7$-1$
                        7E         FA     8F    9A 0002E 2$:      MOVZBL  #250, -(SP)                                    ; 1706
                                          01    DD 00032          PUSHL   #1                                             ;
                             00000000G    8F    DD 00034          PUSHL   #EXCH$_BADLOGIC                                ;
              00000000G  00               03    FB 0003A          CALLS   #3, LIB$STOP                                   ;
```

```
                                    25  11 00041          BRB     8$                                          ;
                      50        12  B3  0F 00043 3$:      REMQUE  @18(SPC), _T_                               ; 1697
                                    04  1C 00047          BVC     4$                                          ;
                                    52  D4 00049          CLRL    ENT                                         ;
                                    03  11 0004B          BRB     5$                                          ;
                      52            50  D0 0004D 4$:      MOVL    _T_, ENT                                    ;
                      0B            52  E9 00050 5$:      BLBC    ENT, 6$                                     ;
                                    52  DD 00053          PUSHL   ENT                                         ; 1699
                                    1C  DD 00055          PUSHL   #28                                         ;
                 FE3B CF            02  FB 00057          CALLS   #2, EXCH$UTIL_VM_RELEASE                    ;
                                    E5  11 0005C          BRB     3$                                          ;
                      50            36  D0 0005E 6$:      MOVL    #54, BLOCK_SIZE                             ; 1701
                                    05  11 00061          BRB     8$                                          ; 1689
                      50      880E  8F  3C 00063 7$:      MOVZWL  #34830, BLOCK_SIZE                          ; 1704
                                    09  BB 00068 8$:      PUSHR   #^M<R0,R3>                                  ; 1710
                 FE28 CF            02  FB 0006A          CALLS   #2, EXCH$UTIL_VM_RELEASE                    ;
                      50            64  0F 0006F 9$:      REMQUE  (PTR), _T_                                  ; 1715
       50 00000000G   EF 000000C8   8F  C1 00072          ADDL3   #200, EXCH$A_GBL, R0                        ; 1719
                   04 B0            64  0E 0007E          INSQUE  (PTR), @4(R0)                               ;
                                        04 00082          RET                                                 ; 1722

; Routine Size:  131 bytes,    Routine Base:  EXCH$UTIL_CODE + 0708
```

```
1 GLOBAL ROUTINE exch$util_up_case (in_siz, in_ptr, out_ptr) : NOVALUE jsb_r1r2r3 =       %SBTTL 'exch$util_up
2 BEGIN
2 !++
2 !
2 ! FUNCTIONAL DESCRIPTION:
2 !
2 !       This routine converts a string to uppercase.  In testing it appears to be faster to do this sort
2 !       of loop than to execute the MOVTC instruction on the 117780.
2 !
2 ! INPUTS:
2 !
2 !       in_siz  = size of input record to convert
2 !       in_ptr  = address of input record to convert
2 !
2 ! IMPLICIT INPUTS:
2 !
2 !       none
2 !
2 ! OUTPUTS:
2 !
2 !       out_ptr = address of output record buffer
2 !
2 ! IMPLICIT OUTPUTS:
2 !
2 !       none
2 !
2 ! ROUTINE VALUE:
2 !
2 !       none
2 !
2 ! SIDE EFFECTS:
2 !
2 !       Input record copied to output record buffer and all
2 !       lowercase alphabetic characters converted to uppercase.
2 !--
2
2 REGISTER
2     char : BYTE                                        ! Character to test
2     ;
2
2 DECR count FROM .in_siz-1 TO 0                         ! Upcase the characters
2 DO
3     BEGIN
3     char = CH$RCHAR_A (in_ptr);                        ! Get next character
3     IF  .char GEQU 'a'  AND  .char LEQU 'z'            ! Lower case letter?
3     THEN
3         char = .char - %O'40';                         ! Convert to upper
3     CH$WCHAR_A (.char,out_ptr);                        ! Move character to buffer
3     END;
2
2 RETURN;
1 END;
```

```
                                     15  11 00000 EXCH$UTIL_UP_CASE::
                                                          BRB     3$                                                  ; 1763
                        50           82  90 00002 1$:     MOVB    (IN_PTR)+, CHAR                                     ; 1766
                 61     8F           50  91 00005         CMPB    CHAR, #97                                           ; 1767
                                     09  1F 00009         BLSSU   2$                                                  ;
                 7A     8F           50  91 0000B         CMPB    CHAR, #122                                          ;
                                     03  1A 0000F         BGTRU   2$                                                  ;
                        50           20  82 00011         SUBB2   #32, CHAR                                           ; 1769
                        83           50  90 00014 2$:     MOVB    CHAR, (OUT_PTR)+                                    ; 1770
                        E8           51  F4 00017 3$:     SOBGEQ  COUNT, 1$                                           ; 1763
                                         05 0001A         RSB                                                         ; 1774

; Routine Size:  27 bytes,    Routine Base:  EXCH$UTIL_CODE + 078B
;                       1 END
;                       0 ELUDOM



                                                                     .EXTRN  LIB$SIGNAL, LIB$STOP

;                                       PSECT SUMMARY
;
;       Name                         Bytes                       Attributes
;
;   EXCH$UTIL_CODE                        1958  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



;                               Library Statistics
;
;                                             -------- Symbols --------        Pages       Processing
;       File                                  Total    Loaded   Percent        Mapped      Time
;
;   _$255$DUA28:[SYSLIB]LIB.L32;1             18619        30         0         1000         00:01.8
;   _$255$DUA28:[EXCHNG.OBJ]EXCLIB.L32;1       1151       140        12           79         00:01.4



;                                       COMMAND QUALIFIERS

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

; Size:          1958 code + 0 data bytes
; Run Time:         00:40.4
; Elapsed Time:     02:18.9
; Lines/CPU Min:     2640
; Lexemes/CPU-Min: 23715
; Memory Used:  141 pages
; Compilation Complete
```

0163 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY