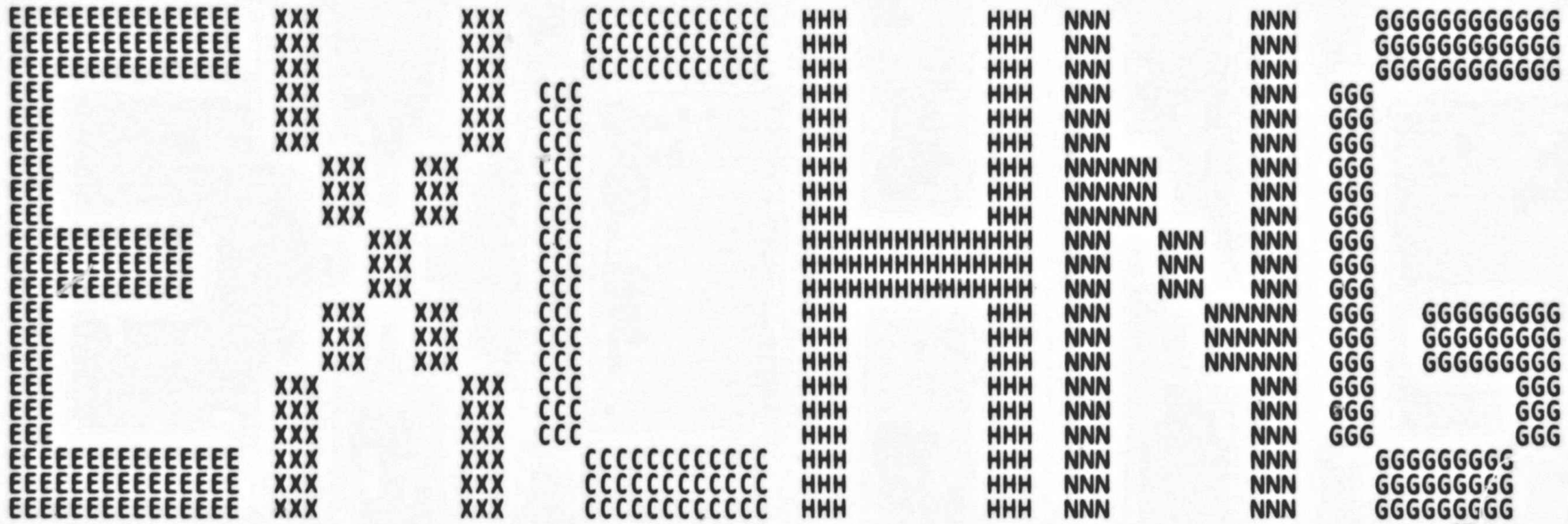
EEEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGGGG
EEEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGGGG
EEEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGGGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG
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 GGGGGGGG
EEE XXX XXX CCC HHH HHH NNN NNNNNN GGG GGGGGGGG
EEE XXX XXX CCC HHH HHH NNN NNNNNN GGG GGGGGGGG
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 CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGG
EEEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGG
EEEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGG

```
EEEEEEEEEE  XX      XX   CCCCCCCC  RRRRRRRR    TTTTTTTTTT    AAAAAA     CCCCCCCC  PPPPPPPP
EEEEEEEEEE  XX      XX   CCCCCCCC  RRRRRRRR    TTTTTTTTTT    AAAAAA     CCCCCCCC  PPPPPPPP
EE          XX      XX  CC        RR      RR      TT       AA      AA  CC        PP      PP
EE          XX      XX  CC        RR      RR      TT       AA      AA  CC        PP      PP
EE            XX  XX    CC        RR      RR      TT       AA      AA  CC        PP      PP
EE            XX  XX    CC        RR      RR      TT       AA      AA  CC        PP      PP
EEEEEEEE        XX      CC        RRRRRRRR        TT       AA      AA  CC        PPPPPPPP
EEEEEEEE        XX      CC        RRRRRRRR        TT       AA      AA  CC        PPPPPPPP
EE            XX  XX    CC        RR  RR          TT       AAAAAAAAAA  CC        PP
EE            XX  XX    CC        RR  RR          TT       AAAAAAAAAA  CC        PP
EE          XX      XX  CC        RR    RR        TT       AA      AA  CC        PP
EE          XX      XX  CC        RR    RR        TT       AA      AA  CC        PP
EEEEEEEEEE  XX      XX   CCCCCCCC  RR      RR      TT       AA      AA   CCCCCCCC  PP
EEEEEEEEEE  XX      XX   CCCCCCCC  RR      RR      TT       AA      AA   CCCCCCCC  PP
```

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

```
;  1     0001  0 MODULE  exch$rtacp                                %TITLE 'RT11 directory routines'
;  2     0002  0                 (
;  3     0003  0                 IDENT = 'V04-000',
;  4     0004  0                 ADDRESSING_MODE (EXTERNAL=LONG_RELATIVE, NONEXTERNAL=WORD_RELATIVE)
;  5     0005  0                 ) =
;  6     0006  1 BEGIN
;  7     0007  1 !
;  8     0008  1 !*****************************************************************************
;  9     0009  1 !*                                                                           *
; 10     0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
; 11     0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
; 12     0012  1 !*  ALL RIGHTS RESERVED.                                                     *
; 13     0013  1 !*                                                                           *
; 14     0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
; 15     0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
; 16     0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
; 17     0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
; 18     0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
; 19     0019  1 !*  TRANSFERRED.                                                             *
; 20     0020  1 !*                                                                           *
; 21     0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
; 22     0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
; 23     0023  1 !*  CORPORATION.                                                             *
; 24     0024  1 !*                                                                           *
; 25     0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
; 26     0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
; 27     0027  1 !*                                                                           *
; 28     0028  1 !*                                                                           *
; 29     0029  1 !*****************************************************************************
; 30     0030  1
; 31     0031  1 !++
; 32     0032  1 ! FACILITY:     EXCHANGE - Foreign volume interchange facility
; 33     0033  1 !
; 34     0034  1 ! ABSTRACT:     RT-11 directory manipulation routines
; 35     0035  1 !
; 36     0036  1 ! ENVIRONMENT:  VAX/VMS User mode
; 37     0037  1 !
; 38     0038  1 ! AUTHOR:       CW Hobbs                  CREATION DATE: 29-Nov-1982
; 39     0039  1 !
; 40     0040  1 ! MODIFIED BY:
; 41     0041  1 !
; 42     0042  1 !       V03-002 CWH3002         CW Hobbs                  12-Apr-1984
; 43     0043  1 !               If the directory shows a device larger than the actual
; 44     0044  1 !               device, then print a warning and write lock the volume.
; 45     0045  1 !
; 46     0046  1 !
; 47     0047  1 !--
; 48     0048  1
; 49     0049  1 ! Include files:
; 50     0050  1 !
; 51     0051  1 MACRO $module_name_string = 'exch$rtacp' %;     ! The require file needs to know our module name
; 52     0052  1 REQUIRE 'SRC$:EXCREQ'                           ! Facility-wide require file
; 53     0053  1     ;
```

```
;    55          0150  1 %SBTTL 'Module table of contents'
;    56          0151  1
;    57          0152  1 ! Module table of contents:
;    58          0153  1 !
;    59          0154  1 FORWARD ROUTINE
;    60          0155  1     exch$rtacp_check_position   : NOVALUE,      ! Find directory entry if it has moved
;    61          0156  1     exch$rtacp_clean_directory  : NOVALUE,      ! Shuffle and/or split directories as needed
;    62          0157  1     exch$rtacp_consolidate,                     ! Compress directory structure
;    63          0158  1     exch$rtacp_find_empty_area,                 ! Find free space
;    64          0159  1     exch$rtacp_find_file,                       ! RT-11 directory search routine
;    65          0160  1     exch$rtacp_next_entry,                      ! Retrieve next entry from RT-11 directory
;    66          0161  1     exch$rtacp_verify_directory                 ! Verify directory structure and compute volume size
;    67          0162  1     ;
;    68          0163  1
;    69          0164  1 ! EXCHANGE facility routines
;    70          0165  1 !
;    71          0166  1 EXTERNAL ROUTINE
;    72          0167  1     exch$cmd_fetch_recfmt_implied : NOVALUE,    ! Get or assume the value for /RECORD_FORMAT
;    73          0168  1     exch$cmd_match_filename,                    ! Compare expanded file names for match
;    74          0169  1     exch$io_rt11_read,                          ! Read blocks from a random access device
;    75          0170  1     exch$io_rt11_write,                     ! Write blocks to a random access device
;    76          0171  1     exch$pdp_flush_write_buffer,                ! Flush any records waiting in the write buffer
;    77          0172  1     exch$pdp_get,                               ! Get functions for small PDP record structure
;    78          0173  1     exch$pdp_put,                               ! Put functions for small PDP record structure
;    79          0174  1     exch$rt11_dirseg_flush,                     ! Write out directory segments
;    80          0175  1     exch$rt11_dirseg_get,                       ! Return pointer to specific directory segment
;    81          0176  1     exch$rt11_dirseg_get_nochk  : jsb_r1r2,     ! Return pointer to directory segment without checking
;    82          0177  1     exch$rt11_dirseg_put,                       ! Write a specific directory segment
;    83          0178  1     exch$rt11_expand_filename,                  ! Convert directory entry to ASCII filename
;    84          0179  1     exch$util_fao_buffer,                       ! Do an FAO conversion
;    85          0180  1     exch$util_radix50_from_ascii,               ! Convert characters to Radix-50 from Ascii
;    86          0181  1     exch$util_radix50_to_ascii,                 ! Convert characters from Radix-50 to Ascii
;    87          0182  1     exch$util_rt11ctx_allocate,                 ! Get an RT-11 context block
;    88          0183  1     exch$util_rt11ctx_release   : NOVALUE,      ! Give it back
;    89          0184  1     exch$util_vm_allocate,                      ! Get some virtual memory
;    90          0185  1     exch$util_vm_allocate_zeroed,               ! Get some virtual memory, initialized to zero
;    91          0186  1     exch$util_vm_release                        ! Return some virtual memory
;    92          0187  1     ;
;    93          0188  1
;    94          0189  1 ! Equated symbols:
;    95          0190  1 !
;    96          0191  1 !LITERAL
;    97          0192  1 !    ;
;    98          0193  1
;    99          0194  1 ! Bound declarations:
;   100          0195  1 !
;   101          0196  1 !BIND
;   102          0197  1 !    ;
```

```
;  104          0198  1 GLOBAL ROUTINE exch$rtacp_check_position (ctx : $ref_bblock) : NOVALUE =        %SBTTL 'exch$rtacp_check_pos
;  105          0199  2 BEGIN
;  106          0200  2 !++
;  107          0201  2 !
;  108          0202  2 ! FUNCTIONAL DESCRIPTION:
;  109          0203  2 !
;  110          0204  2 !       Check that the directory entry described in the context block is at the correct position.  Several e
;  111          0205  2 !       might have caused the directory to be rearranged, such as the creation of a file since the context w
;  112          0206  2 !       found.  If the entry has moved, find it and set the pointers to the new position.
;  113          0207  2 !
;  114          0208  2 ! INPUT/OUTPUT:
;  115          0209  2 !
;  116          0210  2 !       ctx - pointer to block describing the directory context
;  117          0211  2 !
;  118          0212  2 ! IMPLICIT INPUTS:
;  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 !       none
;  133          0227  2 !--
;  134          0228  2
;  135          0229  2 $dbgtrc_prefix ('exch$rtacp_check_position> ');
;  136          0230  2
;  137          0231  2 LOCAL
;  138          0232  2     seg : $ref_bblock,                          ! a pointer to the current directory segment
;  139          0233  2     ctx2 : $ref_bblock,                         ! a pointer to a second context in case we need to scan
;  140          0234  2     flags
;  141          0235  2     ;
;  142          0236  2
;  143          0237  2 BIND
;  144          0238  2     volb = ctx [rt11ctx$a_assoc_volb]   : $ref_bblock
;  145          0239  2     ;
;  146          0240  2
;  147          0241  2 $block_check (2, .ctx, rt11ctx, 443);
;  148          0242  2 $block_check (2, .volb, volb, 445);
;  149          0243  2 $logic_check (5, (exch$rtacp_verify_directory (.volb)), 182);
;  150          0244  2
;  151          0245  2
;  152          0246  2 ! We need to check to make sure that the number of segments has not been lowered.  If so, our entry address
;  153          0247  2 ! be sitting in the hyperspace beyond the segments in use.  Hence, grab a pointer to the root segment.
;  154          0248  2 !
;  155          0249  2 seg = exch$rt11_dirseg_get_nochk (.volb, 1);
;  156          0250  2 $logic_check (2, (.seg NEQ 0), 147);
```

```
;  158          0251  2 ! Check to see if the entry position info is correct, if so we can simply return
;  159          0252  2 !
;  160          0253  2 IF  .ctx [rt11ctx$l_seg_number] LEQU .seg [rt11hdr$w_high_seg]
;  161          0254  2   AND
;  162          0255  2     CH$EQL (rt11ctx$s_entry, ctx [rt11ctx$t_entry], rt11ctx$s_entry, .ctx [rt11ctx$a_ent_address])
;  163          0256  2 THEN
;  164          0257  2     RETURN;
;  165          0258  2
;  166          0259  2 $trace_print_lit ('looking for new position');
;  167          0260  2
;  168          0261  2 ! Get a context block for the search
;  169          0262  2 !
;  170          0263  2 ctx2 = exch$util_rt11ctx_allocate (.volb, 0);
;  171          0264  2
;  172          0265  2 ! Loop through all the segments in the directory looking for this file entry
;  173          0266  2 !
;  174          0267  2 flags = rtnxt$m_permanent OR rtnxt$m_empty OR rtnxt$m_tentative OR rtnxt$m_skip_check OR rtnxt$m_skip_expand
;  175          0268  2 WHILE exch$rtacp_next_entry (.ctx2, .flags) NEQ 0
;  176          0269  2 DO
;  177          0270  3     BEGIN
;  178          0271  3
;  179          0272  3     IF CH$EQL (rt11ctx$s_entry, ctx [rt11ctx$t_entry], rt11ctx$s_entry, ctx2 [rt11ctx$t_entry])
;  180          0273  3     THEN
;  181          0274  4         BEGIN
;  182          0275  4
;  183          0276  4         $trace_print_fao ('Relocated "!AF"', .ctx [rt11ctx$l_exp_fullname_len], ctx [rt11ctx$t_exp_fullname]
;  184          0277  4
;  185          0278  4         ! If the file has moved we are in deep trouble
;  186          0279  4         !
;  187          0280  4         $logic_check (3, (.ctx [rt11ctx$l_start_block] EQL .ctx2 [rt11ctx$l_start_block]), 120);
;  188          0281  4
;  189          0282  4         ! Put the new position info into the context block
;  190          0283  4         !
;  191          0284  4         ctx [rt11ctx$l_seg_number]  = .ctx2 [rt11ctx$l_seg_number];
;  192          0285  4         ctx [rt11ctx$a_seg_address] = .ctx2 [rt11ctx$a_seg_address];
;  193          0286  4         ctx [rt11ctx$a_ent_address] = .ctx2 [rt11ctx$a_ent_address];
;  194          0287  4
;  195          0288  4         ! Give the extra context back
;  196          0289  4         !
;  197          0290  4         exch$util_rt11ctx_release (.ctx2);
;  198          0291  4
;  199          0292  4         RETURN;
;  200          0293  3         END;
;  201          0294  3
;  202          0295  2     END;
;  203          0296  2
;  204          0297  2 $logic_check (0, (false), 121);                 ! We should have found it
;  205          0298  2
;  206          0299  2 RETURN;
;  207          0300  1 END;


                                .TITLE  EXCH$RTACP RT11 directory routines
                                .IDENT  \V04-000\

                                .EXTRN  EXCH$CMD_FETCH_RECFMT_IMPLIED
                                .EXTRN  EXCH$CMD_MATCH_FILENAME
```

```
                                        .EXTRN  EXCH$IO_RT11_READ
                                        .EXTRN  EXCH$IO_RT11_WRITE
                                        .EXTRN  EXCH$PDP_FLUSH_WRITE_BUFFER
                                        .EXTRN  EXCH$PDP_GET, EXCH$PDP_PUT
                                        .EXTRN  EXCH$RT11_DIRSEG_FLUSH
                                        .EXTRN  EXCH$RT11_DIRSEG_GET
                                        .EXTRN  EXCH$RT11_DIRSEG_GET_NOCHK
                                        .EXTRN  EXCH$RT11_DIRSEG_PUT
                                        .EXTRN  EXCH$RT11_EXPAND_FILENAME
                                        .EXTRN  EXCH$UTIL_FAO_BUFFER
                                        .EXTRN  EXCH$UTIL_RADIX50_FROM_ASCII
                                        .EXTRN  EXCH$UTIL_RADIX50_TO_ASCII
                                        .EXTRN  EXCH$UTIL_RT11CTX_ALLOCATE
                                        .EXTRN  EXCH$UTIL_RT11CTX_RELEASE
                                        .EXTRN  EXCH$UTIL_VM_ALLOCATE
                                        .EXTRN  EXCH$UTIL_VM_ALLOCATE_ZEROED
                                        .EXTRN  EXCH$UTIL_VM_RELEASE
                                        .EXTRN  EXCH$UTIL_BLOCK_CHECK
                                        .EXTRN  EXCH$_BADLOGIC

                                        .PSECT  EXCH$RTACP_CODE,NOWRT,2

                                03FC 00000      .ENTRY  EXCH$RTACP_CHECK_POSITION, Save R2,R3,R4,-      ; 0198
                                                        R5,R6,R7,R8,R9
                            59 00000000G   EF   9E 00002    MOVAB   EXCH$UTIL_BLOCK_CHECK, R9           ;
                            58 00000000G   00   9E 00009    MOVAB   LIB$STOP, R8                        ;
                            57 00000000G   8F   D0 00010    MOVL    #EXCH$_BADLOGIC, R7                 ;
                            54       04    AC   D0 00017    MOVL    CTX, R4                             ; 0238
                            52 008200F4    8F   D0 0001B    MOVL    #8519924, R2                        ; 0241
                            51     01BB    8F   3C 00022    MOVZWL  #443, R1                            ;
                            50             54   D0 00027    MOVL    R4, R0                              ;
                                           69   16 0002A    JSB     EXCH$UTIL_BLOCK_CHECK               ;
                            52 041B00F3    8F   D0 0002C    MOVL    #68878579, R2                       ; 0242
                            51     01BD    8F   3C 00033    MOVZWL  #445, R1                            ;
                            50       14    A4   D0 00038    MOVL    20(R4), R0                          ;
                                           69   16 0003C    JSB     EXCH$UTIL_BLOCK_CHECK               ;
                            52             01   D0 0003E    MOVL    #1, R2                              ; 0249
                            51       14    A4   D0 00041    MOVL    20(R4), R1                          ;
                                00000000G  EF   16 00045    JSB     EXCH$RT11_DIRSEG_GET_NOCHK          ;
                            52             50   D0 0004B    MOVL    R0, SEG                             ;
                                           0B   12 0004E    BNEQ    1$                                  ; 0250
                            7E       93    8F   9A 00050    MOVZBL  #147, -(SP)                         ;
                                           01   DD 00054    PUSHL   #1                                  ;
                                           57   DD 00056    PUSHL   R7                                  ;
                            68             03   FB 00058    CALLS   #3, LIB$STOP                        ;
76  A4      04  A2             10         00   ED 0005B 1$: CMPZV   #0, #16, 4(SEG), 118(R4)            ; 0253
                                           08   1F 00062    BLSSU   2$                                  ;
                        7E  B4      38  A4         0E   29 00064    CMPC3   #14, 56(R4), @126(R4)       ; 0255
                                           45   13 0006A    BEQL    5$                                  ;
                                           7E   D4 0006C 2$: CLRL    -(SP)                               ; 0263
                                       14  A4         DD   DD 0006E    PUSHL   20(R4)                      ;
                                09000000G  EF         02   FB 00071    CALLS   #2, EXCH$UTIL_RT11CTX_ALLOCATE  ;
                                           55         50   D0 00078    MOVL    R0, CTX2                    ;
                                           56         37   D0 0007B    MOVL    #55, FLAGS                  ; 0267
                                           7E         55   7D 0007E 3$: MOVQ    CTX2, -(SP)                 ; 0268
                                 0000V     CF         02   FB 00081    CALLS   #2, EXCH$RTACP_NEXT_ENTRY   ;
                                                      50   D5 00086    TSTL    R0                          ;
```

```
                                        1C  13 00088          BEQL    4$                                              ;
                38   A5        38   A4    0E  29 0008A          CMPC3   #14, 56(R4), 56(CTX2)                           ; 0272
                                        EC  12 00090          BNEQ    3$                                              ;
                          76    A4   76   A5  7D 00092          MOVQ    118(CTX2), 118(R4)                              ; 0284
                          7E    A4   7E   A5  D0 00097          MOVL    126(CTX2), 126(R4)                              ; 0286
                                        55  DD 0009C          PUSHL   CTX2                                            ; 0290
                    00000000G   EF        01  FB 0009E          CALLS   #1, EXCH$UTIL_RT11CTX_RELEASE                   ;
                                            04 000A5          RET                                                     ; 0280
                          7E        79  8F  9A 000A6 4$:      MOVZBL  #121, -(SP)                                     ; 0297
                                        01  DD 000AA          PUSHL   #1                                              ;
                                        57  DD 000AC          PUSHL   R7                                              ;
                          68            03  FB 000AE          CALLS   #3, LIB$STOP                                    ;
                                            04 000B1 5$:      RET                                                     ; 0300

; Routine Size:  178 bytes,    Routine Base:  EXCH$RTACP_CODE + 0000
```

```
;  209     0301  1 GLOBAL ROUTINE exch$rtacp_clean_directory (volb : $ref_bblock,  %SBTTL 'exch$rtacp_clean_directory (volb, en
;  210     0302  1                                                                   ent_cnt, ent_len) : NOVALUE =
;  211     0303  2 BEGIN
;  212     0304  2 !++
;  213     0305  2 !
;  214     0306  2 ! FUNCTIONAL DESCRIPTION:
;  215     0307  2 !
;  216     0308  2 !       This routine is used like the RT-11 segment split logic, i.e. it makes room for a directory entry.
;  217     0309  2 !       of a simple segment split, it may restructure the entire directory.  It will add or remove segments
;  218     0310  2 !       the chain to produce a directory with approximately the same number of active entries in each segmen
;  219     0311  2 !
;  220     0312  2 !       This routine assumes that it has been called from exch$rtacp_consolidate.  It assumes that every dir
;  221     0313  2 !       segment contains only valid directory entries and is terminated by an end-segment marker.  It does n
;  222     0314  2 !       require that unnecessary entries (see exch$rtacp_consolidate) be removed.
;  223     0315  2 !
;  224     0316  2 ! INPUTS:
;  225     0317  2 !
;  226     0318  2 !       volb    - pointer to volb which has been connected to the RT-11 device
;  227     0319  2 !       ent_cnt - number of entries in the directory
;  228     0320  2 !       ent_len - length of a single entry, including extra bytes
;  229     0321  2 !
;  230     0322  2 ! IMPLICIT INPUTS:
;  231     0323  2 !
;  232     0324  2 !       none
;  233     0325  2 !
;  234     0326  2 ! OUTPUTS:
;  235     0327  2 !
;  236     0328  2 !       none
;  237     0329  2 !
;  238     0330  2 ! IMPLICIT OUTPUTS:
;  239     0331  2 !
;  240     0332  2 !       none
;  241     0333  2 !
;  242     0334  2 ! ROUTINE VALUE:
;  243     0335  2 !
;  244     0336  2 !       none
;  245     0337  2 !
;  246     0338  2 ! SIDE EFFECTS:
;  247     0339  2 !
;  248     0340  2 !       all error conditions are fatal
;  249     0341  2 !--
;  250     0342  2
;  251     0343  2 $dbgtrc_prefix ('exch$rtacp_clean_directory> ');
;  252     0344  2
;  253     0345  2 LOCAL
;  254     0346  2     seg : $ref_bblock,                  ! a pointer to the current directory segment
;  255     0347  2     cur : $ref_bblock,                  ! a pointer to the current directory entry
;  256     0348  2     dir2 : $ref_bblock,                 ! a pointer to an area where we will assemble a clean direct
;  257     0349  2     seg2 : $ref_bblock,                 ! a pointer to the current segment in the clean directory ar
;  258     0350  2     cur2 : $ref_bblock,                 ! a pointer to the current entry in the clean directory area
;  259     0351  2     ctx : $ref_bblock,                  ! a pointer a context block for looping through the dirty di
;  260     0352  2     ent2_cnt,                           ! count of entries in the current clean segment
;  261     0353  2     seg2_num,                           ! current segment number in the clean directory
;  262     0354  2     seg_cnt,                            ! number of segments for the cleaned directory
;  263     0355  2     ent_per_seg,                        ! goal for number of entries per segment
;  264     0356  2     max_ent_per_seg,                    ! max number of entries per segment
;  265     0357  2     flags                               ! local for our flags to exch$rtacp_next_entry
```

```
;  266          0358  2      ;
;  267          0359  2
;  268          0360  2 $debug_print_lit ('entry');
;  269          0361  2
;  270          0362  2 $block_check (2, .volb, volb, 533);
;  271          0363  2 $logic_check (4, (exch$rtacp_verify_directory (.volb)), 183);
;  272          0364  2
;  273          0365  2
;  274          0366  2 seg = exch$rt11_dirseg_get (.volb, 1);                       ! Get the pointer to the dirty root segment
;  275          0367  2 $logic_check (2, (.seg NEQ 0), 151);
;  276          0368  2
;  277          0369  2 ! Figure out the maximum number of file entries which can fit into a single segment
;  278          0370  2 !
;  279          0371  3 max_ent_per_seg =  (rt11$k_dirseglen -           ! Length of entire segment
;  280          0372  3                     rt11hdr$k_length -           !  less the length of the segment header
;  281          0373  3                     2)                           !  less two bytes for the end-of-segment marker
;  282          0374  2                     /  .ent_len;                 ! Divided by the length of a single entry
;  283          0375  2
;  284          0376  2 $trace_print_fao ('desired entries !UL, max_ent_per_seg !UL', .ent_cnt, .max_ent_per_seg);
;  285          0377  2
;  286          0378  2 ! Determine how many segments we need to leave two free entries per segment.  This gives the final segment c
;  287          0379  2 !
;  288          0380  2 ent_per_seg = .max_ent_per_seg - 2;                        ! Leave room for two entries in each segment
;  289          0381  2 seg_cnt = (.ent_cnt / .ent_per_seg) +                     ! Need one extra segment if not even division
;  290          0382  2         (IF ((.ent_cnt MOD .ent_per_seg) NEQ 0) THEN 1 ELSE 0);
;  291          0383  2 IF .seg_cnt GTRU .seg [rt11hdr$w_num_segs]
;  292          0384  2 THEN
;  293          0385  2     seg_cnt = .seg [rt11hdr$w_num_segs];                   ! We know that we can be at most one over the limit
;  294          0386  2
;  295          0387  2 ! Using the final segment count, now try to evenly distribute the entries around the segments.  This gives t
;  296          0388  2 ! final entries per segment count.
;  297          0389  2 !
;  298          0390  2 ent_per_seg = (.ent_cnt / .seg_cnt) +                     ! Need one extra entry if not even division
;  299          0391  2         (IF ((.ent_cnt MOD .seg_cnt) NEQ 0) THEN 1 ELSE 0);
;  300          0392  2 $logic_check (4, (.ent_per_seg LEQU .max_ent_per_seg), 180);
;  301          0393  2
;  302          0394  2 $trace_print_fao ('final segs !UL, goal ent_per_seg !UL', .seg_cnt, .ent_per_seg);
;  303          0395  2
;  304          0396  2 ! Grab a chunk of memory which will be used to build a clean copy of the directory.  We haven't modified any
;  305          0397  2 ! yet, so there will be no problem with corrupting anything if this allocate fails (it will SIGNAL_STOP).
;  306          0398  2 !
;  307          0399  2 dir2 = exch$util_vm_allocate_zeroed (.seg_cnt * rt11$k_dirseglen);
```

```
;  309          0400  2 ! OK, our new directory will contain .SEG_CNT segments each containing .ENT_PER_SEG entries, with possibly f
;  310          0401  2 ! entries in the last segment.  Start building a clean copy of the directory.
;  311          0402  2 !
;  312          0403  2
;  313          0404  2 ! Set the segment header for the root segment.  rt11hdr$w_high_seg won't be set until later
;  314          0405  2 !
;  315          0406  2 seg2 = .dir2;                                          ! Get the pointer to the clean root segment
;  316          0407  2 seg2 [rt11hdr$w_num_segs] = .seg [rt11hdr$w_num_segs];  ! Haven't changed the total number of segs
;  317          0408  2 seg2 [rt11hdr$w_next_seg] = (IF .seg_cnt EQL 1 THEN 0 ELSE 2);  ! We will link them in order
;  318          0409  2 seg2 [rt11hdr$w_extra_bytes] = .seg [rt11hdr$w_extra_bytes];  ! This stays the same
;  319          0410  2 seg2 [rt11hdr$w_start_block] = .seg [rt11hdr$w_start_block];  ! Same now, but may not be for other segment
;  320          0411  2
;  321          0412  2 ! Set up the control variables for the transfer loop
;  322          0413  2 !
;  323          0414  2 seg2_num = 1;                                         ! First segment in the clean directory
;  324          0415  2 cur2 = .seg2 + rt11hdr$k_length;                      ! Point at the first available entry position in the clean
;  325          0416  2 ent2_cnt = 0;                                         ! No entries in the clean directory
;  326          0417  2
;  327          0418  2 ctx = exch$util_rt11ctx_allocate (.volb, 0);    ! Get a context block for the search
;  328          0419  2
;  329          0420  2 ! Loop through all the segments in the dirty directory, transferring entries to the clean directory
;  330          0421  2 !
;  331          0422  2 flags = rtnxt$m_permanent OR rtnxt$m_empty OR rtnxt$m_tentative OR rtnxt$m_skip_check OR rtnxt$m_skip_expand
;  332          0423  2 WHILE (cur = exch$rtacp_next_entry (.ctx, .flags)) NEQ 0
;  333          0424  2 DO
;  334          0425  3     BEGIN
;  335          0426  3
;  336          0427  3     CASE .cur [rt11ent$v_type] FROM 0 TO rt11ent$m_typ_end_segment OF
;  337          0428  3     SET
;  338          0429  3
;  339          0430  3     [INRANGE, OUTRANGE] :                             ! We had better give up before something happens
;  340          0431  3             $logic_check (0, (false), 153);
;  341          0432  3
;  342          0433  3     [rt11ent$m_typ_tentative, rt11ent$m_typ_empty, rt11ent$m_typ_permanent] :
;  343          0434  4             BEGIN
;  344          0435  4
;  345          0436  4             ! If we have filled this clean segment, then start a new segment before we do any moves
;  346          0437  4             !
;  347          0438  5             IF  (.ent2_cnt GEQU .ent_per_seg)
;  348          0439  4             THEN
;  349          0440  5                 BEGIN
;  350          0441  5
;  351          0442  5                 ! Set the end-of-segment marker in the current clean segment
;  352          0443  5                 !
;  353          0444  5                 $logic_check (3, (.cur2 LSSU .seg2 + rt11$k_dirseglen), 157);   ! We need room for marker
;  354          0445  5                 cur2 [rt11ent$b_type_byte] = rt11ent$m_typ_end_segment;
;  355          0446  5
;  356          0447  5                 ! Reset the pointers and counters for the clean segment
;  357          0448  5                 !
;  358          0449  5                 ent2_cnt = 0;                                          ! No entries yet
;  359          0450  5                 seg2_num = .seg2_num + 1;                              ! Adding one more segment
;  360          0451  5                 $logic_check (3, (.seg2_num LEQU .seg_cnt), 158);                 !  which had better
;  361          0452  5                 seg2 = .seg2 + rt11$k_dirseglen;                       ! Move to the new segment
;  362          0453  5                 cur2 = .seg2 + rt11hdr$k_length;                       ! First entry in the seg
;  363          0454  5
;  364          0455  5                 ! Initialize the segment header for the new segment
;  365          0456  5                 !
```

```
;  366     0457  5                 seg2 [rt11hdr$w_num_segs] = .seg [rt11hdr$w_num_segs];  ! Same total # of segs
;  367     0458  6                 seg2 [rt11hdr$w_next_seg] = (IF .seg2_num EQL .seg_cnt  ! We will link them in order
;  368     0459  5                                             THEN 0 ELSE .seg2_num+1);   !  marking the end with a 0
;  369     0460  5                 seg2 [rt11hdr$w_extra_bytes] = .seg [rt11hdr$w_extra_bytes];! This stays the same
;  370     0461  5                 seg2 [rt11hdr$w_start_block] = .ctx [rt11ctx$l_start_block];! First file starts here
;  371     0462  5
;  372     0463  4                 END;
;  373     0464  4
;  374     0465  4             ! Transfer this entry to the clean segment
;  375     0466  4             !
;  376     0467  4             CH$MOVE (.ent_len, .cur, .cur2);
;  377     0468  4
;  378     0469  4             ! Bump the clean segment pointers
;  379     0470  4             !
;  380     0471  4             cur2 = .cur2 + .ent_len;    ! Move the entry pointer
;  381     0472  4             ent2_cnt = .ent2_cnt + 1;   ! Adjust the entry count for this segment
;  382     0473  4             ent_cnt = .ent_cnt - 1;             ! Adjust the total entries remaining count, just a check
;  383     0474  4
;  384     0475  3             END;
;  385     0476  3     TES;
;  386     0477  2     END;
;  387     0478  2
;  388     0479  2 exch$util_rt11ctx_release (.ctx);               ! Give the context block back
;  389     0480  2
;  390     0481  2 ! Set the end-of-segment marker in the final clean segment
;  391     0482  2 !
;  392     0483  2 $logic_check (3, (.cur2 LSSU .seg2 + rt11$k_dirseglen), 159);   ! We need room for marker
;  393     0484  2 cur2 [rt11ent$b_type_byte] = rt11ent$m_typ_end_segment;
;  394     0485  2 $logic_check (3, ((.ent_cnt EQL 0)), 160);                       ! Make sure we moved everybody
;  395     0486  2
;  396     0487  2 ! Set the directory header info in the first and final segments
;  397     0488  2 !
;  398     0489  2 seg2 [rt11hdr$w_next_seg] = 0;                  ! Last segment is end of chain
;  399     0490  2 dir2 [rt11hdr$w_high_seg] = .seg2_num;          ! Last segment is highest segment in use
;  400     0491  2
;  401     0492  2 !?? need to deal with the following situation (most likely to occur when /EXTRA=119):
;  402     0493  2 !??  attempting to completely fill each segment (e.g. seg_cnt=31 and ent_per_seg=4), however we end up
;  403     0494  2 !??  filling 23 segs with 4, the 24th seg gets 3, we quit but the insert fails because only one entry availa
;  404     0495  2 !??  in segment.
;  405     0496  2 !?? (If seg2_num<seg_cnt and <2 free entries in last segment split the last segment into 2?
;  406     0497  2 !??  This doesn't help if free space is in an earlier segment unless you squeeze)
```

```
;  408          0498  2 ! Now that we have a clean "duplicate" of the directory, make it the real directory and update things on dis
;  409          0499  2 ! Since it is possible (in fact likely as the volume gets full) that some of the "clean" directory segments
;  410          0500  2 ! identical to "dirty" ones, we will compare each of the segments before moving and updating them.
;  411          0501  2 !
;  412          0502  2 seg2 = .dir2;                                        ! Get the pointer to the clean root segment
;  413          0503  2
;  414          0504  2 INCRU num FROM 1 TO .seg2_num
;  415          0505  2 DO
;  416          0506  2     BEGIN
;  417          0507  2
;  418          0508  2     ! Compare the dirty segment with the clean one
;  419          0509  2     !
;  420          0510  3     IF CH$NEQ (rt11$k_dirseglen, .seg, rt11$k_dirseglen, .seg2)
;  421          0511  3     THEN
;  422          0512  4         BEGIN
;  423          0513  4
;  424          0514  4         CH$MOVE (rt11$k_dirseglen, .seg2, .seg);        ! Move the clean segment on top of the dirty segment
;  425          0515  4         exch$rt11_dirseg_put (.volb, .num);             ! Now write the segment to disk
;  426          0516  4
;  427          0517  4         END;
;  428          0518  3
;  429          0519  3     ! Move our segment pointers
;  430          0520  3     !
;  431          0521  3     seg = .seg + rt11$k_dirseglen;
;  432          0522  3     seg2 = .seg2 + rt11$k_dirseglen;
;  433          0523  3
;  434          0524  2     END;
;  435          0525  2
;  436          0526  2 ! Return the memory for the duplicate directory
;  437          0527  2 !
;  438          0528  2 exch$util_vm_release ((.seg_cnt * rt11$k_dirseglen), .dir2);
;  439          0529  2
;  440          0530  2 ! Check the new directory out
;  441          0531  2 !
;  442          0532  2 $logic_check (3, (exch$rtacp_verify_directory (.volb)), 184);
;  443          0533  2
;  444          0534  2 ! Now, consolidate again.  This is necessary because we might have adjacent empty entries which can be colla
;  445          0535  2 ! We give the "2" flag to prevent another clean which would result in a recursive loop.
;  446          0536  2 !
;  447          0537  2 exch$rtacp_consolidate (.volb, 2);
;  448          0538  2
;  449          0539  2 RETURN;
;  450          0540  1 END;


                                 0FFC 00000          .ENTRY  EXCH$RTACP_CLEAN_DIRECTORY, Save R2,R3,R4,- ; 0301
                                                             R5,R6,R7,R8,R9,R10,R11
                      5E                 18   C2 00002      SUBL2   #24, SP                                     ;
                      52 041B00F3        8F   D0 00005      MOVL    #68878579, R2                               ; 0362
                      51     0215        8F   3C 0000C      MOVZWL  #533, R1                                    ;
                      50       04        AC   D0 00011      MOVL    VOLB, R0                                    ;
                         00000000G       EF   16 00015      JSB     EXCH$UTIL_BLOCK_CHECK                       ;
                                 01        DD   DD 0001B      PUSHL   #1                                          ; 0366
                                   04      AC   DD 0001D      PUSHL   VOLB                                        ;
```

```
                00000000G  EF              02  FB 00020          CALLS   #2, EXCH$RT11_DIRSEG_GET                          ;
                           5B              50  D0 00027          MOVL    R0, SEG                                          ;
                                           13  12 0002A          BNEQ    1$                                               ; 0367
                           7E       97     8F  9A 0002C          MOVZBL  #151, -(SP)                                      ;
                                           01  DD 00030          PUSHL   #1                                               ;
                    00000000G      8F      DD 00032          PUSHL   #EXCH$_BADLOGIC                                  ;
                00000000G  00              03  FB 00038          CALLS   #3, LIB$STOP                                     ;
             50 000003F4   8F       0C     AC  C7 0003F 1$:     DIVL3   ENT_LEN, #1012, MAX_ENT_PER_SEG                  ; 0374
                           5A       FE     A0  9E 00048          MOVAB   -2(R0), ENT_PER_SEG                              ; 0380
             51         08  AC              5A  C7 0004C          DIVL3   ENT_PER_SEG, ENT_CNT, R1                         ; 0381
          7E 00         08  AC              01  7A 00051          EMUL    #1, ENT_CNT, #0, -(SP)                           ; 0382
          50 50            8E              5A  7B 00057          EDIV    ENT_PER_SEG, (SP)+, R0, R0                       ;
                                           50  D5 0005C          TSTL    R0                                               ;
                                           05  13 0005E          BEQL    2$                                               ;
                           50              01  D0 00060          MOVL    #1, R0                                           ;
                                           02  11 00063          BRB     3$                                               ;
                                           50  D4 00065 2$:      CLRL    R0                                               ;
             58            51              50  C1 00067 3$:      ADDL3   R0, R1, SEG_CNT                                  ;
          58 6B            10              00  ED 0006B          CMPZV   #0, #16, (SEG), SEG_CNT                          ; 0383
                                           03  1E 00070          BGEQU   4$                                               ;
                           58              6B  3C 00072          MOVZWL  (SEG), SEG_CNT                                   ; 0385
             51         08  AC              58  C7 00075 4$:      DIVL3   SEG_CNT, ENT_CNT, R1                             ; 0390
          7E 00         08  AC              01  7A 0007A          EMUL    #1, ENT_CNT, #0, -(SP)                           ; 0391
          50 50            8E              58  7B 00080          EDIV    SEG_CNT, (SP)+, R0, R0                           ;
                                           50  D5 00085          TSTL    R0                                               ;
                                           05  13 00087          BEQL    5$                                               ;
                           50              01  D0 00089          MOVL    #1, R0                                           ;
                                           02  11 0008C          BRB     6$                                               ;
                                           50  D4 0008E 5$:      CLRL    R0                                               ;
          5A               51              50  C1 00090 6$:      ADDL3   R0, R1, ENT_PER_SEG                              ;
       14 AE               58              0A  78 00094          ASHL    #10, SEG_CNT, 20(SP)                             ; 0399
                       14  AE              DD 00099          PUSHL   20(SP)                                           ;
                00000000G  EF              01  FB 0009C          CALLS   #1, EXCH$UTIL_VM_ALLOCATE_ZEROED                 ;
                        0C  AE              50  D0 000A3          MOVL    R0, DIR2                                         ;
                           56           0C  AE  D0 000A7          MOVL    DIR2, SEG2                                       ; 0406
                           66              6B  B0 000AB          MOVW    (SEG), (SEG2)                                    ; 0407
                           01              58  D1 000AE          CMPL    SEG_CNT, #1                                      ; 0408
                                           04  12 000B1          BNEQ    7$                                               ;
                                           50  D4 000B3          CLRL    R0                                               ;
                                           03  11 000B5          BRB     8$                                               ;
                           50              02  D0 000B7 7$:      MOVL    #2, R0                                           ;
                        02 A6              50  B0 000BA 8$:      MOVW    R0, 2(SEG2)                                      ;
                        06 A6           06  AB  D0 000BE          MOVL    6(SEG), 6(SEG2)                                  ; 0409
                           59              01  D0 000C3          MOVL    #1, SEG2_NUM                                     ; 0414
                           57           0A  A6  9E 000C6          MOVAB   10(R6), CUR2                                     ; 0415
                                        08  AE  D4 000CA          CLRL    ENT2_CNT                                         ; 0416
                                           7E  D4 000CD          CLRL    -(SP)                                            ; 0418
                                        04  AC  DD 000CF          PUSHL   VOLB                                             ;
                00000000G  EF              02  FB 000D2          CALLS   #2, EXCH$UTIL_RT11CTX_ALLOCATE                   ;
                           6E              50  D0 000D9          MOVL    R0, CTX                                          ;
                        10 AE              37  D0 000DC          MOVL    #55, FLAGS                                       ; 0422
                                        10  AE  DD 000E0 9$:      PUSHL   FLAGS                                            ; 0423
                                        04  AE  DD 000E3          PUSHL   CTX                                              ;
                     0000V CF              02  FB 000E6          CALLS   #2, EXCH$RTACP_NEXT_ENTRY                        ;
                        04 AE              50  D0 000EB          MOVL    R0, CUR                                          ;
                                           03  12 000EF          BNEQ    10$                                              ;
                                     0084  31  00F1              BRW     18$                                              ;
```

```
                7E      04   AE              01  C1 000F4 10$:    ADDL3   #1, CUR, -(SP)                                  ;0427
             7E 9E           04              00  EF 000F9         EXTZV   #0, #4, @(SP)+, -(SP)
                08           00              8E  CF 000FE         CASEL   (SP)+, #0, #8
          0012      0027           0027          0012     00102 11$:  .WORD  12$-11$,-
          0012      0012           0012          0027     0010A              14$-11$,-
                                                 0012     00112              14$-11$,-
                                                                             12$-11$,-
                                                                             14$-11$,-
                                                                             12$-11$,-
                                                                             12$-11$,-
                                                                             12$-11$,-
                                                                             12$-11$
                        7E         99    8F  9A 00114 12$:    MOVZBL  #153, -(SP)                                     ;0431
                                         01  DD 00118         PUSHL   #1
                     00000000G           8F  DD 0011A         PUSHL   #EXCH$_BADLOGIC
                00000000G   00           03  FB 00120         CALLS   #3, LIB$STOP
                                         B7  11 00127 13$:    BRB     9$
                        5A         08    AE  D1 00129 14$:    CMPL    ENT2_CNT, ENT_PER_SEG                           ;0438
                                         37  1F 0012D         BLSSU   17$
                     01    A7            08  90 0012F         MOVB    #8, 1(CUR2)                                     ;0445
                                   08    AE  D4 00133         CLRL    ENT2_CNT                                        ;0449
                                         59  D6 00136         INCL    SEG2_NUM                                        ;0450
                        56       0400     C6  9E 00138         MOVAB   1024(R6), SEG2                                  ;0452
                        57         0A     A6  9E 0013D         MOVAB   10(R6), CUR2                                    ;0453
                        66                6B  B0 00141         MOVW    (SEG), (SEG2)                                   ;0457
                        58                59  D1 00144         CMPL    SEG2_NUM, SEG_CNT                               ;0458
                                          04  12 00147         BNEQ    15$
                                          50  D4 00149         CLRL    R0
                                          04  11 0014B         BRB     16$
                        50         01     A9  9E 0014D 15$:    MOVAB   1(R9), R0                                       ;0459
                   02   A6                50  B0 00151 16$:    MOVW    R0, 2(SEG2)                                     ;0458
                   06   A6          06    AB  B0 00155         MOVW    6(SEG), 6(SEG2)                                 ;0460
             50         6E  00000072      8F  C1 0015A         ADDL3   #114, CTX, R0                                   ;0461
                   08   A6                60  B0 00162         MOVW    (R0), 8(SEG2)
          67       04   BE          0C    AC  28 00166 17$:    MOVC3   ENT_LEN, @CUR, (CUR2)                           ;0467
                        57          0C    AC  C0 0016C         ADDL2   ENT_LEN, CUR2                                   ;0471
                                    08    AE  D6 00170         INCL    ENT2_CNT                                        ;0472
                                    08    AC  D7 00173         DECL    ENT_CNT                                         ;0473
                                          AF  11 00176         BRB     13$                                             ;0423
                                          6E  DD 00178 18$:    PUSHL   CTX                                             ;0479
                00000000G   EF            01  FB 0017A         CALLS   #1, EXCH$UTIL_RT11CTX_RELEASE
                      01    A7            08  90 00181         MOVB    #8, 1(CUR2)                                     ;0484
                                    02    A6  B4 00185         CLRW    2(SEG2)                                         ;0489
             50         0C   AE          04  C1 00188         ADDL3   #4, DIR2, R0                                    ;0490
                             60           59  B0 0018D         MOVW    SEG2_NUM, (R0)
                             56          0C   AE  D0 00190     MOVL    DIR2, SEG2                                      ;0502
                             58               01  D0 00194     MOVL    #1, NUM                                         ;0504
                                              26  11 00197     BRB     21$
          66                 6B        0400   8F  29 00199 19$:    CMPC3   #1024, (SEG), (SEG2)                        ;0510
                                              12  13 0019F     BEQL    20$
          6B                 66        0400   8F  28 001A1     MOVC3   #1024, (SEG2), (SEG)                        ;0514
                                              58  DD 001A7     PUSHL   NUM                                             ;0515
                                        04    AC  DD 001A9     PUSHL   VOLB
                00000000G   EF                02  FB 001AC     CALLS   #2, EXCH$RT11_DIRSEG_PUT
                             5B        0400   CB  9E 001B3 20$:    MOVAB   1024(R11), SEG                              ;0521
                             56        0400   C6  9E 001B8     MOVAB   1024(R6), SEG2                                  ;0522
                                              58  D6 001BD     INCL    NUM
```

```
                                59              58  D1 001BF 21$:    CMPL    NUM, SEG2_NUM                           ;
                                                D5  1B 001C2         BLEQU   19$                                     ;
                                            0C  AE  DD 001C4         PUSHL   DIR2                                    ; 0528
                                            18  AE  DD 001C7         PUSHL   24(SP)                                  ;
                          00000000G         EF      02 FB 001CA      CALLS   #2, EXCH$UTIL_VM_RELEASE                 ;
                                                    02 DD 001D1      PUSHL   #2                                      ; 0537
                                            04  AC  DD 001D3         PUSHL   VOLB                                    ;
                                0000V       CF      02 FB 001D6      CALLS   #2, EXCH$RTACP_CONSOLIDATE              ;
                                                    04 001DB         RET                                             ; 0540

; Routine Size:  476 bytes,    Routine Base:  EXCH$RTACP_CODE + 00B2
```

```
;  452          0541  1 GLOBAL ROUTINE exch$rtacp_consolidate (volb : $ref_bblock, clean) =     %SBTTL 'exch$rtacp_consolidate (volb
;  453          0542  2 BEGIN
;  454          0543  2 !++
;  455          0544  2 !
;  456          0545  2 ! FUNCTIONAL DESCRIPTION:
;  457          0546  2 !
;  458          0547  2 !       Make a pass to compress the directory.  The following types of entries are unnecessary:
;  459          0548  2 !
;  460          0549  2 !               Multiple consecutive empty entries
;  461          0550  2 !               Tentative entries
;  462          0551  2 !               Empties of length 0 following a permanent entry
;  463          0552  2 !
;  464          0553  2 ! INPUTS:
;  465          0554  2 !
;  466          0555  2 !       volb   - pointer to volb which has been connected to the RT-11 device
;  467          0556  2 !       clean  - flag telling whether to restructure the directory, with the following meanings:
;  468          0557  2 !                       0 - don't force a restructure, but allow it if find segment with single entry
;  469          0558  2 !                       1 - force a restructure regardless
;  470          0559  2 !                       2 - don't allow a restructure
;  471          0560  2 !
;  472          0561  2 ! IMPLICIT INPUTS:
;  473          0562  2 !
;  474          0563  2 !       none
;  475          0564  2 !
;  476          0565  2 ! OUTPUTS:
;  477          0566  2 !
;  478          0567  2 !       none
;  479          0568  2 !
;  480          0569  2 ! IMPLICIT OUTPUTS:
;  481          0570  2 !
;  482          0571  2 !       none
;  483          0572  2 !
;  484          0573  2 ! ROUTINE VALUE:
;  485          0574  2 !
;  486          0575  2 !       true if succeeded, false or error status if failed
;  487          0576  2 !
;  488          0577  2 ! SIDE EFFECTS:
;  489          0578  2 !
;  490          0579  2 !       error conditions will be signaled
;  491          0580  2 !--
;  492          0581  2
;  493          0582  2 $dbgtrc_prefix ('exch$rtacp_consolidate> ');
```

```
;  495          0583  2 LOCAL
;  496          0584  2     seg : $ref_bblock,                    ! a pointer to the current directory segment
;  497          0585  2     cur : $ref_bblock,                    ! a pointer to the current directory entry
;  498          0586  2     prv : $ref_bblock,                    ! a pointer to the previous directory entry
;  499          0587  2     segment_modified : BITVECTOR [32],    ! bits to say whether we need to write out directory segment
;  500          0588  2     seg_num,                              ! current segment number
;  501          0589  2     end_segment_seen,                     ! we have seen an end of segment marker
;  502          0590  2     segment_with_single_entry_seen,       ! a segment was consolidated to a single entry
;  503          0591  2     ent_len,                              ! length of a single directory entry
;  504          0592  2     ent_cnt                               ! number of entries in the segment
;  505          0593  2     ;
;  506          0594  2
;  507          0595  2 BIND
;  508          0596  2     modified_segments = segment_modified  ! map a longword onto the bitvector
;  509          0597  2     ;
;  510          0598  2
;  511          0599  2 $trace_print_fao ('entry');
;  512          0600  2
;  513          0601  2 $block_check (2, .volb, volb, 551);
;  514          0602  2 $logic_check (4, (exch$rtacp_verify_directory (.volb)), 211);
;  515          0603  2
;  516          0604  2 ! Set up some initial conditions
;  517          0605  2 !
;  518          0606  2 modified_segments = 0;                     ! No directory segments have been modified, clear all bits
;  519          0607  2 ent_cnt = 0;                               ! No entries yet
;  520          0608  2 segment_with_single_entry_seen = false;   ! No segment was consolidated to a single entry
```

```
;  522          0609  2 ! Scan the directory, compressing unnecessary file entries
;  523          0610  2 !
;  524          0611  2 ! Start with the first directory segment
;  525          0612  2 !
;  526          0613  2 seg_num = 1;
;  527          0614  2
;  528          0615  2 ! Loop through all the segments in the directory
;  529          0616  2 !
;  530          0617  2 WHILE .seg_num NEQ 0
;  531          0618  2 DO
;  532          0619  3     BEGIN
;  533          0620  3
;  534          0621  3     ! Get a pointer to the current segment
;  535          0622  3     !
;  536          0623  3     seg = exch$rt11_dirseg_get (.volb, .seg_num);
;  537          0624  3     $logic_check (2, (.seg NEQ 0), 152);
;  538          0625  3     ent_len = rt11ent$k_length + .seg [rt11hdr$w_extra_bytes];
;  539          0626  3
;  540          0627  3     ! Get a pointer to the first directory entry, delete it if it is tentative
;  541          0628  3     !
;  542          0629  3     prv = .seg + rt11hdr$k_length;
;  543          0630  4     IF (.prv [rt11ent$v_type] EQL rt11ent$m_typ_tentative)
;  544          0631  3     THEN
;  545          0632  4         BEGIN
;  546          0633  4         segment_modified [.seg_num] = true;
;  547          0634  4         prv [rt11ent$b_type_byte] = rt11ent$m_typ_empty;        ! Zap whole byte to clear protect bit
;  548          0635  3         END;
;  549          0636  3
;  550          0637  3     ! We've taken care of the first entry, now starting with the second scan the rest of this segment.  If e
;  551          0638  3     ! of the first two entries is the end marker we don't need to scan
;  552          0639  3     !
;  553          0640  3     cur = .prv + .ent_len;                          ! Add the length of one entry
;  554          0641  4     IF  (.prv [rt11ent$v_type] NEQ rt11ent$m_typ_end_segment)
;  555          0642  3       AND
;  556          0643  4         (.cur [rt11ent$v_type] NEQ rt11ent$m_typ_end_segment)
;  557          0644  3     THEN
;  558          0645  4         BEGIN
;  559          0646  4         end_segment_seen = false;
;  560          0647  5         WHILE (.cur LSSU (.seg + rt11$k_dirseglen))
;  561          0648  4         DO
;  562          0649  5             BEGIN
;  563          0650  5
;  564          0651  5             ! If tentative, turn it into an empty before we move forward
;  565          0652  5             !
;  566          0653  6             IF (.cur [rt11ent$v_type] EQL rt11ent$m_typ_tentative)
;  567          0654  5             THEN
;  568          0655  6                 BEGIN
;  569          0656  6                 segment_modified [.seg_num] = true;
;  570          0657  6                 cur [rt11ent$b_type_byte] = rt11ent$m_typ_empty;        ! Zap whole byte to clear protect bi
;  571          0658  5                 END;
;  572          0659  5
;  573          0660  5             ! Advance our count
;  574          0661  5             !
;  575          0662  5             CASE .cur [rt11ent$v_type] FROM 0 TO rt11ent$m_typ_end_segment OF
;  576          0663  5             SET
;  577          0664  5             [rt11ent$m_typ_tentative] :
;  578          0665  5                     $logic_check (0, (false), 148);
```

```
;  579          0666  5
;  580          0667  5                [rt11ent$m_typ_permanent] :
;  581          0668  5                        ;
;  582          0669  5
;  583          0670  5                [rt11ent$m_typ_empty] :
;  584          0671  6                        BEGIN
;  585          0672  6
;  586          0673  6                        ! If the previous entry is permanent and the size of this empty is zero we can get rid o
;  587          0674  6                        ! entry.
;  588          0675  6                        !
;  589          0676  7                        IF  (.prv [rt11ent$v_type] EQL rt11ent$m_typ_permanent)
;  590          0677  6                          AND
;  591          0678  7                            (.cur [rt11ent$w_blocks] EQL 0)
;  592          0679  6                        THEN
;  593          0680  7                            BEGIN
;  594          0681  7                            LOCAL
;  595          0682  7                                sl;                     ! Length of directory to slide down one entry
;  596          0683  7
;  597          0684  7                            ! Slide the rest of the segment down one entry
;  598          0685  7                            !
;  599          0686  7                            sl = rt11$k_dirseglen - .ent_len - (.cur - .seg);       ! Length of segment remainin
;  600          0687  7                            CH$MOVE (.sl, .cur + .ent_len, .cur);
;  601          0688  7                            segment_modified [.seg_num] = true;
;  602          0689  7                            cur = .cur - .ent_len;  ! Back up so that next pass sees the slid down entry
;  603          0690  7                            END
;  604          0691  7
;  605          0692  7                        ! Or if the previous entry is also empty we can combine these two into a single empty en
;  606          0693  7                        !
;  607          0694  7                        ELSE IF (.prv [rt11ent$v_type] EQL rt11ent$m_typ_empty)
;  608          0695  6                        THEN
;  609          0696  7                            BEGIN
;  610          0697  7                            LOCAL
;  611          0698  7                                sl;                     ! Length of directory to slide down one entry
;  612          0699  7
;  613          0700  7                            ! Add the size of this empty to the previous empty, plus count it in the total
;  614          0701  7                            !
;  615          0702  7                            prv [rt11ent$w_blocks] = .prv [rt11ent$w_blocks] + .cur [rt11ent$w_blocks];
;  616          0703  7
;  617          0704  7                            ! Slide the rest of the segment down one entry
;  618          0705  7                            !
;  619          0706  7                            sl = rt11$k_dirseglen - .ent_len - (.cur - .seg);       ! Length of segment remainin
;  620          0707  7                            CH$MOVE (.sl, .cur + .ent_len, .cur);
;  621          0708  7                            segment_modified [.seg_num] = true;                     ! Remember that we have chan
;  622          0709  7                            cur = .cur - .ent_len;  ! Back up so that next pass sees the slid down entry
;  623          0710  6                            END;
;  624          0711  5                        END;
;  625          0712  5
;  626          0713  5                [rt11ent$m_typ_end_segment] :
;  627          0714  6                        BEGIN
;  628          0715  6                        end_segment_seen = true;
;  629          0716  6                        EXITLOOP;
;  630          0717  5                        END;
;  631          0718  5
;  632          0719  5                ! Volumes with corrupted directories should have been write-locked when the directory was verifi
;  633          0720  5                ! during mount.  Therefore we must assume an I/O error or programming error within EXCHANGE has
;  634          0721  5                ! corrupted this directory.
;  635          0722  5                !
```

```
;  636     0723  5              [INRANGE, OUTRANGE] :                                   ! We had better give up before something hap
;  637     0724  6                      BEGIN
;  638     0725  6                      exch$rt11_dirseg_flush (.volb, .modified_segments); ! Write out any changes up til now
;  639     0726  6                      $exch_signal_stop (exch$_rt11_baddirect, 2, .volb [volb$l_vol_ident_len], volb [volb$t_v
;  640     0727  5                      END;
;  641     0728  5
;  642     0729  5              TES;
;  643     0730  5
;  644     0731  5              ! Make the current the previous entry and skip to the next
;  645     0732  5              !
;  646     0733  5              prv = .cur;
;  647     0734  5              cur = .cur + .ent_len;
;  648     0735  5
;  649     0736  4              END;
;  650     0737  4
;  651     0738  4          ! Volumes with corrupted directories should have been write-locked when the directory was verified
;  652     0739  4          ! during mount.  Therefore we must assume an I/O error or programming error within EXCHANGE has
;  653     0740  4          ! corrupted this directory.
;  654     0741  4          !
;  655     0742  5          IF NOT (.end_segment_seen)
;  656     0743  4          THEN
;  657     0744  5              BEGIN
;  658     0745  5              exch$rt11_dirseg_flush (.volb, .modified_segments); ! Write out any changes up til now
;  659     0746  5              $exch_signal_stop (exch$_rt11_baddirect, 2, .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident
;  660     0747  4              END;
;  661     0748  3          END;
;  662     0749  3
;  663     0750  3      ! Count the number of entries.  CUR points to the end marker.
;  664     0751  3      !
;  665     0752  5      (BEGIN LOCAL cnt_this_seg;
;  666     0753  5      cnt_this_seg = (.cur - (.seg + rt11hdr$k_length)) / .ent_len;       ! Divide space used by length of ent
;  667     0754  5      ent_cnt = .ent_cnt + .cnt_this_seg;                                 ! Divide space by length
;  668     0755  5      IF .cnt_this_seg EQL 1                                              ! This segment has a single entry
;  669     0756  5      THEN
;  670     0757  5          segment_with_single_entry_seen = true;
;  671     0758  3      END);
;  672     0759  3      $logic_check (4, (((.cur - (.seg + rt11hdr$k_length)) MOD .ent_len) EQL 0), 150);   ! Better be even
;  673     0760  3
;  674     0761  3      ! Skip to the next segment
;  675     0762  3      !
;  676     0763  3      seg_num = .seg [rt11hdr$w_next_seg];
;  677     0764  3
;  678     0765  2      END;
;  679     0766  2
;  680     0767  2  ! Now that we are done, write out any of the segments that we have modified
;  681     0768  2  !
;  682     0769  2  exch$rt11_dirseg_flush (.volb, .modified_segments);
;  683     0770  2
;  684     0771  2  $logic_check (3, (exch$rtacp_verify_directory (.volb)), 191);
;  685     0772  2
;  686     0773  2  ! Now clean it up if one of the flags is set
;  687     0774  2  !
;  688     0775  3  IF (.clean EQL 1)
;  689     0776  2    OR
;  690     0777  3     (.segment_with_single_entry_seen
;  691     0778  3     AND
;  692     0779  3      .ent_cnt GTRU 1                                 ! Avoid recursive loops when down to one entry
```

```
;  693          0780  3    AND
;  694          0781  3      .clean NEQ 2)                          ! Even then, don't clean if it has been prohibited
;  695          0782  2 THEN
;  696          0783  2     exch$rtacp_clean_directory (.volb, .ent_cnt, .ent_len);
;  697          0784  2
;  698          0785  2 RETURN true;
;  699          0786  1 END;


                                                                .EXTRN  EXCH$_RT11_BADDIRECT
                                                                .EXTRN  LIB$STOP

                                        0FFC 00000              .ENTRY  EXCH$RTACP_CONSOLIDATE, Save R2,R3,R4,R5,-      ; 0541
                                                                        R6,R7,R8,R9,R10,R11
                                    5E          18  C2 00002            SUBL2   #24, SP
                                    59      04  AC  D0 00005            MOVL    VOLB, R9                                ; 0601
                                    52 041B00F3 8F  D0 00009            MOVL    #68878579, R2
                                    51    0227  8F  3C 00010            MOVZWL  #551, R1
                                    50          59  D0 00015            MOVL    R9, R0
                                 00000000G      EF  16 00018            JSB     EXCH$UTIL_BLOCK_CHECK
                                           08   AE  7C 0001E            CLRQ    ENT_CNT                                 ; 0607
                                           04   AE  D4 00021            CLRL    SEGMENT_WITH_SINGLE_ENTRY_SEEN          ; 0608
                                    5B          01  D0 00024            MOVL    #1, SEG_NUM                             ; 0613
                                                03  12 00027 1$:        BNEQ    2$                                      ; 0617
                                              0132  31 00029            BRW     24$
                                        0A00      8F  BB 0002C 2$:      PUSHR   #^M<R9,R11>                             ; 0623
                                 00000000G  EF      02  FB 00030        CALLS   #2, EXCH$RT11_DIRSEG_GET
                                    5A              50  D0 00037        MOVL    R0, SEG
                                                    13  12 0003A        BNEQ    3$                                      ; 0624
                                    7E      98      8F  9A 0003C        MOVZBL  #152, -(SP)
                                                    01  DD 00040        PUSHL   #1
                                         00000000G  8F  DD 00042        PUSHL   #EXCH$_BADLOGIC
                                 00000000G  00          03  FB 00048    CALLS   #3, LIB$STOP
                                    58      06      AA  3C 0004F 3$:    MOVZWL  6(SEG), ENT_LEN                         ; 0625
                                    58              0E  C0 00053        ADDL2   #14, ENT_LEN
                                        14  AE      0A  AA 9E 00056      MOVAB   10(R10), 20(SP)                         ; 0629
                                    57      14  AE  D0 0005B            MOVL    20(SP), PRV
                01      01  A7              04      00  ED 0005F        CMPZV   #0, #4, 1(PRV), #1                      ; 0630
                                                    09  12 00065        BNEQ    5$
                        00      0C  AE          5B  E2 00067            BBSS    SEG_NUM, SEGMENT_MODIFIED, 4$           ; 0633
                                01  A7          02  90 0006C 4$:        MOVB    #2, 1(PRV)                              ; 0634
                            56              57      58  C1 00070 5$:    ADDL3   ENT_LEN, PRV, CUR                       ; 0640
                08      01  A7              04      00  ED 00074        CMPZV   #0, #4, 1(PRV), #8                      ; 0641
                                                    06  13 0007A        BEQL    6$
                08      01  A6              04      00  ED 0007C        CMPZV   #0, #4, 1(CUR), #8                      ; 0643
                                                    03  12 00082 6$:    BNEQ    7$
                                                00BB  31 00084        BRW     22$
                                                    6E  D4 00087 7$:    CLRL    END_SEGMENT_SEEN                        ; 0646
                                    10  AE      CA  9E 00089            MOVAB   1024(R10), T6(SP)                       ; 0647
                                    10  AE      56  D1 0008F 8$:        CMPL    CUR, 16(SP)
                                                7D  1E 00093            BGEQU   18$
                01      01  A6              04      00  ED 00095        CMPZV   #0, #4, 1(CUR), #1                      ; 0653
                                                    09  12 0009B        BNEQ    10$
                        00      0C  AE          5B  E2 0009D            BBSS    SEG_NUM, SEGMENT_MODIFIED, 9$           ; 0656
                                01  A6          02  90 000A2 9$:        MOVB    #2, 1(CUR)                              ; 0657
                7E      01  A6              04      00  EF 000A6 10$:    EXTZV   #0, #4, 1(CUR), -(SP)                   ; 0662
```

```
                08              00              8E  CF 000AC               CASEL   (SP)+, #0, #8                                   ;
        0070            0029            0014    0070    000B0 11$:      .WORD   21$-11$,-                                       ;
        0070            0070            0070    0064    000B8           12$-11$,-                                               ;
                                                005F    000C0           13$-11$,-                                               ;
                                                                        21$-11$,-                                               ;
                                                                        19$-11$,-                                               ;
                                                                        21$-11$,-                                               ;
                                                                        21$-11$,-                                               ;
                                                                        21$-11$,-                                               ;
                                                                        17$-11$                                                 ;
                                                        5C  11  000C2           BRB     21$                                     ; 0725
                                7E      94      8F  9A  000C4 12$:      MOVZBL  #148, -(SP)                                     ; 0665
                                                    01  DD  000C8           PUSHL   #1                                              ;
                                        00000000G   8F  DD  000CA           PUSHL   #EXCH$_BADLOGIC                                 ;
                        00000000G   00              03  FB  000D0           CALLS   #3, LIB$STOP                                    ;
                                                    3B  11  000D7           BRB     19$                                             ;
        04      01  A7              04              00  ED  000D9 13$:      CMPZV   #0, #4, 1(PRV), #4                              ; 0676
                                                    05  12  000DF           BNEQ    14$                                             ;
                                        08      A6  B5  000E1           TSTW    8(CUR)                                          ; 0678
                                                    0D  13  000E4           BEQL    15$                                             ;
        02      01  A7              04              00  ED  000E6 14$:      CMPZV   #0, #4, 1(PRV), #2                              ; 0694
                                                    26  12  000EC           BNEQ    19$                                             ;
                        08      A7      08      A6  A0  000EE           ADDW2   8(CUR), 8(PRV)                                  ; 0702
                50              56              5A  C3  000F3 15$:      SUBL3   SEG, CUR, R0                                    ; 0706
                                50      FC00 C048   9E  000F7           MOVAB   -1024(R0)[ENT_LEN], SL                                  ;
                                50              50  CE  000FD           MNEGL   SL, SL                                                  ;
                66              6846            50  28  00100           MOVC3   SL, (ENT_LEN)[CUR], (CUR)                       ; 0707
                00      0C      AE              5B  E2  00105           BBSS    SEG_NUM, SEGMENT_MODIFIED, 16$                  ; 0708
                                56              58  C2  0010A 16$:      SUBL2   ENT_LEN, CUR                                    ; 0709
                                                05  11  0010D           BRB     19$                                             ; 0662
                                6E              01  D0  0010F 17$:      MOVL    #1, END_SEGMENT_SEEN                            ; 0715
                                                09  11  00112 18$:      BRB     20$                                             ; 0714
                                57              56  D0  00114 19$:      MOVL    CUR, PRV                                        ; 0733
                                56              58  C0  00117           ADDL2   ENT_LEN, CUR                                    ; 0734
                                            FF72    31  0011A           BRW     8$                                              ; 0647
                                22              6E  E8  0011D 20$:      BLBS    END_SEGMENT_SEEN, 22$                           ; 0742
                                        0C      AE  DD  00120 21$:      PUSHL   MODIFIED_SEGMENTS                               ; 0745
                                                59  DD  00123           PUSHL   R9                                              ;
                    00000000G   EF              02  FB  00125           CALLS   #2, EXCH$RT11_DIRSEG_FLUSH                      ;
                                        69      A9  9F  0012C           PUSHAB  105(R9)                                         ; 0746
                                        65      A9  DD  0012F           PUSHL   101(R9)                                         ;
                                                02  DD  00132           PUSHL   #2                                              ;
                                    00000000G   8F  DD  00134           PUSHL   #EXCH$_RT11_BADDIRECT                           ;
                    00000000G   00              04  FB  0013A           CALLS   #4, LIB$STOP                                    ;
                                                04  00141             RET                                                     ;
        50              56      14      AE  C3  00142 22$:      SUBL3   20(SP), CUR, R0                                 ; 0753
                                50              58  C6  00147           DIVL2   ENT_LEN, CNT_THIS_SEG                           ;
                08      AE              50  C0  0014A           ADDL2   CNT_THIS_SEG, ENT_CNT                           ; 0754
                        01              50  D1  0014E           CMPL    CNT_THIS_SEG, #1                                ; 0755
                                            04  12  00151           BNEQ    23$                                             ;
                04      AE              01  D0  00153           MOVL    #1, SEGMENT_WITH_SINGLE_ENTRY_SEEN              ; 0757
                        5B      02      AA  3C  00157 23$:      MOVZWL  2(SEG), SEG_NUM                                 ; 0763
                                        FEC9    31  0015B           BRW     1$                                              ; 0617
                                        0C      AE  DD  0015E 24$:      PUSHL   MODIFIED_SEGMENTS                               ; 0769
                                                59  DD  00161           PUSHL   R9                                              ;
                    00000000G   EF              02  FB  00163           CALLS   #2, EXCH$RT11_DIRSEG_FLUSH                      ;
                                01      08      AC  D1  0016A           CMPL    CLEAN, #1                                       ; 0775
```

```
                         10  13 0016E          BEQL    25$
        18      04  AE   E9 00170          BLBC    SEGMENT_WITH_SINGLE_ENTRY_SEEN, 26$     ; 0777
        01      08  AE   D1 00174          CMPL    ENT_CNT, #1                             ; 0779
                         12  1B 00178          BLEQU   26$
        02      08  AC   D1 0017A          CMPL    CLEAN, #2                               ; 0781
                         0C  13 0017E          BEQL    26$
                         58  DD 00180 25$:     PUSHL   ENT_LEN                                 ; 0783
                0C  AE   DD 00182          PUSHL   ENT_CNT
                         59  DD 00185          PUSHL   R9
   FC98 CF               03  FB 00187          CALLS   #3, EXCH$RTACP_CLEAN_DIRECTORY
        50               01  D0 0018C 26$:     MOVL    #1, R0                                  ; 0785
                             04 0018F          RET                                             ; 0786

; Routine Size:  400 bytes,    Routine Base:  EXCH$RTACP_CODE + 028E
```

```
;  701     0787  1 GLOBAL ROUTINE exch$rtacp_find_empty_area (ctx : $ref_bblock,   %SBTTL 'exch$rtacp_find_empty_area (ctx, blo
;  702     0788  1                                                 blocks, start) =
;  703     0789  2 BEGIN
;  704     0790  2 !++
;  705     0791  2 !
;  706     0792  2 ! FUNCTIONAL DESCRIPTION:
;  707     0793  2 !
;  708     0794  2 !       First, have EXCH$RT11_CONSOLIDATE make a pass to compress the directory.
;  709     0795  2 !
;  710     0796  2 !       Second, make a pass to prepare to enter a new file.  The BLOCKS parameter gives the size of the area
;  711     0797  2 !       is needed.  We will return the address of the first empty area which is as close to this size as we
;  712     0798  2 !       get.  We will also put a zero-length empty area after this area so that the rt-11 close routine can
;  713     0799  2 !       move any excess blocks into the second entry.
;  714     0800  2 !
;  715     0801  2 ! INPUTS:
;  716     0802  2 !
;  717     0803  2 !       ctx    - pointer to rt11 context block
;  718     0804  2 !       blocks - number of blocks needed, 0 means largest possible entry
;  719     0805  2 !       start  - pbn of block to start allocation (blocks parameter ignored if start <> 0)
;  720     0806  2 !       (dummy) - this routine will call itself a second time if it is unable to satisfy the request.  Recur
;  721     0807  2 !                 calls are flagged with a 4th dummy parameter, ACTUALCOUNT () is examined to prevent loops
;  722     0808  2 !
;  723     0809  2 ! IMPLICIT INPUTS:
;  724     0810  2 !
;  725     0811  2 !       none
;  726     0812  2 !
;  727     0813  2 ! OUTPUTS:
;  728     0814  2 !
;  729     0815  2 !       none
;  730     0816  2 !
;  731     0817  2 ! IMPLICIT OUTPUTS:
;  732     0818  2 !
;  733     0819  2 !       none
;  734     0820  2 !
;  735     0821  2 ! ROUTINE VALUE:
;  736     0822  2 !
;  737     0823  2 !       true if able to find a suitable empty area, false or error status if failed
;  738     0824  2 !
;  739     0825  2 ! SIDE EFFECTS:
;  740     0826  2 !
;  741     0827  2 !       error conditions will be signaled
;  742     0828  2 !--
;  743     0829  2
;  744     0830  2 $dbgtrc_prefix ('exch$rtacp_find_empty_area> ');
;  745     0831  2
;  746     0832  2 LOCAL
;  747     0833  2     seg : $ref_bblock,                  ! a pointer to the current directory segment
;  748     0834  2     cur : $ref_bblock,                  ! a pointer to the current directory entry
;  749     0835  2     exact_match,                        ! found an entry of exactly the correct length
;  750     0836  2     ent_len,                            ! length of a single directory entry
;  751     0837  2     flags,                              ! a temporary for the flags mask
;  752     0838  2     mat_ent : $ref_bblock,              ! address of current match
;  753     0839  2     mat_blk,                            ! start pbn of current match
;  754     0840  2     mat_len,                            ! size of current match
;  755     0841  2     mat_seg,                            ! segment number containing the match
;  756     0842  2     recursive,                          ! flag that we have been called from here
;  757     0843  2     status
```

```
;  758          0844  2     ;
;  759          0845  2
;  760          0846  2 BIND
;  761          0847  2     volb = ctx [rt11ctx$a_assoc_volb] : $ref_bblock
;  762          0848  2     ;
;  763          0849  2
;  764          0850  2 BUILTIN
;  765          0851  2     ACTUALCOUNT
;  766          0852  2     ;
;  767          0853  2
;  768          0854  2
;  769          0855  2 $trace_print_fao ('entry - ctx !XL,  blocks !UL,  start !UL', .ctx, .blocks, .start);
;  770          0856  2
;  771          0857  2 $block_check (2, .ctx, rt11ctx, 535);
;  772          0858  2 $block_check (2, .volb, volb, 530);
;  773          0859  2
;  774          0860  2 ! Set up some initial conditions
;  775          0861  2 !
;  776          0862  2 ctx [rt11ctx$a_ent_address] = 0;                ! Set the output to an invalid address
;  777          0863  2 ctx [rt11ctx$l_seg_number] = 0;                 ! Reset to scan from the start of the directory
;  778          0864  2 mat_len = 0;                                     ! No length of any matched entry
;  779          0865  2 recursive = (IF ACTUALCOUNT () EQL 4 THEN 1 ELSE 0);
;  780          0866  2
;  781          0867  2 ! Compress the directory into a known, clean state.  Consolidate will do any verification we need.  The seco
;  782          0868  2 ! parameter determines whether a directory restructuring should take place.
;  783          0869  2 !
;  784          0870  3 IF NOT (status = exch$rtacp_consolidate (.volb, .recursive))
;  785          0871  2 THEN
;  786          0872  2     RETURN .status;
;  787          0873  2
;  788          0874  2 ! Loop through all the segments in the directory
;  789          0875  2 !
;  790          0876  2 exact_match = false;                            ! We haven't seen an empty entry of exactly the correct size
;  791          0877  2 flags = rtnxt$m_empty OR rtnxt$m_skip_check OR rtnxt$m_skip_expand;
;  792          0878  3 WHILE ((cur = exch$rtacp_next_entry (.ctx, .flags)) NEQ 0)
;  793          0879  2 DO
;  794          0880  3     BEGIN
;  795          0881  3     LOCAL
;  796          0882  3         len;
;  797          0883  3
;  798          0884  3     len = .cur [rt11ent$w_blocks];      ! Put the length into a local for speed
;  799          0885  3
;  800          0886  3     ! If we are looking for a particular start block, see if this empty area contains the start
;  801          0887  3     !
;  802          0888  3     IF .start NEQ 0
;  803          0889  3     THEN
;  804          0890  4         BEGIN
;  805          0891  4         IF .start LSSU .ctx [rt11ctx$l_start_block]
;  806          0892  4         THEN
;  807          0893  4             RETURN exch$_stnotavail                 ! Already past the block, we will never find it
;  808          0894  4         ELSE
;  809          0895  5             BEGIN
;  810          0896  6             IF .start LEQU (.len + .ctx [rt11ctx$l_start_block])        ! Requested block inside this area
;  811          0897  5             THEN
;  812          0898  6                 BEGIN
;  813          0899  6                 exact_match = true;             ! Set the flag to stop scanning
;  814          0900  6
```

```
;  815     0901  6                ! Save the entry as a match.  Note that this and the following three saves are written
;  816     0902  6                ! so that the optimizer can collapse them into a single segment of object code.
;  817     0903  6                !
;  818     0904  6                mat_len = .len;                             ! Save the length of this entry,
;  819     0905  6                mat_ent = .cur;                             !  its address,
;  820     0906  6                mat_seg = .ctx [rt11ctx$l_seg_number];      !  its segment number,
;  821     0907  6                mat_blk = .ctx [rt11ctx$l_start_block];     !  and the pbn where the free space starts
;  822     0908  5                END;
;  823     0909  4            END;
;  824     0910  4        END
;  825     0911  4
;  826     0912  4    ! We are looking for any old empty area, see if the size is appropriate
;  827     0913  4    !
;  828     0914  3    ELSE IF .len GEQU .blocks           ! If this length is a candidate, then examine it more closely
;  829     0915  3    THEN
;  830     0916  4        BEGIN
;  831     0917  4
;  832     0918  4        IF .blocks EQL 0                ! Block=0 means look for the biggest free space
;  833     0919  4        THEN
;  834     0920  5            BEGIN
;  835     0921  5            IF .len GTRU .mat_len               ! Bigger than what we've seen so far
;  836     0922  5            THEN
;  837     0923  6                BEGIN
;  838     0924  6
;  839     0925  6                ! Save the entry as a match.  Note that all 4 saves can be collapsed by the optimizer.
;  840     0926  6                !
;  841     0927  6                mat_len = .len;                             ! Save the length of this entry,
;  842     0928  6                mat_ent = .cur;                             !  its address,
;  843     0929  6                mat_seg = .ctx [rt11ctx$l_seg_number];      !  its segment number,
;  844     0930  6                mat_blk = .ctx [rt11ctx$l_start_block];     !  and the pbn where the free space starts
;  845     0931  5                END;
;  846     0932  5            END
;  847     0933  4        ELSE                    ! Block<>0 means look for a close match
;  848     0934  5            BEGIN
;  849     0935  5            IF .len EQL .blocks         ! Exactly what we are looking for
;  850     0936  5            THEN
;  851     0937  6                BEGIN
;  852     0938  6
;  853     0939  6                ! Save the entry as a match.  Note that all 4 saves can be collapsed by the optimizer.
;  854     0940  6                !
;  855     0941  6                exact_match = true;                         ! Set the flag to stop scanning
;  856     0942  6                mat_len = .len;                             ! Save the length of this entry,
;  857     0943  6                mat_ent = .cur;                             !  its address,
;  858     0944  6                mat_seg = .ctx [rt11ctx$l_seg_number];      !  its segment number,
;  859     0945  6                mat_blk = .ctx [rt11ctx$l_start_block];     !  and the pbn where the free space starts
;  860     0946  6                END
;  861     0947  7            ELSE IF ((.len LSSU .mat_len)       ! Closer to correct than the previous match
;  862     0948  6                    OR (.mat_len EQL 0))        !  or no previous match
;  863     0949  5            THEN
;  864     0950  6                BEGIN
;  865     0951  6
;  866     0952  6                ! Save the entry as a match.  Note that all 4 saves can be collapsed by the optimizer.
;  867     0953  6                !
;  868     0954  6                mat_len = .len;                             ! Save the length of this entry,
;  869     0955  6                mat_ent = .cur;                             !  its address,
;  870     0956  6                mat_seg = .ctx [rt11ctx$l_seg_number];      !  its segment number,
;  871     0957  6                mat_blk = .ctx [rt11ctx$l_start_block];     !  and the pbn where the free space starts
```

```
;  872     0958  5                    END;
;  873     0959  4                END;
;  874     0960  3            END;
;  875     0961
;  876     0962  3            ! If we have seen an exact match we can stop scanning this segment
;  877     0963  3            !
;  878     0964  3            IF .exact_match THEN EXITLOOP;
;  879     0965  3
;  880     0966  2        END;
```

```
;  882     0967  2 ! Now put a zero-length entry after the matched entry and set the return values
;  883     0968  2 !
;  884     0969  2 IF .mat_len NEQ 0                                       ! Means that we found one
;  885     0970  2 THEN
;  886     0971  2     BEGIN
;  887     0972  2     LOCAL
;  888     0973  2         eos : $ref_bblock,                              ! End of segment entry
;  889     0974  2         emp : $ref_bblock,                              ! Entry to be turned into a zero-length empty
;  890     0975  2         sl;                                             ! Length of directory to slide up one entry
;  891     0976  2
;  892     0977  2     ! Get a pointer to the segment containing the match, return if error
;  893     0978  2     !
;  894     0979  2     seg = exch$rt11_dirseg_get (.volb, .mat_seg);
;  895     0980  2     $logic_check (2, (.seg NEQ 0), 155);
;  896     0981  2     ent_len = rt11ent$k_length + .seg [rt11hdr$w_extra_bytes];
;  897     0982  2
;  898     0983  2     ! Find the end of the segment.  We verified that one exists in the first scan.
;  899     0984  2     !
;  900     0985  2     eos = .mat_ent;                                     ! Point to the matched entry
;  901     0986  2     WHILE 1
;  902     0987  2     DO
;  903     0988  4         BEGIN
;  904     0989  4         eos = .eos + .ent_len;                          ! Advance to the next entry
;  905     0990  4         $logic_check (1, (.eos LSSU (.seg + rt11$k_dirseglen)), 144);   ! It can't loop forever, can it?
;  906     0991  4         IF .eos [rt11ent$v_type] EQL rt11ent$m_typ_end_segment
;  907     0992  4         THEN
;  908     0993  4             EXITLOOP;
;  909     0994  4         END;
;  910     0995  3
;  911     0996  3     ! If we are looking for a specific block, we might need to split the current empty into two entries, the
;  912     0997  3     ! second of which will start on the requested block.
;  913     0998  3     !
;  914     0999  3     IF .start NEQ 0                                     ! /START_BLOCK has been requested
;  915     1000  3       AND
;  916     1001  3        .start NEQ .mat_blk                              ! We are lucky, it already begins on exactly the right block
;  917     1002  3     THEN
;  918     1003  4         BEGIN
;  919     1004  4
;  920     1005  4         ! Make sure that there is room to add two more entries to this segment.  If not, signal a full direc
;  921     1006  4         !
;  922     1007  5         IF ((.eos+2 + (2*.ent_len)) GEQU (.seg + rt11$k_dirseglen))
;  923     1008  4         THEN
;  924     1009  5             BEGIN
;  925     1010  5             IF .recursive                                                   ! If we have already tried it a seco
;  926     1011  5             THEN                                                            !  then cleaning didn't help and we
;  927     1012  5                 RETURN exch$_rt11_overflow                                  !  return an error.  Otherwise, call
;  928     1013  5             ELSE                                                            !  routine again with a dummy parame
;  929     1014  5                 RETURN exch$rtacp_find_empty_area (.ctx, .blocks, .start, 0); !  flag the call as recursive
;  930     1015  4             END;
;  931     1016  4
;  932     1017  4         ! Slide the rest of the segment up one entry and split the empty entry
;  933     1018  4         !
;  934     1019  4         emp = .mat_ent;                                                     ! Point to the matched entry (will become an
;  935     1020  4         sl = .eos+2 - .emp;                                                 ! Length of segment between empty and end
;  936     1021  4         CH$MOVE (.sl, .emp, .emp + .ent_len);                               ! Slide rest of segment up
;  937     1022  4         mat_ent = .mat_ent + .ent_len;                                      ! Move the pointer to the matched entry
;  938     1023  4         eos = .eos + .ent_len;                                              ! Move the pointer to the end of the segment
```

```
;  939          1024  4          CH$FILL (0, rt11ent$k_length, .emp);                                ! Set all fields of empty entry to null
;  940          1025  4          emp [rt11ent$b_type_byte] = rt11ent$m_typ_empty;                    ! Set to an empty entry
;  941          1026  4          sl = .start - .mat_blk;                                             ! Get number of blocks to put in the empty
;  942          1027  4          emp [rt11ent$w_blocks] = .sl;                                       ! Set the size of the new empty
;  943          1028  4          mat_ent [rt11ent$w_blocks] = .mat_ent [rt11ent$w_blocks] - .sl; ! Subtract the blocks from the match
;  944          1029  4          mat_blk = .mat_blk + .sl;                                           ! Move the starting block
;  945          1030  4
;  946          1031  3          END;
;  947          1032  3
;  948          1033  3      ! Make sure that there is room to add one more entry to this segment.  If not, signal a full directory
;  949          1034  3      !
;  950          1035  4      IF ((.eos+2 + .ent_len) GEQU (.seg + rt11$k_dirseglen))
;  951          1036  3      THEN
;  952          1037  4          BEGIN
;  953          1038  4          IF .recursive                                                       ! If we have already tried it a seco
;  954          1039  4          THEN                                                                !  then cleaning didn't help and we
;  955          1040  4              status = exch$_rt11_overflow                                    !  return an error.  Otherwise, call
;  956          1041  4          ELSE                                                                !  routine again with a dummy parame
;  957          1042  4              RETURN exch$rtacp_find_empty_area (.ctx, .blocks, .start, 0); !  flag the call as recursive
;  958          1043  4          END
;  959          1044  4
;  960          1045  3      ELSE
;  961          1046  4          BEGIN
;  962          1047  4
;  963          1048  4          ! Slide the rest of the segment up one entry and make the empty entry
;  964          1049  4          !
;  965          1050  4          emp = .mat_ent + .ent_len;                                          ! Point to one past the matched entry
;  966          1051  4          sl = .eos+2 - .emp;                                                 ! Length of segment between empty and end
;  967          1052  4          CH$MOVE (.sl, .emp, .emp + .ent_len);                               ! Slide rest of segment up
;  968          1053  4          CH$FILL (0, rt11ent$k_length, .emp);                                ! Set all fields of entry to null
;  969          1054  4          emp [rt11ent$b_type_byte] = rt11ent$m_typ_empty;                    ! Zap whole byte to clear protect bit
;  970          1055  4
;  971          1056  4          ! Put the info in the context block
;  972          1057  4          !
;  973          1058  4          ctx [rt11ctx$l_start_block] = .mat_blk;                             ! Copy the pbn where the file starts
;  974          1059  4          ctx [rt11ctx$l_seg_number]  = .mat_seg;                             ! The segment number
;  975          1060  4          ctx [rt11ctx$a_seg_address] = .seg;                                 ! The address of the start of the se
;  976          1061  4          ctx [rt11ctx$a_ent_address] = .mat_ent;                             ! The address of the entry
;  977          1062  4          CH$MOVE (rt11ctx$s_entry, .mat_ent, ctx [rt11ctx$t_entry]);         ! And the entry itself
;  978          1063  4
;  979          1064  4          exch$rt11_dirseg_put (.volb, .mat_seg);                     ! Save the changes
;  980          1065  4          status = true;
;  981          1066  3          END;
;  982          1067  3      END
;  983          1068  3
;  984          1069  3  ! Either no free space at all (.blocks = 0) or none as large as requested
;  985          1070  3  !
;  986          1071  2  ELSE
;  987          1072  3      BEGIN
;  988          1073  3      IF .start NEQ 0                                     ! If specific start block was requested, then it was past th
;  989          1074  3      THEN                                                !  end of the volume (or contained in last entry in director
;  990          1075  3          status = exch$_stnotavail
;  991          1076  3      ELSE IF .blocks EQL 0                               ! If blocks requested is 0, then there was no space anyplace
;  992          1077  3      THEN                                                !  we should return a volume full error
;  993          1078  3          status = exch$_volume_full
;  994          1079  3      ELSE
;  995          1080  4          BEGIN
```

```
;  996     1081  4          IF .recursive                                                    ! If we have already tried it a seco
;  997     1082  4          THEN                                                             !  then cleaning didn't help and we
;  998     1083  4              status = exch$_volume_full                                   !  return an error.  Otherwise, call
;  999     1084  4          ELSE                                                             !  routine again with a dummy parame
; 1000     1085  4              RETURN exch$rtacp_find_empty_area (.ctx, .blocks, .start, 0);!  flag the call as recursive
; 1001     1086  3          END;
; 1002     1087  2      END;
; 1003     1088  2
; 1004     1089  2 $logic_check (2, (exch$rtacp_verify_directory (.volb)), 192);
; 1005     1090  2
; 1006     1091  2 RETURN .status;
; 1007     1092  1 END;


                                                        .EXTRN  EXCH$_STNOTAVAIL
                                                        .EXTRN  EXCH$_RT11_OVERFLOW
                                                        .EXTRN  EXCH$_VOLUME_FULL

                                    0FFC 00000          .ENTRY  EXCH$RTACP_FIND_EMPTY_AREA, Save R2,R3,R4,- ; 0787
                                                                R5,R6,R7,R8,R9,R10,R11                      ;
                              5E            20  C2 00002  SUBL2   #32, SP                                    ;
                              59        04  AC  D0 00005  MOVL    CTX, R9                                     ; 0847
                              52 008200F4   8F  D0 00009  MOVL    #8519924, R2                                ; 0857
                              51      0217  8F  3C 00010  MOVZWL  #535, R1                                    ;
                              50            59  D0 00015  MOVL    R9, R0                                      ;
                                 00000000G  EF  16 00018  JSB     EXCH$UTIL_BLOCK_CHECK                       ;
                           04 AE        14  A9  D0 0001E  MOVL    20(R9), 4(SP)                               ; 0858
                              52 041B00F3   8F  D0 00023  MOVL    #68878579, R2                               ;
                              51      0212  8F  3C 0002A  MOVZWL  #530, R1                                    ;
                              50        04  AE  D0 0002F  MOVL    4(SP), R0                                   ;
                                 00000000G  EF  16 00033  JSB     EXCH$UTIL_BLOCK_CHECK                       ;
                                       7E  A9  D4 00039  CLRL    126(R9)                                     ; 0862
                           6E          76  A9  9E 0003C  MOVAB   118(R9), (SP)                              ; 0863
                                       00  BE  D4 00040  CLRL    @0(SP)                                      ;
                                           53  D4 00043  CLRL    MAT_LEN                                     ; 0864
                           04              6C  91 00045  CMPB    (AP), #4                                    ; 0865
                                           06  12 00048  BNEQ    1$                                          ;
                        1C AE              01  D0 0004A  MOVL    #1, RECURSIVE                               ;
                                           03  11 0004E  BRB     2$                                          ;
                                    1C  AE  D4 00050 1$:  CLRL    RECURSIVE                                   ;
                                    1C  AE  DD 00053 2$:  PUSHL   RECURSIVE                                   ; 0870
                                    08  AE  DD 00056      PUSHL   8(SP)                                       ;
                     FE12           CF      02  FB 00059      CALLS   #2, EXCH$RTACP_CONSOLIDATE              ;
                        0C          AE      50  D0 0005E      MOVL    R0, STATUS                              ;
                                    03      0C  AE E8 00062      BLBS    STATUS, 3$                          ;
                                            01DA 31 00066      BRW     26$                                 ;
                                           54  D4 00069 3$:  CLRL    EXACT_MATCH                             ; 0876
                                    55          32  D0 0006B      MOVL    #50, FLAGS                          ; 0877
                                                55  DD 0006E 4$:  PUSHL   FLAGS                               ; 0878
                                                59  DD 00070      PUSHL   R9                                  ;
                     0000V          CF          02  FB 00072      CALLS   #2, EXCH$RTACP_NEXT_ENTRY           ;
                                                50  D5 00077      TSTL    CUR                                 ;
                                                5E  13 00079      BEQL    12$                                 ;
                                    51      08  A0  3C 0007B      MOVZWL  8(CUR), LEN                         ; 0884
                                            0C  AC  D5 0007F      TSTL    START                               ; 0888
                                                1C  13 00082      BEQL    6$                                  ;
```

```
                  72   A9         0C   AC  D1 00084          CMPL    START, 114(R9)                          ; 0891
                                       08  1E 00089          BGEQU   5$
                       50  00000000G   8F  D0 0008B          MOVL    #EXCH$_STNOTAVAIL, R0                   ; 0893
                                       04  00092             RET
          52                51    72   A9  C1 00093 5$:      ADDL3   114(R9), LEN, R2                        ; 0896
                            52    0C   AC  D1 00098          CMPL    START, R2
                                       38  1A 0009C          BGTRU   11$
                                       18  11 0009E          BRB     8$                                      ; 0899
                  08   AC              51  D1 000A0 6$:      CMPL    LEN, BLOCKS                             ; 0914
                                       30  1F 000A4          BLSSU   11$
                             08        AC  D5 000A6          TSTL    BLOCKS                                  ; 0918
                                       07  12 000A9          BNEQ    7$
                       53              51  D1 000AB          CMPL    LEN, MAT_LEN                            ; 0921
                                       26  1B 000AE          BLEQU   11$
                                       14  11 000B0          BRB     10$                                     ; 0927
                  08   AC              51  D1 000B2 7$:      CMPL    LEN, BLOCKS                             ; 0935
                                       05  12 000B6          BNEQ    9$
                       54              01  D0 000B8 8$:      MOVL    #1, EXACT_MATCH                         ; 0941
                                       09  11 000BB          BRB     10$                                     ; 0942
                       53              51  D1 000BD 9$:      CMPL    LEN, MAT_LEN                            ; 0947
                                       04  1F 000C0          BLSSU   10$
                                       53  D5 000C2          TSTL    MAT_LEN                                 ; 0948
                                       10  12 000C4          BNEQ    11$
                                 53    51  D0 000C6 10$:     MOVL    LEN, MAT_LEN                            ; 0954
                                 5B    50  D0 000C9          MOVL    CUR, MAT_ENT                            ; 0955
                       14   AE    00   BE  D0 000CC          MOVL    @0(SP), MAT_SEG                         ; 0956
                       10   AE    72   A9  D0 000D1          MOVL    114(R9), MAT_BLK                        ; 0957
                                 95    54  E9 000D6 11$:     BLBC    EXACT_MATCH, 4$                         ; 0964
                                       53  D5 000D9 12$:     TSTL    MAT_LEN                                 ; 0969
                                       03  12 000DB          BNEQ    13$
                                     0115  31 000DD          BRW     21$
                                 14    AE  DD 000E0 13$:     PUSHL   MAT_SEG                                 ; 0979
                                 08    AE  DD 000E3          PUSHL   8(SP)
                  00000000G   EF       02  FB 000E6          CALLS   #2, EXCH$RT11_DIRSEG_GET
                         18   AE       50  D0 000ED          MOVL    R0, SEG
                                       13  12 000F1          BNEQ    14$                                     ; 0980
                       7E         9B   8F  9A 000F3          MOVZBL  #155, -(SP)
                                       01  DD 000F7          PUSHL   #1
                           00000000G   8F  DD 000F9          PUSHL   #EXCH$_BADLOGIC
                  00000000G   00       03  FB 000FF          CALLS   #3, LIB$STOP
          50            18    AE       06  C1 00106 14$:     ADDL3   #6, SEG, R0                             ; 0981
                              5A       60  3C 0010B          MOVZWL  (R0), ENT_LEN
                              5A       0E  C0 0010E          ADDL2   #14, ENT_LEN
                              56       5B  D0 00111          MOVL    MAT_ENT, EOS                            ; 0985
          08   AE       18    AE  00000400 8F C1 00114       ADDL3   #1024, SEG, 8(SP)                       ; 0990
                              56       5A  C0 0011E 15$:     ADDL2   ENT_LEN, EOS                            ; 0989
                  08   AE              56  D1 00121          CMPL    EOS, 8(SP)                              ; 0990
                                       13  1F 00125          BLSSU   16$
                       7E         90   8F  9A 00127          MOVZBL  #144, -(SP)
                                       01  DD 0012B          PUSHL   #1
                           00000000G   8F  DD 0012D          PUSHL   #EXCH$_BADLOGIC
                  00000000G   00       03  FB 00133          CALLS   #3, LIB$STOP
 08       01   A6             04       00  ED 0013A 16$:     CMPZV   #0, #4, 1(EOS), #8                      ; 0991
                                       DC  12 00140          BNEQ    15$
                             0C        AC  D5 00142          TSTL    START                                   ; 0999
                                       50  13 00145          BEQL    18$
                  10   AE         0C   AC  D1 00147          CMPL    START, MAT_BLK                          ; 1001
```

```
                         49  13 0014C         BEQL    18$                                    ;
              50      02 A64A  3E 0014E         MOVAW   2(EOS)[ENT_LEN], R0                    ; 1007
          08  AE          50  D1 00153         CMPL    R0, 8(SP)                              ;
                          0C  1F 00157         BLSSU   17$                                    ;
          45  1C          AE  E9 00159         BLBC    RECURSIVE, 19$                         ; 1010
          50 00000000G    8F  D0 0015D         MOVL    #EXCH$_RT11_OVERFLOW, R0               ; 1014
                          04  00164            RET                                            ;
                  58      5B  D0 00165 17$:    MOVL    MAT_ENT, EMP                           ; 1019
      50          56      58  C3 00168         SUBL3   EMP, EOS, R0                           ; 1020
                  57  02  A0  9E 0016C         MOVAB   2(R0), SL                              ;
    6A48          68      57  28 00170         MOVC3   SL, (EMP), (ENT_LEN)[EMP]              ; 1021
                  5B      5A  C0 00175         ADDL2   ENT_LEN, MAT_ENT                       ; 1022
                  56      5A  C0 00178         ADDL2   ENT_LEN, EOS                           ; 1023
 0E       00      6E      00  2C 0017B         MOVC5   #0, (SP), #0, #14, (EMP)               ; 1024
                          68     00180                                                        ;
          01  A8          02  90 00181         MOVB    #2, 1(EMP)                             ; 1025
      57  0C  AC  10      AE  C3 00185         SUBL3   MAT_BLK, START, SL                     ; 1026
          08  A8          57  B0 0018B         MOVW    SL, 8(EMP)                             ; 1027
          08  AB          57  A2 0018F         SUBW2   SL, 8(MAT_ENT)                         ; 1028
          10  AE          57  C0 00193         ADDL2   SL, MAT_BLK                            ; 1029
              50      02 AA46  9E 00197 18$:   MOVAB   2(ENT_LEN)[EOS], R0                    ; 1035
          08  AE          50  D1 0019C         CMPL    R0, 8(SP)                              ;
                          0E  1F 001A0         BLSSU   20$                                    ;
          71  1C          AE  E9 001A2 19$:    BLBC    RECURSIVE, 24$                         ; 1038
          0C  AE 00000000G    8F  D0 001A6         MOVL    #EXCH$_RT11_OVERFLOW, STATUS           ; 1040
                          75  11 001AE         BRB     25$                                    ;
      58          5B      5A  C1 001B0 20$:    ADDL3   ENT_LEN, MAT_ENT, EMP                  ; 1050
                  56      58  C2 001B4         SUBL2   EMP, R6                                ; 1051
                  57  02  A6  9E 001B7         MOVAB   2(R6), SL                              ;
    6A48          68      57  28 001BB         MOVC3   SL, (EMP), (ENT_LEN)[EMP]              ; 1052
 0E       00      6E      00  2C 001C0         MOVC5   #0, (SP), #0, #14, (EMP)               ; 1053
                          68     001C5                                                        ;
          01  A8          02  90 001C6         MOVB    #2, 1(EMP)                             ; 1054
          72  A9  10      AE  D0 001CA         MOVL    MAT_BLK, 114(R9)                       ; 1058
          00  BE  14      AE  D0 001CF         MOVL    MAT_SEG, @0(SP)                        ; 1059
          7A  A9  18      AE  D0 001D4         MOVL    SEG, 122(R9)                           ; 1060
          7E  A9          5B  D0 001D9         MOVL    MAT_ENT, 126(R9)                       ; 1061
     38   A9      6B      0E  28 001DD         MOVC3   #14, (MAT_ENT), 56(R9)                 ; 1062
                  14      AE  DD 001E2         PUSHL   MAT_SEG                                ; 1064
                  08      AE  DD 001E5         PUSHL   8(SP)                                  ;
   00000000G  EF          02  FB 001E8         CALLS   #2, EXCH$RT11_DIRSEG_PUT               ;
          0C  AE          01  D0 001EF         MOVL    #1, STATUS                             ; 1065
                          30  11 001F3         BRB     25$                                    ; 0969
                  0C      AC  D5 001F5 21$:    TSTL    START                                  ; 1073
                          0A  13 001F8         BEQL    22$                                    ;
          0C  AE 00000000G    8F  D0 001FA         MOVL    #EXCH$_STNOTAVAIL, STATUS              ; 1075
                          21  11 00202         BRB     25$                                    ;
                  08      AC  D5 00204 22$:    TSTL    BLOCKS                                 ; 1076
                          04  13 00207         BEQL    23$                                    ;
          0A  1C          AE  E9 00209         BLBC    RECURSIVE, 24$                         ; 1081
          0C  AE 00000000G    8F  D0 0020D 23$:    MOVL    #EXCH$_VOLUME_FULL, STATUS             ; 1083
                          0E  11 00215         BRB     25$                                    ;
                          7E  D4 00217 24$:    CLRL    -(SP)                                  ; 1085
          7E  08          AC  7D 00219         MOVQ    BLOCKS, -(SP)                          ;
                          59  DD 0021D         PUSHL   R9                                     ;
        FDDC  CF          04  FB 0021F         CALLS   #4, EXCH$RTACP_FIND_EMPTY_AREA         ;
                          04  00224            RET                                            ;
```

```
                                04   AE  DD 00225 25$:    PUSHL   4(SP)                                          ; 1089
                      0000V  CF        01  FB 00228       CALLS   #1, EXCH$RTACP_VERIFY_DIRECTORY                ;
                             13        50  E8 0022D       BLBS    R0, 26$                                        ;
                             7E   C0   8F  9A 00230       MOVZBL  #192, -(SP)                                    ;
                                       01  DD 00234       PUSHL   #1                                             ;
                                00000000G 8F  DD 00236    PUSHL   #EXCH$_BADLOGIC                                ;
                          00000000G  00       03  FB 0023C    CALLS   #3, LIB$STOP                               ;
                             50        0C   AE  D0 00243 26$:    MOVL    STATUS, R0                             ; 1091
                                            04  00247       RET                                                    ; 1092

; Routine Size:  584 bytes,    Routine Base:  EXCH$RTACP_CODE + 041E
```

```
;  1009          1093  1 GLOBAL ROUTINE exch$rtacp_find_file (ctx : $ref_bblock, name, name_len) =       %SBTTL 'exch$rtacp_find_file
;  1010          1094  2 BEGIN
;  1011          1095  2 !++
;  1012          1096  2 !
;  1013          1097  2 ! FUNCTIONAL DESCRIPTION:
;  1014          1098  2 !
;  1015          1099  2 !       Search an RT-11 directory for a specific file
;  1016          1100  2 !
;  1017          1101  2 ! INPUT:
;  1018          1102  2 !
;  1019          1103  2 !       ctx      - pointer to an RT11CTX structure
;  1020          1104  2 !       name     - pointer to the start of the file name for which to search
;  1021          1105  2 !       name_len - length of the name
;  1022          1106  2 !
;  1023          1107  2 ! IMPLICIT INPUTS:
;  1024          1108  2 !
;  1025          1109  2 !       none
;  1026          1110  2 !
;  1027          1111  2 ! OUTPUTS:
;  1028          1112  2 !
;  1029          1113  2 !       ctx      - if a file is found, the context block will describe it
;  1030          1114  2 !
;  1031          1115  2 ! IMPLICIT OUTPUTS:
;  1032          1116  2 !
;  1033          1117  2 !       none
;  1034          1118  2 !
;  1035          1119  2 ! ROUTINE VALUE:
;  1036          1120  2 !
;  1037          1121  2 !       true if able to find the file, false otherwise
;  1038          1122  2 !
;  1039          1123  2 ! SIDE EFFECTS:
;  1040          1124  2 !
;  1041          1125  2 !       none
;  1042          1126  2 !--
;  1043          1127  2 $dbgtrc_prefix ('rtacp_find_file> ');
;  1044          1128  2
;  1045          1129  2 BIND
;  1046          1130  2     volb = ctx [rt11ctx$a_assoc_volb]   : $ref_bblock
;  1047          1131  2     ;
;  1048          1132  2
;  1049          1133  2 $block_check (2, .volb, volb, 557);
;  1050          1134  2 $debug_print_fao ('entry - ctx !XL,  name "!AF"', .ctx, .name_len, .name);
;  1051          1135  2
;  1052          1136  2 ! Loop through the directory looking for permanent files
;  1053          1137  2 !
;  1054          1138  2 WHILE exch$rtacp_next_entry (.ctx, rtnxt$m_permanent) NEQ 0
;  1055          1139  2 DO
;  1056          1140  2
;  1057          1141  2     ! Check to see if the this filename matches the one we are looking for.  If so, simply return true.
;  1058          1142  2     !
;  1059          1143  2     IF exch$cmd_match_filename (.ctx [rt11ctx$l_exp_fullname_len], ctx [rt11ctx$t_exp_fullname], .name_len,
;  1060          1144  2     THEN
;  1061          1145  2         RETURN true;
;  1062          1146  2
;  1063          1147  2 RETURN false;                                    ! Made it through without a match, return false
;  1064          1148  1 END;
```

```
                                        000C 00000          .ENTRY  EXCH$RTACP_FIND_FILE, Save R2,R3        ; 1093
                53      04      AC      14   C1 00002          ADDL3   #20, CTX, R3                        ; 1130
                                52 041B00F3  8F   D0 00007          MOVL    #68878579, R2                   ; 1133
                                51    022D   8F   3C 0000E          MOVZWL  #557, R1                        ;
                                50           63   D0 00013          MOVL    (R3), R0                        ;
                                   00000000G EF   16 00016          JSB     EXCH$UTIL_BLOCK_CHECK           ;
                53      04      AC 00000054  8F   C1 0001C          ADDL3   #84, CTX, R3                    ; 1143
                                52       04  AC   D0 00025          MOVL    CTX, R2                         ;
                                             01   DD 00029 1$:      PUSHL   #1                              ; 1138
                                         04  AC   DD 0002B          PUSHL   CTX                             ;
                        0000V   CF           02   FB 0002E          CALLS   #2, EXCH$RTACP_NEXT_ENTRY       ;
                                             50   D5 00033          TSTL    R0                              ;
                                             19   13 00035          BEQL    2$                              ;
                                         08  AC   DD 00037          PUSHL   NAME                            ; 1143
                                         0C  AC   DD 0003A          PUSHL   NAME_LEN                        ;
                                             53   DD 0003D          PUSHL   R3                              ;
                                         46  A2   DD 0003F          PUSHL   70(R2)                          ;
                    00000000G   EF           04   FB 00042          CALLS   #4, EXCH$CMD_MATCH_FILENAME     ;
                                DD           50   E9 00049          BLBC    R0, 1$                          ;
                                50           01   D0 0004C          MOVL    #1, R0                          ; 1145
                                             04   04 0004F          RET                                     ;
                                             50   D4 00050 2$:      CLRL    R0                              ; 1147
                                             04   04 00052          RET                                     ; 1148

; Routine Size:  83 bytes,    Routine Base:  EXCH$RTACP_CODE + 0666
```

```
;  1066          1149  1 GLOBAL ROUTINE exch$rtacp_next_entry (ctx : $ref_bblock, flags : $bblock) =     %SBTTL 'exch$rtacp_next_entr
;  1067          1150  2 BEGIN
;  1068          1151  2 !++
;  1069          1152  2 !
;  1070          1153  2 ! FUNCTIONAL DESCRIPTION:
;  1071          1154  2 !
;  1072          1155  2 !       Return the next entry from an RT-11 directory
;  1073          1156  2 !
;  1074          1157  2 ! INPUT:
;  1075          1158  2 !
;  1076          1159  2 !       ctx   - pointer to an RT11CTX structure
;  1077          1160  2 !       flags - a structure of bits with the following meanings:
;  1078          1161  2 !
;  1079          1162  2 !                               flags [rtnxt$v_permanent]     - return the address of the next permanent entry
;  1080          1163  2 !                               flags [rtnxt$v_empty]         - return the address of the next empty entry
;  1081          1164  2 !                               flags [rtnxt$v_tentative]     - return the address of the next tentative entry
;  1082          1165  2 !                               flags [rtnxt$v_unknown]       - return the address of the next entry with invalid
;  1083          1166  2 !                               flags [rtnxt$v_skip_check]    - skip the check for a moved entry
;  1084          1167  2 !                               flags [rtnxt$v_skip_expand]   - skip expanding the radix-50 name to ascii
;  1085          1168  2 !
;  1086          1169  2 ! More than one flag can be set if more than one type of entry is desired.  The end-of-segment entry can not
;  1087          1170  2 ! returned, but the caller can examine the [rt11ctx$l_seg_number] and interpret a change as an implicit eos.
;  1088          1171  2 !
;  1089          1172  2 ! IMPLICIT INPUTS:
;  1090          1173  2 !
;  1091          1174  2 !       anything we can find hanging off the context block
;  1092          1175  2 !
;  1093          1176  2 ! OUTPUTS:
;  1094          1177  2 !
;  1095          1178  2 !       none
;  1096          1179  2 !
;  1097          1180  2 ! IMPLICIT OUTPUTS:
;  1098          1181  2 !
;  1099          1182  2 !       the context block is updated with context necessary for wildcard processing
;  1100          1183  2 !
;  1101          1184  2 ! ROUTINE VALUE:
;  1102          1185  2 !
;  1103          1186  2 !       address of the next entry, 0 if no more entries of the desired type
;  1104          1187  2 !
;  1105          1188  2 ! SIDE EFFECTS:
;  1106          1189  2 !
;  1107          1190  2 !       none
;  1108          1191  2 !--
;  1109          1192  2
;  1110          1193  2 $dbgtrc_prefix ('rtacp_next_entry> ');
;  1111          1194  2
;  1112          1195  2 LOCAL
;  1113          1196  2     get_dirseg,
;  1114          1197  2     start_block,
;  1115          1198  2     seg_num,
;  1116          1199  2     seg : $ref_bblock,                      ! a pointer to the current directory segment
;  1117          1200  2     ent : $ref_bblock                       ! a pointer to the current directory entry
;  1118          1201  2     ;
;  1119          1202  2
;  1120          1203  2 BIND
;  1121          1204  2     volb = ctx [rt11ctx$a_assoc_volb]   : $ref_bblock
;  1122          1205  2     ;
```

```
;  1124          1206  2 $debug_print_fao ('entry - ctx !XL', .ctx);
;  1125          1207  2 $block_check (2, .ctx, rt11ctx, 552);
;  1126          1208  2 $block_check (2, .volb, volb, 553);
;  1127          1209  2 $logic_check (5, (exch$rtacp_verify_directory (.volb)), 212);
;  1128          1210  2
;  1129          1211  2 ! If the context segment number is null, then we are starting from scratch to find the file
;  1130          1212  2 !
;  1131          1213  2 IF .ctx [rt11ctx$l_seg_number] EQL 0
;  1132          1214  2 THEN
;  1133          1215  3     BEGIN
;  1134          1216  3
;  1135          1217  3     ! Start with the first directory segment
;  1136          1218  3     !
;  1137          1219  3     seg_num = 1;
;  1138          1220  3
;  1139          1221  3     ! Set flag that we must read a directory segment pointer
;  1140          1222  3     !
;  1141          1223  3     get_dirseg = true;
;  1142          1224  3
;  1143          1225  3     END
;  1144          1226  3
;  1145          1227  3 ! If non-null, we are doing a subsequent lookup in a wildcard search
;  1146          1228  3 !
;  1147          1229  2 ELSE
;  1148          1230  3     BEGIN
;  1149          1231  3
;  1150          1232  3     ! Check that the directory entry positions are still good.  The entry might have moved - if so repositio
;  1151          1233  3     ! the new location.  The skip_check option cannot be used if there is any possibility of any directory
;  1152          1234  3     ! modifications between calls.
;  1153          1235  3     !
;  1154          1236  4     IF NOT (.flags [rtnxt$v_skip_check])
;  1155          1237  3     THEN
;  1156          1238  3         exch$rtacp_check_position (.ctx);
;  1157          1239  3
;  1158          1240  3     ! Get the segment, entry, and start block, and adjust to point to the next entry
;  1159          1241  3     !
;  1160          1242  3     seg = .ctx [rt11ctx$a_seg_address];          ! Get our seg from the context block
;  1161          1243  3     ent = .ctx [rt11ctx$a_ent_address];          ! Get the entry from the context block too
;  1162          1244  3     seg_num = .ctx [rt11ctx$l_seg_number];       ! Get the segment number
;  1163          1245  3     start_block =                                ! Add length of current entry
;  1164          1246  3                 .ctx [rt11ctx$l_start_block] + .ent [rt11ent$w_blocks];
;  1165          1247  3     ent = .ent + rt11ent$k_length                ! Skip the pointer to the next entry
;  1166          1248  3                + .seg [rt11hdr$w_extra_bytes];
;  1167          1249  3     ctx [rt11ctx$a_ent_address] = .ent;          ! Save it in the context block too
;  1168          1250  3
;  1169          1251  3     ! Clear the flag, we don't need to read a directory segment pointer right now
;  1170          1252  3     !
;  1171          1253  3     get_dirseg = false;
;  1172          1254  3
;  1173          1255  2     END;
```

```
;  1175          1256  2 ! Loop through the directory looking for this file
;  1176          1257  2 !
;  1177          1258  2 WHILE .seg_num NEQ 0
;  1178          1259  2 DO
;  1179          1260  3     BEGIN
;  1180          1261  3
;  1181          1262  3     IF .get_dirseg
;  1182          1263  3     THEN
;  1183          1264  4         BEGIN
;  1184          1265  4
;  1185          1266  4         ! Get a pointer to the current segment
;  1186          1267  4         !
;  1187          1268  4         seg = exch$rt11_dirseg_get (.volb, .seg_num);
;  1188          1269  4         $logic_check (2, (.seg NEQ 0), 156);
;  1189          1270  4         start_block = .seg [rt11hdr$w_start_block];     ! Fetch the pbn where the first file in this segment
;  1190          1271  4         ent = .seg + rt11hdr$k_length;                  ! Get a pointer to the first directory entry
;  1191          1272  4
;  1192          1273  3         END;
;  1193          1274  3
;  1194          1275  3     get_dirseg = true;                              ! We always want to get additional segs
;  1195          1276  3
;  1196          1277  4     WHILE (.ent LSSU (.seg + rt11$k_dirseglen))
;  1197          1278  3     DO
;  1198          1279  4         BEGIN
;  1199          1280  4
;  1200          1281  4         ! Process the entry depending on its type
;  1201          1282  4         !
;  1202          1283  4         CASE .ent [rt11ent$v_type] FROM 0 TO rt11ent$m_typ_end_segment OF
;  1203          1284  4         SET
;  1204          1285  4
;  1205          1286  4         [rt11ent$m_typ_permanent] :
;  1206          1287  4
;  1207          1288  5                 BEGIN
;  1208          1289  5
;  1209          1290  5                 IF .flags [rtnxt$v_permanent]
;  1210          1291  5                 THEN
;  1211          1292  6                     BEGIN
;  1212          1293  6
;  1213          1294  6                     ! Copy the standard portion of the entry to the context block
;  1214          1295  6                     !
;  1215          1296  6                     CH$MOVE (rt11ent$k_length, .ent, ctx [rt11ctx$t_entry]);
;  1216          1297  6
;  1217          1298  6                     ! Expand the directory entry filename information into the context block
;  1218          1299  6                     !
;  1219          1300  7                     IF NOT (.flags [rtnxt$v_skip_expand])
;  1220          1301  6                     THEN
;  1221          1302  6                         exch$rt11_expand_filename (.ctx);
;  1222          1303  6
;  1223        P 1304  6                     $debug_print_fao ('Found "!AF!AF"', .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident
;  1224          1305  6                                                         .ctx [rt11ctx$l_exp_fullname_len], ctx [rt11ctx$t_exp_fullna
;  1225          1306  6
;  1226          1307  6                     ! Save the directory position in the context block
;  1227          1308  6                     !
;  1228          1309  6                     ctx [rt11ctx$l_start_block] = .start_block;
;  1229          1310  6                     ctx [rt11ctx$l_seg_number]  = .seg_num;
;  1230          1311  6                     ctx [rt11ctx$a_seg_address] = .seg;
;  1231          1312  6                     ctx [rt11ctx$a_ent_address] = .ent;
```

```
;  1232          1313  6                        RETURN .ent;
;  1233          1314  6                        END;
;  1234          1315  5
;  1235          1316  5                    start_block = .start_block + .ent [rt11ent$w_blocks];
;  1236          1317  5
;  1237          1318  5                    END;
;  1238          1319  4
;  1239          1320  4            [rt11ent$m_typ_tentative] :
;  1240          1321  4
;  1241          1322  4                    BEGIN
;  1242          1323  5
;  1243          1324  5                    IF .flags [rtnxt$v_tentative]
;  1244          1325  5                    THEN
;  1245          1326  5                        BEGIN
;  1246          1327  6
;  1247          1328  6                        ! Copy the standard portion of the entry to the context block
;  1248          1329  6                        !
;  1249          1330  6                        CH$MOVE (rt11ent$k_length, .ent, ctx [rt11ctx$t_entry]);
;  1250          1331  6
;  1251          1332  6                        ! Expand the directory entry filename information into the context block
;  1252          1333  6                        !
;  1253          1334  6                        IF NOT (.flags [rtnxt$v_skip_expand])
;  1254          1335  7                        THEN
;  1255          1336  6                            exch$rt11_expand_filename (.ctx);
;  1256          1337  6
;  1257          1338  6                        $debug_print_fao ('Found "!AF!AF"', .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident
;  1258        P 1339  6                                                          .ctx [rt11ctx$l_exp_fullname_len], ctx [rt11ctx$t_exp_fullna
;  1259          1340  6
;  1260          1341  6
;  1261          1342  6                        ! Save the directory position in the context block
;  1262          1343  6                        !
;  1263          1344  6                        ctx [rt11ctx$l_start_block] = .start_block;
;  1264          1345  6                        ctx [rt11ctx$l_seg_number]  = .seg_num;
;  1265          1346  6                        ctx [rt11ctx$a_seg_address] = .seg;
;  1266          1347  6                        ctx [rt11ctx$a_ent_address] = .ent;
;  1267          1348  6
;  1268          1349  6                        RETURN .ent;
;  1269          1350  5                        END;
;  1270          1351  5
;  1271          1352  5                    start_block = .start_block + .ent [rt11ent$w_blocks];
;  1272          1353  5
;  1273          1354  4                    END;
;  1274          1355  4
;  1275          1356  4            [rt11ent$m_typ_empty] :
;  1276          1357  4
;  1277          1358  5                    BEGIN
;  1278          1359  5
;  1279          1360  5                    IF .flags [rtnxt$v_empty]
;  1280          1361  5                    THEN
;  1281          1362  6                        BEGIN
;  1282          1363  6
;  1283          1364  6                        ! Copy the standard portion of the entry to the context block
;  1284          1365  6                        !
;  1285          1366  6                        CH$MOVE (rt11ent$k_length, .ent, ctx [rt11ctx$t_entry]);
;  1286          1367  6
;  1287          1368  6                        ! Expand the directory entry filename information into the context block
;  1288          1369  6                        !
```

```
;  1289          1370  7                    IF NOT (.flags [rtnxt$v_skip_expand])
;  1290          1371  6                    THEN
;  1291          1372  6                        exch$rt11_expand_filename (.ctx);
;  1292          1373  6
;  1293        P 1374  6                    $debug_print_fao ('Found "!AF!AF"', .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident
;  1294          1375  6                                                      .ctx [rt11ctx$l_exp_fullname_len], ctx [rt11ctx$t_exp_fullna
;  1295          1376  6
;  1296          1377  6                    ! Save the directory position in the context block
;  1297          1378  6                    !
;  1298          1379  6                    ctx [rt11ctx$l_start_block] = .start_block;
;  1299          1380  6                    ctx [rt11ctx$l_seg_number]  = .seg_num;
;  1300          1381  6                    ctx [rt11ctx$a_seg_address] = .seg;
;  1301          1382  6                    ctx [rt11ctx$a_ent_address] = .ent;
;  1302          1383  6
;  1303          1384  6                    RETURN .ent;
;  1304          1385  5                    END;
;  1305          1386  5
;  1306          1387  5                start_block = .start_block + .ent [rt11ent$w_blocks];
;  1307          1388  5
;  1308          1389  4                END;
;  1309          1390  4
;  1310          1391  4            [INRANGE, OUTRANGE] :
;  1311          1392  4
;  1312          1393  5                BEGIN
;  1313          1394  5
;  1314          1395  5                IF .flags [rtnxt$v_unknown]
;  1315          1396  5                THEN
;  1316          1397  6                    BEGIN
;  1317          1398  6
;  1318          1399  6                    ! Copy the standard portion of the entry to the context block
;  1319          1400  6                    !
;  1320          1401  6                    CH$MOVE (rt11ent$k_length, .ent, ctx [rt11ctx$t_entry]);
;  1321          1402  6
;  1322          1403  6                    ! Expand the directory entry filename information into the context block
;  1323          1404  6                    !
;  1324          1405  7                    IF NOT (.flags [rtnxt$v_skip_expand])
;  1325          1406  6                    THEN
;  1326          1407  6                        exch$rt11_expand_filename (.ctx);
;  1327          1408  6
;  1328        P 1409  6                    $debug_print_fao ('Found "!AF!AF"', .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident
;  1329          1410  6                                                      .ctx [rt11ctx$l_exp_fullname_len], ctx [rt11ctx$t_exp_fullna
;  1330          1411  6
;  1331          1412  6                    ! Save the directory position in the context block
;  1332          1413  6                    !
;  1333          1414  6                    ctx [rt11ctx$l_start_block] = .start_block;
;  1334          1415  6                    ctx [rt11ctx$l_seg_number]  = .seg_num;
;  1335          1416  6                    ctx [rt11ctx$a_seg_address] = .seg;
;  1336          1417  6                    ctx [rt11ctx$a_ent_address] = .ent;
;  1337          1418  6
;  1338          1419  6                    RETURN .ent;
;  1339          1420  5                    END;
;  1340          1421  5
;  1341          1422  5                start_block = .start_block + .ent [rt11ent$w_blocks];
;  1342          1423  5
;  1343          1424  4                END;
;  1344          1425  4
;  1345          1426  4            [rt11ent$m_typ_end_segment] :
```

```
;  1346          1427  4
;  1347          1428  4                     EXITLOOP;
;  1348          1429  4
;  1349          1430  4                 TES;
;  1350          1431  4
;  1351          1432  4                 ! Skip to the next entry
;  1352          1433  4                 !
;  1353          1434  4                 ent = .ent + rt11ent$k_length + .seg [rt11hdr$w_extra_bytes];
;  1354          1435  4
;  1355          1436  3                 END;
;  1356          1437  3
;  1357          1438  3             ! Skip to the next segment
;  1358          1439  3             !
;  1359          1440  3             seg_num = .seg [rt11hdr$w_next_seg];
;  1360          1441  3
;  1361          1442  2             END;
;  1362          1443  2
;  1363          1444  2 ! Mark the context with an invalid segment number to prevent recycling through the directory
;  1364          1445  2 !
;  1365          1446  2 ctx [rt11ctx$l_seg_number] = -1;
;  1366          1447  2
;  1367          1448  2 RETURN 0;
;  1368          1449  1 END;


                                    0FFC 00000          .ENTRY  EXCH$RTACP_NEXT_ENTRY, Save R2,R3,R4,R5,R6,-; 1149
                                                                R7,R8,R9,R10,R11
                              5E            0C  C2 00002          SUBL2   #12, SP
                              57            04  AC D0 00005          MOVL    CTX, R7                             ; 1204
                              52    008200F4    8F D0 00009          MOVL    #8519924, R2                        ; 1207
                              51        0228    8F 3C 00010          MOVZWL  #552, R1
                              50              57 D0 00015          MOVL    R7, R0
                                00000000G     EF 16 00018          JSB     EXCH$UTIL_BLOCK_CHECK
                              52    041B00F3    8F D0 0001E          MOVL    #68878579, R2                       ; 1208
                              51        0229    8F 3C 00025          MOVZWL  #553, R1
                              50          14    A7 D0 0002A          MOVL    20(R7), R0
                                00000000G     EF 16 0002E          JSB     EXCH$UTIL_BLOCK_CHECK
                              5B            76  A7 9E 00034          MOVAB   118(R7), R11                        ; 1213
                                              6B D5 00038          TSTL    (R11)
                                              08 12 0003A          BNEQ    1$
                              5A              01 D0 0003C          MOVL    #1, SEG_NUM                         ; 1219
                              6E              01 D0 0003F          MOVL    #1, GET_DIRSEG                      ; 1223
                                              2E 11 00042          BRB     3$                                  ; 1213
                07        08  AC              04 E0 00044 1$:      BBS     #4, FLAGS, 2$                       ; 1236
                                              57 DD 00049          PUSHL   R7                                  ; 1238
                          F8F7  CF              01 FB 0004B          CALLS   #1, EXCH$RTACP_CHECK_POSITION
                              58          7A  A7 D0 00050 2$:      MOVL    122(R7), SEG                        ; 1242
                              56          7E  A7 D0 00054          MOVL    126(R7), ENT                        ; 1243
                              5A              6B D0 00058          MOVL    (R11), SEG_NUM                      ; 1244
                              59          08  A6 3C 0005B          MOVZWL  8(ENT), START_BLOCK                 ; 1246
                              59          72  A7 C0 0005F          ADDL2   114(R7), START_BLOCK
                              50          06  A8 3C 00063          MOVZWL  6(SEG), R0                          ; 1248
                              56        0E A046 9E 00067          MOVAB   14(R0)[ENT], ENT
                            7E  A7            56 D0 0006C          MOVL    ENT, 126(R7)                        ; 1249
```

```
                                                  6E  D4  00070          CLRL    GET_DIRSEG                                      ; 1253
                                                  5A  D5  00072  3$:     TSTL    SEG_NUM                                         ; 1258
                                                  03  12  00074          BNEQ    4$                                              ;
                                              00C1  31  00076          BRW     18$                                             ;
                                         2C       6E  E9  00079  4$:     BLBC    GET_DIRSEG, 6$                                  ; 1262
                                                  5A  DD  0007C          PUSHL   SEG_NUM                                         ; 1268
                                            14    A7  DD  0007E          PUSHL   20(R7)                                          ;
                                 00000000G    EF       02  FB  00081          CALLS   #2, EXCH$RT11_DIRSEG_GET                        ;
                                         50       58  D0  00088          MOVL    R0, SEG                                         ;
                                                  13  12  0008B          BNEQ    5$                                              ; 1269
                                         7E    9C       8F  9A  0008D          MOVZBL  #156, -(SP)                             ;
                                                  01  DD  00091          PUSHL   #1                                              ;
                                      00000000G   8F  DD  00093          PUSHL   #EXCH$_BADLOGIC                                 ;
                                 00000000G    00       03  FB  00099          CALLS   #3, LIB$STOP                            ;
                                         59       08  A8  3C  000A0  5$:     MOVZWL  8(SEG), START_BLOCK                     ; 1270
                                         56       0A  A8  9E  000A4          MOVAB   10(R8), ENT                             ; 1271
                                         6E           01  D0  000A8  6$:     MOVL    #1, GET_DIRSEG                          ; 1275
                                   08    AE     0400  C8  9E  000AB          MOVAB   1024(R8), 8(SP)                         ; 1277
                                   04    AE       06  A8  9E  000B1          MOVAB   6(SEG), 4(SP)                           ; 1434
                                   08    AE           56  D1  000B6  7$:     CMPL    ENT, 8(SP)                              ; 1277
                                                  77  1E  000BA          BGEQU   17$                                             ;
                7E     01    A6                 04      00  EF  000BC          EXTZV   #0, #4, 1(ENT), -(SP)                   ; 1283
                            08                 00      8E  CF  000C2          CASEL   (SP)+, #0, #8                           ;
              0012        0030               0029          000C6  8$:     .WORD   9$-8$,-                                         ;
              0012        0012               0012          000CE                  12$-8$,-                                        ;
                                                  006D      000D6                  13$-8$,-                                        ;
                                                                                   9$-8$,-                                         ;
                                                                                   11$-8$,-                                        ;
                                                                                   9$-8$,-                                         ;
                                                                                   9$-8$,-                                         ;
                                                                                   9$-8$,-                                         ;
                                                                                   9$-8$,-                                         ;
                                                                                   17$-8$                                          ;
          44        08    AC                       03  E1  000D8  9$:     BBC     #3, FLAGS, 16$                                  ; 1395
    38    A7              66                       0E  28  000DD  10$:    MOVC3   #14, (ENT), 56(R7)                              ; 1401
          1E        08    AC                       05  E1  000E2          BBC     #5, FLAGS, 14$                                  ; 1405
                                                  25  11  000E7          BRB     15$                                             ; 1414
                          34               08      AC  E9  000E9  11$:    BLBC    FLAGS, 16$                                      ; 1290
                                                  EE  11  000ED          BRB     10$                                             ; 1296
          2D        08    AC                       02  E1  000EF  12$:    BBC     #2, FLAGS, 16$                                  ; 1325
                                                  E7  11  000F4          BRB     10$                                             ; 1331
          26        08    AC                       01  E1  000F6  13$:    BBC     #1, FLAGS, 16$                                  ; 1360
    38    A7              66                       0E  28  000FB          MOVC3   #14, (ENT), 56(R7)                              ; 1366
          09        08    AC                       05  E0  00100          BBS     #5, FLAGS, 15$                                  ; 1370
                                                  57  DD  00105  14$:    PUSHL   R7                                              ; 1372
                 00000000G  EF                       01  FB  00107          CALLS   #1, EXCH$RT11_EXPAND_FILENAME                   ;
                       72   A7                       59  D0  0010E  15$:    MOVL    START_BLOCK, T14(R7)                    ; 1379
                       6B                           5A  D0  00112          MOVL    SEG_NUM, (R11)                          ; 1380
                       7A   A7                       58  D0  00115          MOVL    SEG, 122(R7)                            ; 1381
                       7E   A7                       56  D0  00119          MOVL    ENT, 126(R7)                            ; 1382
                            50                       56  D0  0011D          MOVL    ENT, R0                                 ; 1384
                                                      04  00120          RET                                                     ;
                            50               08      A6  3C  00121  16$:    MOVZWL  8(ENT), R0                              ; 1387
                            59                       50  C0  00125          ADDL2   R0, START_BLOCK                         ;
                            50               04      BE  3C  00128          MOVZWL  @4(SP), R0                              ; 1434
                            56               0E  A046  9E  0012C          MOVAB   14(R0)[ENT], ENT                        ;
                                                  83  11  00131          BRB     7$                                              ; 1277
                            5A               02      A8  3C  00133  17$:    MOVZWL  2(SEG), SEG_NUM                         ; 1440
```

```
                                FF38  31 00137         BRW     3$                              ; 1258
                                   6B   01  CE 0013A 18$:     MNEGL   #1, (R11)               ; 1446
                                        50  D4 0013D         CLRL    R0                      ; 1448
                                            04 0013F         RET                             ; 1449

; Routine Size:  320 bytes,    Routine Base:  EXCH$RTACP_CODE + 06B9
```

```
;1370     1450  1 GLOBAL ROUTINE exch$rtacp_verify_directory (volb : $ref_bblock) =       %SBTTL 'exch$rtacp_verify_directory
;1371     1451  2 BEGIN
;1372     1452  2 !++
;1373     1453  2 !
;1374     1454  2 ! FUNCTIONAL DESCRIPTION:
;1375     1455  2 !
;1376     1456  2 !       Traverse the directory and check it for validity.  We also count the number of blocks.
;1377     1457  2 !
;1378     1458  2 ! INPUTS:
;1379     1459  2 !
;1380     1460  2 !       volb - pointer to volb which has been connected to the RT-11 device
;1381     1461  2 !
;1382     1462  2 ! IMPLICIT INPUTS:
;1383     1463  2 !
;1384     1464  2 !       none
;1385     1465  2 !
;1386     1466  2 ! OUTPUTS:
;1387     1467  2 !
;1388     1468  2 !       blocks - address of longword to receive number of blocks
;1389     1469  2 !
;1390     1470  2 ! IMPLICIT OUTPUTS:
;1391     1471  2 !
;1392     1472  2 !       none
;1393     1473  2 !
;1394     1474  2 ! ROUTINE VALUE:
;1395     1475  2 !
;1396     1476  2 !       true if valid, false if not
;1397     1477  2 !
;1398     1478  2 ! SIDE EFFECTS:
;1399     1479  2 !
;1400     1480  2 !       error conditions will be signaled
;1401     1481  2 !--
;1402     1482  2
;1403     1483  2 $dbgtrc_prefix ('exch$rtacp_verify_directory> ');
;1404     1484  2
;1405     1485  2 LOCAL
;1406     1486  2     rtv : $ref_bblock,                          ! a pointer to the rt11 volb extension
;1407     1487  2     seg : $ref_bblock,                          ! a pointer to the current directory segment
;1408     1488  2     ent : $ref_bblock,                          ! a pointer to the current directory entry
;1409     1489  2     sum_blocks,                                 ! running block count
;1410     1490  2     unknowns,                                   ! count of unknown directory entries
;1411     1491  2     seg_num,
;1412     1492  2     end_segment_seen,
;1413     1493  2     missing_end,
;1414     1494  2     status
;1415     1495  2     ;
;1416     1496  2
;1417     1497  2 $debug_print_lit ('entry');
;1418     1498  2
;1419     1499  2 $block_check (2, .volb, volb, 470);
;1420     1500  2
;1421     1501  2 ! Assume that we will find a bad directory
;1422     1502  2 !
;1423     1503  2 status = 0;
;1424     1504  2 unknowns = false;                              ! Clear the unknown entry flag
;1425     1505  2 missing_end = 0;
```

```
; 1427          1506  2 ! Get the pointer to our volb extension and to the root segment
; 1428          1507  2 !
; 1429          1508  2 rtv = .volb [volb$a_vfmt_specific];
; 1430          1509  2 $block_check (2, .rtv, rt11, 471);
; 1431          1510  2 seg = rtv [rt11$t_block_0] + (512 * rt11$k_root_block);
; 1432          1511  2
; 1433          1512  2 ! Start with the first directory segment
; 1434          1513  2 !
; 1435          1514  2 seg_num = 1;
; 1436          1515  2
; 1437          1516  2 ! Assume that the files will start after the last segment
; 1438          1517  2 !
; 1439          1518  2 sum_blocks = rt11$k_root_block + (2 * .seg [rt11hdr$w_num_segs]);
; 1440          1519  2 $debug_print_fao ('number of segments !UL', .seg [rt11hdr$w_num_segs]);
; 1441          1520  2
; 1442          1521  2 ! Loop through all the segments in the directory
; 1443          1522  2 !
; 1444          1523  2 WHILE .seg_num NEQ 0
; 1445          1524  2 DO
; 1446          1525  3     BEGIN
; 1447          1526  3
; 1448          1527  3     ! Get a pointer to the current segment
; 1449          1528  3     !
; 1450          1529  3     seg = exch$rt11_dirseg_get (.volb, .seg_num);
; 1451          1530  3     IF .seg EQL 0
; 1452          1531  3     THEN
; 1453          1532  3         RETURN false;
; 1454          1533  3
; 1455          1534  3     ! If the current segment start block is not what we expect, signal and use new value
; 1456          1535  3     !
; 1457          1536  3     $debug_print_fao ('expected block !UL, actual block !UL', .sum_blocks, .seg [rt11hdr$w_start_block]);
; 1458          1537  4     IF (.sum_blocks NEQ .seg [rt11hdr$w_start_block])
; 1459          1538  3     THEN
; 1460          1539  4         BEGIN
; 1461          1540  4         $exch_signal (exch$_rt11_stblock);
; 1462          1541  4         sum_blocks = .seg [rt11hdr$w_start_block];
; 1463          1542  3         END;
```

```
; 1465          1543  3     ! Get a pointer to the first directory entry, then loop
; 1466          1544  3     !
; 1467          1545  3     ent = .seg + rt11hdr$k_length;
; 1468          1546  3     end_segment_seen = false;
; 1469          1547  4     WHILE (.ent LSSU (.seg + rt11$k_dirseglen))
; 1470          1548  3     DO
; 1471          1549  4         BEGIN
; 1472          1550  4
; 1473          1551  4         ! Advance our count
; 1474          1552  4         !
; 1475          1553  4         CASE .ent [rt11ent$v_type] FROM 0 TO rt11ent$m_typ_end_segment OF
; 1476          1554  4         SET
; 1477          1555  4         [rt11ent$m_typ_tentative, rt11ent$m_typ_empty, rt11ent$m_typ_permanent] :
; 1478          1556  5                 BEGIN
; 1479          1557  5                 $debug_print_fao ('entry type !XB, size !UL', .ent [rt11ent$b_type_byte], .ent [rt11ent$w_bl
; 1480          1558  5                 sum_blocks = .sum_blocks + .ent [rt11ent$w_blocks];
; 1481          1559  4                 END;
; 1482          1560  4         [rt11ent$m_typ_end_segment] :
; 1483          1561  5                 BEGIN
; 1484          1562  5                 end_segment_seen = true;
; 1485          1563  5                 EXITLOOP;
; 1486          1564  4                 END;
; 1487          1565  4         [INRANGE, OUTRANGE] :
; 1488          1566  5                 BEGIN
; 1489          1567  5                 unknowns = true;
; 1490          1568  5                 $debug_print_fao ('(unknown) entry type !XB, size !UL', .ent [rt11ent$b_type_byte], .ent [rt
; 1491          1569  5                 sum_blocks = .sum_blocks + .ent [rt11ent$w_blocks];
; 1492          1570  4                 END;
; 1493          1571  4         TES;
; 1494          1572  4
; 1495          1573  4         ! Skip to the next entry
; 1496          1574  4         !
; 1497          1575  4         ent = .ent + rt11ent$k_length + .seg [rt11hdr$w_extra_bytes];
; 1498          1576  4
; 1499          1577  3         END;
; 1500          1578  3
; 1501          1579  3     ! Holler if we didn't see the end segment
; 1502          1580  3     !
; 1503          1581  4     IF NOT (.end_segment_seen)
; 1504          1582  3     THEN
; 1505          1583  3         missing_end = true;
; 1506          1584  3
; 1507          1585  3     ! Skip to the next segment
; 1508          1586  3     !
; 1509          1587  3     seg_num = .seg [rt11hdr$w_next_seg];
; 1510          1588  3
; 1511          1589  2     END;
```

```
;  1513          1590  2 ! Assume now that we have found a valid directory
;  1514          1591  2 !
;  1515          1592  2 status = true;
;  1516          1593  2
;  1517          1594  2 ! If we saw any bad entries, signal and write lock the volume
;  1518          1595  2 !
;  1519          1596  2 IF .unknowns OR .missing_end                    ! Unknown format of some entries or missing marker
;  1520          1597  2 THEN
;  1521          1598  2     BEGIN
;  1522          1599  2     LOCAL
;  1523          1600  2         sig_stat;
;  1524          1601  2
;  1525          1602  2     sig_stat = (IF .unknowns THEN exch$_rt11_unkent ELSE exch$_rt11_noend);     ! Get the proper signal name
;  1526          1603  2
;  1527          1604  2     ! Signal if this is the first verify, or if we have write access
;  1528          1605  2     !
;  1529          1606  2     IF  NOT .volb [volb$v_verified]
;  1530          1607  2       OR
;  1531          1608  2         .volb [volb$v_write]
;  1532          1609  2     THEN
;  1533          1610  2         $exch_signal (.sig_stat, 0, exch$_rt11_errlock);
;  1534          1611  2
;  1535          1612  2     ! Fail the directory validation this is not the first and we have write access
;  1536          1613  2     !
;  1537          1614  2     IF  .volb [volb$v_verified]
;  1538          1615  2       AND
;  1539          1616  2         .volb [volb$v_write]
;  1540          1617  2     THEN
;  1541          1618  2         status = false;
;  1542          1619  2
;  1543          1620  2     volb [volb$v_write] = false;                 ! Force a write lock
;  1544          1621  2     END;
;  1545          1622  2
;  1546          1623  2 ! Check if the block count is what we expected
;  1547          1624  2 !
;  1548          1625  2 $debug_print_fao ('volmaxblock !UL, sum_blocks !UL', .volb [volb$l_volmaxblock], .sum_blocks);
;  1549          1626  2 IF .volb [volb$l_volmaxblock] NEQ .sum_blocks
;  1550          1627  2 THEN
;  1551          1628  2     BEGIN
;  1552          1629  3
;  1553          1630  3     ! If the volume is larger than the device there is no hope
;  1554          1631  3     !
;  1555          1632  3     $debug_print_fao ('devmaxblock !UL, sum_blocks !UL', .volb [volb$l_devmaxblock], .sum_blocks);
;  1556          1633  3     IF .sum_blocks GTRU .volb [volb$l_devmaxblock]
;  1557          1634  3     THEN
;  1558          1635  4         BEGIN
;  1559        P 1636  4         $exch_signal (exch$_rt11_dirsize, 4, .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident],
;  1560          1637  4                       .sum_blocks, .volb [volb$l_devmaxblock], exch$_rt11_errlock);
;  1561          1638  4         volb [volb$v_write] = false;
;  1562          1639  3         END;
;  1563          1640  3
;  1564          1641  3     ! If this is the first time that the directory has been verified, store
;  1565          1642  3     ! the calculated value as the real value.
;  1566          1643  3     !
;  1567          1644  3     IF NOT .volb [volb$v_verified]
;  1568          1645  3     THEN
;  1569          1646  4         BEGIN
```

```
; 1570    1647 4          $logic_check (3, (.volb [volb$l_volmaxblock] EQL .volb [volb$l_devmaxblock]), 190);
; 1571    1648 4          volb [volb$l_volmaxblock] = .sum_blocks;
; 1572    1649 4          END
; 1573    1650 4
; 1574    1651 4      ! Otherwise, we have corrupted the directory since we started
; 1575    1652 4      !
; 1576    1653 3      ELSE
; 1577    1654 3          $logic_check (0, (false), 316);
; 1578    1655 2      END;
; 1579    1656 2
; 1580    1657 2  ! Set the verification flag
; 1581    1658 2  !
; 1582    1659 2  volb [volb$v_verified] = true;
; 1583    1660 2
; 1584    1661 2  RETURN .status;
; 1585    1662 1  END;


                                                               .EXTRN  EXCH$_RT11_STBLOCK
                                                               .EXTRN  EXCH$_RT11_UNKENT
                                                               .EXTRN  EXCH$_RT11_NOEND
                                                               .EXTRN  EXCH$_RT11_ERRLOCK
                                                               .EXTRN  EXCH$_RT11_DIRSIZE

                                       0FFC 00000              .ENTRY  EXCH$RTACP_VERIFY_DIRECTORY, Save R2,R3,R4,-; 1450
                                                                       R5,R6,R7,R8,R9,R10,R11                     ;
                 5E               04    C2 00002               SUBL2   #4, SP                                     ;
                 54          04   AC    D0 00005               MOVL    VOLB, R4                                   ; 1499
                 52    041B00F3   8F    D0 00009               MOVL    #68878579, R2                              ;
                 51        01D6   8F    3C 00010               MOVZWL  #470, R1                                   ;
                 50               54    D0 00015               MOVL    R4, R0                                     ;
                      00000000G   EF    16 00018               JSB     EXCH$UTIL_BLOCK_CHECK                      ;
                                  5A    7C 0001E               CLRQ    UNKNOWNS                                   ; 1504
                                  59    D4 00020               CLRL    MISSING_END                                ; 1505
                 53          54   A4    D0 00022               MOVL    84(R4), RTV                                ; 1508
                 52    880E00F5   8F    D0 00026               MOVL    #-2012348171, R2                           ; 1509
                 51        01D7   8F    3C 0002D               MOVZWL  #471, R1                                   ;
                 50               53    D0 00032               MOVL    RTV, R0                                    ;
                      00000000G   EF    16 00035               JSB     EXCH$UTIL_BLOCK_CHECK                      ;
                 55        0C0E   C3    9E 0003B               MOVAB   3086(R3), SEG                              ; 1510
                 58               01    D0 00040               MOVL    #1, SEG_NUM                                ; 1514
                 52               65    3C 00043               MOVZWL  (SEG), R2                                  ; 1518
                 52               02    C4 00046               MULL2   #2, SUM_BLOCKS                             ;
                 52               06    C0 00049               ADDL2   #6, SUM_BLOCKS                             ;
                                  58    D5 0004C 1$:           TSTL    SEG_NUM                                    ; 1523
                                  03    12 0004E               BNEQ    2$                                         ;
                                0084    31 00050               BRW     13$                                        ;
                           0110   8F    BB 00053 2$:           PUSHR   #^M<R4,R8>                                 ; 1529
            00000000G  EF         02    FB 00057               CALLS   #2, EXCH$RT11_DIRSEG_GET                   ;
                       55         50    D0 0005E               MOVL    R0, SEG                                    ;
                                  03    12 00061               BNEQ    3$                                         ; 1530
                                0110    31 00063               BRW     24$                                        ;
   52         08   A5          10   00    ED 00066 3$:           CMPZV   #0, #16, 8(SEG), SUM_BLOCKS                ; 1537
                                  11    13 0006C               BEQL    4$                                         ;
                      00000000G   8F    DD 0006E               PUSHL   #EXCH$_RT11_STBLOCK                        ; 1540
            00000000G  00         01    FB 00074               CALLS   #1, LIB$SIGNAL                             ;
```

```
                                52      08   A5  3C 0007B          MOVZWL  8(SEG), SUM_BLOCKS                      ; 1541
                                53      0A   A5  9E 0007F 4$:      MOVAB   10(R5), ENT                             ; 1545
                                             56  D4 00083          CLRL    END_SEGMENT_SEEN                        ; 1546
                                50    0400   C5  9E 00085          MOVAB   1024(R5), R0                            ; 1547
                                57      06   A5  9E 0008A          MOVAB   6(SEG), R7                              ; 1575
                                50           53  D1 0008E 5$:      CMPL    ENT, R0                                 ; 1547
                                             37  1E 00091          BGEQU   11$                                     ;
          51        01   A3                04  00  EF 00093          EXTZV   #0, #4, 1(ENT), R1                      ; 1553
                         08                00  51  CF 00099          CASEL   R1, #0, #8                              ;
        0012          0015              0015      0012   0009D 6$:   .WORD   7$-6$,-                                 ;
        0012          0012              0012      0015   000A5          8$-6$,-                                     ;
                                                  001E   000AD          8$-6$,-                                     ;
                                                                       7$-6$,-                                     ;
                                                                       8$-6$,-                                     ;
                                                                       7$-6$,-                                     ;
                                                                       7$-6$,-                                     ;
                                                                       7$-6$,-                                     ;
                                                                       9$-6$                                       ;
                                5A           01  D0 000AF 7$:      MOVL    #1, UNKNOWNS                            ; 1567
                                6E      08   A3  3C 000B2 8$:      MOVZWL  8(ENT), (SP)                            ; 1558
                                52           6E  C0 000B6          ADDL2   (SP), SUM_BLOCKS                        ;
                                             05  11 000B9          BRB     10$                                     ; 1553
                                56           01  D0 000BB 9$:      MOVL    #1, END_SEGMENT_SEEN                    ; 1562
                                             0A  11 000BE          BRB     11$                                     ; 1561
                                51           67  3C 000C0 10$:     MOVZWL  (R7), R1                                ; 1575
                                53      0E A143  9E 000C3          MOVAB   14(R1)[ENT], ENT                        ;
                                             C4  11 000C8          BRB     5$                                      ; 1547
                                03           56  E8 000CA 11$:     BLBS    END_SEGMENT_SEEN, 12$                   ; 1581
                                59           01  D0 000CD          MOVL    #1, MISSING_END                         ; 1583
                                58      02   A5  3C 000D0 12$:     MOVZWL  2(SEG), SEG_NUM                         ; 1587
                                          FF75   31 000D4          BRW     1$                                      ; 1523
                                5B           01  D0 000D7 13$:     MOVL    #1, STATUS                              ; 1592
                                06           5A  E8 000DA          BLBS    UNKNOWNS, 14$                           ; 1596
                                3E           59  E9 000DD          BLBC    MISSING_END, 20$                        ;
                                09           5A  E9 000E0          BLBC    UNKNOWNS, 15$                           ; 1602
                                50 00000000G 8F  D0 000E3 14$:     MOVL    #EXCH$_RT11_UNKENT, SIG_STAT            ;
                                             07  11 000EA          BRB     16$                                     ;
                                50 00000000G 8F  D0 000EC 15$:     MOVL    #EXCH$_RT11_NOEND, SIG_STAT             ;
                                       48    A4  95 000F3 16$:     TSTB    72(R4)                                  ; 1606
                                             05  18 000F6          BGEQ    17$                                     ;
          11        48   A4                05  E1 000F8          BBC     #5, 72(R4), 18$                         ; 1608
                                00000000G    8F  DD 000FD 17$:     PUSHL   #EXCH$_RT11_ERRLOCK                     ; 1610
                                             7E  D4 00103          CLRL    -(SP)                                   ;
                                             50  DD 00105          PUSHL   SIG_STAT                                ;
             00000000G  00                   03  FB 00107          CALLS   #3, LIB$SIGNAL                          ;
                                       48    A4  95 0010E 18$:     TSTB    72(R4)                                  ; 1614
                                             07  18 00111          BGEQ    19$                                     ;
          02        48   A4                05  E1 00113          BBC     #5, 72(R4), 19$                         ; 1616
                                             5B  D4 00118          CLRL    STATUS                                  ; 1618
                    48   A4                20  8A 0011A 19$:     BICB2   #32, 72(R4)                             ; 1620
                                52      44   A4  D1 0011E 20$:     CMPL    68(R4), SUM_BLOCKS                      ; 1626
                                             49  13 00122          BEQL    23$                                     ;
                    40   A4                52  D1 00124          CMPL    SUM_BLOCKS, 64(R4)                      ; 1633
                                             24  1B 00128          BLEQU   21$                                     ;
                                00000000G    8F  DD 0012A          PUSHL   #EXCH$_RT11_ERRLOCK                     ; 1637
                                       40    A4  DD 00130          PUSHL   64(R4)                                  ;
                                             52  DD 00133          PUSHL   SUM_BLOCKS                              ;
```

```
                                        69    A4  9F 00135          PUSHAB  105(R4)                                          ;
                                        65    A4  DD 00138          PUSHL   101(R4)                                          ;
                                              04  DD 0013B          PUSHL   #4                                               ;
                                 00000000G    8F  DD 0013D          PUSHL   #EXCH$_RT11_DIRSIZE                              ;
                      00000000G  00              07  FB 00143          CALLS   #7, LIB$SIGNAL                                   ;
                              48  A4            20  8A 0014A          BICB2   #32, 72(R4)                                      ; 1638
                                        48    A4  95 0014E 21$:      TSTB    72(R4)                                           ; 1644
                                              06  19 00151          BLSS    22$                                              ;
                              44  A4            52  D0 00153          MOVL    SUM_BLOCKS, 68(R4)                               ; 1648
                                              14  11 00157          BRB     23$                                              ; 1644
                                    7E    013C    8F  3C 00159 22$:      MOVZWL  #316, -(SP)                              ; 1654
                                              01  DD 0015E          PUSHL   #1                                               ;
                                 00000000G    8F  DD 00160          PUSHL   #EXCH$_BADLOGIC                                  ;
                      00000000G  00              03  FB 00166          CALLS   #3, LIB$STOP                                     ;
                              48  A4        80    8F  88 0016D 23$:      BISB2   #128, 72(R4)                             ; 1659
                                    50            5B  D0 00172          MOVL    STATUS, R0                               ; 1661
                                                  04  00175          RET                                                      ;
                                              50  D4  00176 24$:      CLRL    R0                                       ; 1662
                                                  04  00178          RET                                                      ;

; Routine Size:  377 bytes,    Routine Base:  EXCH$RTACP_CODE + 07F9
```

```
; 1587          1663  1 END
; 1588          1664  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL, LIB$STOP

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  EXCH$RTACP_CODE                 2418  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         1         0        1000        00:01.9
;  _$255$DUA28:[EXCHNG.OBJ]EXCLIB.L32;1      1151        74         6          79        00:01.4


;                               COMMAND QUALIFIERS

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

; Size:         2418 code + 0 data bytes
; Run Time:       00:51.6
; Elapsed Time:   02:35.2
; Lines/CPU Min:   1936
; Lexemes/CPU-Min: 17630
; Memory Used:  236 pages
; Compilation Complete
```

EXCSHOW
LIS

F11A

F11AACP
MAP

EXCRTNAM
LIS

EXCRTACP
LIS

EXCUTIL
LIS