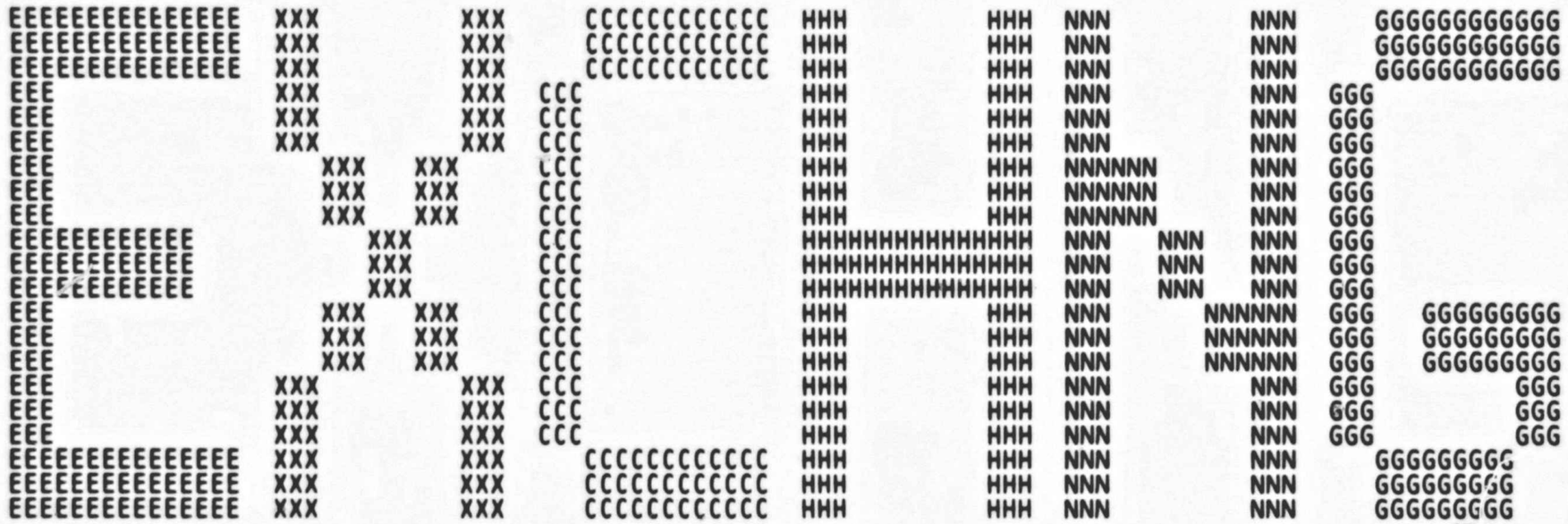
EEEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGGGG
EEEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGGGG
EEEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGGGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG
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

FILEID**EXCRT11

```
EEEEEEEEEE  XX      XX    CCCCCCCC  RRRRRRRR    TTTTTTTTTT      11          11
EEEEEEEEEE  XX      XX    CCCCCCCC  RRRRRRRR    TTTTTTTTTT      11          11
EE          XX      XX  CC          RR      RR      TT        1111        1111
EE          XX      XX  CC          RR      RR      TT        1111        1111
EE            XX  XX    CC          RR      RR      TT          11          11
EE            XX  XX    CC          RR      RR      TT          11          11
EEEEEEEE        XX      CC          RRRRRRRR        TT          11          11
EEEEEEEE        XX      CC          RRRRRRRR        TT          11          11
EE            XX  XX    CC          RR  RR          TT          11          11
EE            XX  XX    CC          RR  RR          TT          11          11
EE          XX      XX  CC          RR    RR        TT          11          11        ....
EE          XX      XX  CC          RR    RR        TT          11          11        ....
EEEEEEEEEE  XX      XX    CCCCCCCC  RR      RR      TT      111111      111111        ....
EEEEEEEEEE  XX      XX    CCCCCCCC  RR      RR      TT      111111      111111        ....

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$rt11                                %TITLE 'RT11 file and 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 volume specific routines
;    35          0035  1 !
;    36          0036  1 ! ENVIRONMENT:  VAX/VMS User mode
;    37          0037  1 !
;    38          0038  1 ! AUTHOR:       CW Hobbs                  CREATION DATE: 26-Aug-1982
;    39          0039  1 !
;    40          0040  1 ! MODIFIED BY:
;    41          0041  1 !
;    42          0042  1 !       V03-004 CWH3004         CW Hobbs                  25-Jul-1984
;    43          0043  1 !               Move logic check 175 to after a test for global caching,
;    44          0044  1 !               since globally cached write-locked volumes were hitting
;    45          0045  1 !               the trap.
;    46          0046  1 !
;    47          0047  1 !       V03-003 CWH3003         CW Hobbs                  12-Apr-1984
;    48          0048  1 !               Disable message about recovering devices, and force
;    49          0049  1 !               /TRANSFER=BLOCK to be global
;    50          0050  1 !
;    51          0051  1 !       V03-002 CWH9001         CW Hobbs                  30-Apr-1983
;    52          0052  1 !               Remove debugging call accidentally checked in.
;    53          0053  1 !
;    54          0054  1 !--
;    55          0055  1
;    56          0056  1 ! Include files:
;    57          0057  1 !
```

```
;   58          0058  1 MACRO $module_name_string = 'exch$rt11' %;      ! The require file needs to know our module name
;   59          0059  1 REQUIRE 'SRC$:EXCREQ'                          ! Facility-wide require file
;   60          0060  1     ;
```

```
;    62          0157  1 %SBTTL 'Module table of contents'
;    63          0158  1
;    64          0159  1 ! Module table of contents:
;    65          0160  1 !
;    66          0161  1 FORWARD ROUTINE
;    67          0162  1     exch$rt11_bad_file  : NOVALUE,              ! Create a bad file over a bad spot on the volume
;    68          0163  1     exch$rt11_close_file,                       ! RT-11 specific file close routine
;    69          0164  1     exch$rt11_create_file,                      ! RT-11 volume file create
;    70          0165  1     exch$rt11_delete_file,                      ! RT-11 specific file delete routine
;    71          0166  1     exch$rt11_dircache_exit_handler : NOVALUE,  ! Routine to flush the cache
;    72          0167  1     exch$rt11_dircache_start    : NOVALUE,      ! Engage write-back caching on the directory
;    73          0168  1     exch$rt11_dircache_stop     : NOVALUE,      ! Disengage caching and flush the directory
;    74          0169  1     exch$rt11_dirseg_flush,                     ! Write out directory segments
;    75          0170  1     exch$rt11_dirseg_get,                       ! Return pointer to specific directory segment
;    76          0171  1     exch$rt11_dirseg_get_nochk  : jsb_r1r2,     ! Return pointer to directory segment without checking
;    77          0172  1     exch$rt11_dirseg_put,                       ! Write a specific directory segment
;    78          0173  1     exch$rt11_expand_filename,                  ! Convert directory entry to ASCII filename
;    79          0174  1     exch$rt11_format_current_date : NOVALUE jsb_r1,     ! Put the current date into a directory entry
;    80          0175  1     exch$rt11_mount,                            ! RT-11 specific volume mount routine
;    81          0176  1     exch$rt11_open_file,                        ! RT-11 specific file open routine
;    82          0177  1     exch$rt11_write_cleanup     : NOVALUE,      ! Finish writing to the volume
;    83          0178  1     exch$rt11_write_prepare     : NOVALUE,      ! Prepare to write to the volume
;    84          0179  1     exch$rt11_zero_marks        : NOVALUE       ! Clear marks on volume
;    85          0180  1     ;
;    86          0181  1
;    87          0182  1 ! EXCHANGE facility routines
;    88          0183  1 !
;    89          0184  1 EXTERNAL ROUTINE
;    90          0185  1     exch$cmd_fetch_recfmt_implied : NOVALUE,    ! Get or assume the value for /RECORD_FORMAT
;    91          0186  1     exch$cmd_match_filename,                    ! Compare expanded file names for match
;    92          0187  1     exch$cmd_related_file_parse,                ! Build an output file name
;    93          0188  1     exch$io_rt11_read,                          ! Read blocks from a random access device
;    94          0189  1     exch$io_rt11_write,                         ! Write blocks to a random access device
;    95          0190  1     exch$pdp_filter_filename,                   ! Remove invalid characters from a filename
;    96          0191  1     exch$pdp_flush_write_buffer,                ! Flush any records waiting in the write buffer
;    97          0192  1     exch$pdp_get,                               ! Get functions for small PDP record structure
;    98          0193  1     exch$pdp_put,                               ! Put functions for small PDP record structure
;    99          0194  1     exch$rtacp_check_position : NOVALUE,        ! Find directory entry if it has moved
;   100          0195  1     exch$rtacp_clean_directory,                 ! Shuffle and/or split directories as needed
;   101          0196  1     exch$rtacp_find_empty_area,                 ! Compress directory structure and find free space
;   102          0197  1     exch$rtacp_find_file,                       ! RT-11 directory search routine
;   103          0198  1     exch$rtacp_verify_directory,                ! Verify directory structure and compute volume size
;   104          0199  1     exch$util_fao_buffer,                       ! Do an FAO conversion
;   105          0200  1     exch$util_file_error,                       ! Signal an RMS error
;   106          0201  1     exch$util_radix50_from_ascii,               ! Convert characters to Radix-50 from Ascii
;   107          0202  1     exch$util_radix50_to_ascii,                 ! Convert characters from Radix-50 to Ascii
;   108          0203  1     exch$util_rt11ctx_allocate,                 ! Get an RT-11 context block
;   109          0204  1     exch$util_rt11ctx_release   : NOVALUE,      ! Give it back
;   110          0205  1     exch$util_vm_allocate,                      ! Get some virtual memory
;   111          0206  1     exch$util_vm_allocate_zeroed,               ! Get some virtual memory, initialized to zero
;   112          0207  1     exch$util_vm_release                        ! Return some virtual memory
;   113          0208  1     ;
;   114          0209  1
;   115          0210  1 ! Equated symbols:
;   116          0211  1 !
;   117          0212  1 !LITERAL
;   118          0213  1 !    ;
```

```
;  119          0214  1
;  120          0215  1 ! Bound declarations:
;  121          0216  1 !
;  122          0217  1 !BIND
;  123          0218  1 !    ;
```

```
;   125     0219  1 GLOBAL ROUTINE exch$rt11_bad_file (filb : $ref_bblock) : NOVALUE =      %SBTTL 'exch$rt11_bad_file (filb)'
;   126     0220  2 BEGIN
;   127     0221  2 !++
;   128     0222  2 !
;   129     0223  2 ! FUNCTIONAL DESCRIPTION:
;   130     0224  2 !
;   131     0225  2 !       Perform RT-11 bad block handling by placing a FILE.BAD file over the bad block.  This routine will b
;   132     0226  2 !       called when a bad block is detected on the output file during a copy operation.  We assume that ther
;   133     0227  2 !       a zero-block empty file entry following the current entry.
;   134     0228  2 !
;   135     0229  2 !       One of the following actions will be taken:
;   136     0230  2 !
;   137     0231  2 !          The bad block is the first block in the output file:
;   138     0232  2 !                  The output file is renamed to a 1 block FILE.BAD and made permanent.  Remaining bloc
;   139     0233  2 !                  moved to the empty file.  (Single block files are treated as this case)
;   140     0234  2 !
;   141     0235  2 !          The bad block is in the middle of the output file:
;   142     0236  2 !                  The output file will be left as a tentative file.  If there is room for another entr
;   143     0237  2 !                  the current directory segment, then a 1 block FILE.BAD is created and the remaining
;   144     0238  2 !                  blocks are moved to a newly created empty file.  If there is no room to add an entry
;   145     0239  2 !                  FILE.BAD will contain all the free blocks in addition to the one known to be bad.
;   146     0240  2 !
;   147     0241  2 !          The bad block is at the end of the file:
;   148     0242  2 !                  The output file will be left as a tentative file, and a 1 block FILE.BAD will be cre
;   149     0243  2 !
;   150     0244  2 ! INPUT/OUTPUT:
;   151     0245  2 !
;   152     0246  2 !       filb - pointer to block describing the file
;   153     0247  2 !
;   154     0248  2 ! IMPLICIT INPUTS:
;   155     0249  2 !
;   156     0250  2 !       things hanging from the filb, notably the bad block number is in RAB$L_BKT in the volb rab.
;   157     0251  2 !
;   158     0252  2 ! OUTPUTS:
;   159     0253  2 !
;   160     0254  2 !       filb - receive info pertaining to the file to be closed
;   161     0255  2 !
;   162     0256  2 ! IMPLICIT OUTPUTS:
;   163     0257  2 !
;   164     0258  2 !       none
;   165     0259  2 !
;   166     0260  2 ! ROUTINE VALUE:
;   167     0261  2 !
;   168     0262  2 !       none
;   169     0263  2 !
;   170     0264  2 ! SIDE EFFECTS:
;   171     0265  2 !
;   172     0266  2 !       none
;   173     0267  2 !--
;   174     0268  2
;   175     0269  2 $dbgtrc_prefix ('exch$rt11_bad_file> ');
;   176     0270  2
;   177     0271  2 LOCAL
;   178     0272  2     bad_pbn,                                    ! pbn of the bad block
;   179     0273  2     blks_before,                                ! blocks before the bad block
;   180     0274  2     blks_after,                                 ! blocks after the bad one
;   181     0275  2     ent_len,                                    ! length of a directory entry
```

```
;  182     0276  2      emp : $ref_bblock,                      ! pointer to the empty entry after this one
;  183     0277  2      eos : $ref_bblock,                      ! pointer to the end of segment marker
;  184     0278  2      status
;  185     0279  2      ;
;  186     0280  2
;  187     0281  2  BIND
;  188     0282  2      copy = exch$a_gbl [excg$a_copy_work]: $ref_bblock,
;  189     0283  2      ctx  = filb [filb$a_context]        : $ref_bblock,
;  190     0284  2      namb = filb [filb$a_assoc_namb]     : $ref_bblock,
;  191     0285  2      volb = filb [filb$a_assoc_volb]     : $ref_bblock,
;  192     0286  2      rab  = volb [volb$a_rab]            : $ref_bblock,
;  193     0287  2      seg  = ctx [rt11ctx$a_seg_address]  : $ref_bblock,  ! pointer to the directory segment
;  194     0288  2      ent  = ctx [rt11ctx$a_ent_address]  : $ref_bblock   ! and the directory entry for this file
;  195     0289  2      ;
;  196     0290  2
;  197     0291  2  $debug_print_lit ('entry');
;  198     0292  2
;  199     0293  2  $block_check (2, .filb, filb, 565);                    !?? definitely over-zealous checking
;  200     0294  2  $block_check (2, .namb, namb, 566);
;  201     0295  2  $block_check (2, .volb, volb, 567);
;  202     0296  2  $block_check (2, .ctx, rt11ctx, 568);
;  203     0297  2  $logic_check (2, (.ctx [rt11ctx$a_assoc_filb] EQL .filb), 217);
;  204     0298  2  $logic_check (2, (.ctx [rt11ctx$a_assoc_volb] EQL .volb), 218);
;  205     0299  2  $logic_check (2, (.ctx [rt11ctx$v_output_file]), 219);
;  206     0300  2  $logic_check (3, (exch$rtacp_verify_directory (.volb)), 220);
;  207     0301  2
;  208     0302  2  ! The bad block number is sitting in the rab
;  209     0303  2  !
;  210     0304  2  bad_pbn = .rab [rab$l_bkt];
;  211     0305  2  IF .volb [volb$v_virtual]                          ! Undo pbn -> vbn mapping
;  212     0306  2  THEN
;  213     0307  2      bad_pbn = .bad_pbn - 1;
;  214     0308  2  $logic_check (2, ((.bad_pbn GEQU .ctx [rt11ctx$l_start_block]) AND (.bad_pbn LEQU .ctx [rt11ctx$l_eof_block]
;  215     0309  2  $trace_print_fao ('bad_pbn !UL', .bad_pbn);
;  216     0310  2
;  217     0311  2  ! Let the outer routines know that we have erased this file
;  218     0312  2  !
;  219     0313  2  filb [filb$v_file_erased] = true;
;  220     0314  2
;  221     0315  2  ! Get the pointer to the empty entry after this one
;  222     0316  2  !
;  223     0317  2  ent_len = rt11ent$k_length + .seg [rt11hdr$w_extra_bytes];
;  224     0318  2  emp = .ent + .ent_len;
;  225     0319  2  $logic_check (3, ((.emp [rt11ent$b_type_byte] EQL rt11ent$m_typ_empty) AND (.emp [rt11ent$w_blocks] EQL 0)),
;  226     0320  2  $logic_check (3, ((.ent [rt11ent$v_type] EQL rt11ent$m_typ_tentative) AND (.ent [rt11ent$w_blocks] NEQ 0)),
;  227     0321  2
;  228     0322  2  ! A structured GOTO follows.  EXITLOOPs will be used to rejoin common code at the end of the routine
;  229     0323  2  !
;  230     0324  2  WHILE 1
;  231     0325  2  DO
;  232     0326  3      BEGIN
;  233     0327  3
;  234     0328  3      ! How many blocks were written before the bad block
;  235     0329  3      !
;  236     0330  3      blks_before = .bad_pbn - .ctx [rt11ctx$l_start_block];
;  237     0331  3
;  238     0332  3      ! If blocks before is zero, we can tie this off right now
```

```
;   239     0333  3     !
;   240     0334  3     IF .blks_before EQL 0
;   241     0335  3     THEN
;   242     0336  4         BEGIN
;   243     0337  4
;   244     0338  4         ! Move any remaining blocks to the empty entry
;   245     0339  4         !
;   246     0340  4         emp [rt11ent$w_blocks] = .ent [rt11ent$w_blocks] - 1;
;   247     0341  4
;   248     0342  4         ! Create a one block permanent FILE.BAD in the entry
;   249     0343  4         !
;   250     0344  4         ent [rt11ent$w_blocks] = 1;
;   251     0345  4
;   252     0346  4         $exch_signal (exch$_rt11_badfile, 1, .bad_pbn);         ! Tell the guy that we made a .BAD file
;   253     0347  4
;   254     0348  4         EXITLOOP;                                       ! Done here, jump to the end to fill in the rest of the bad
;   255     0349  3         END;
;   256     0350  3
;   257     0351  3     ! How many blocks are after the bad one
;   258     0352  3     !
;   259     0353  3     blks_after = .ctx [rt11ctx$l_eof_block] - .bad_pbn;
;   260     0354  3
;   261     0355  3     ! If blocks after is zero, we can also do the work and exit
;   262     0356  3     !
;   263     0357  3     IF .blks_before EQL 0
;   264     0358  3     THEN
;   265     0359  4         BEGIN
;   266     0360  4
;   267     0361  4         ! Remove one block from the tentative file
;   268     0362  4         !
;   269     0363  4         ent [rt11ent$w_blocks] = .ent [rt11ent$w_blocks] - 1;
;   270     0364  4
;   271     0365  4         ! Move the empty pointer to the ent pointer, where the common code expects to find the bad entry
;   272     0366  4         !
;   273     0367  4         ent = .emp;
;   274     0368  4
;   275     0369  4         ! Create a one block permanent FILE.BAD in the empty entry
;   276     0370  4         !
;   277     0371  4         ent [rt11ent$w_blocks] = 1;
;   278     0372  4
;   279     0373  4         $exch_signal (exch$_rt11_badfile, 1, .bad_pbn);         ! Tell the guy
;   280     0374  4
;   281     0375  4         EXITLOOP;
;   282     0376  3         END;
;   283     0377  3
;   284     0378  3     ! OK, the bad block is in the middle of the tentative file.  We have two choices now, depending on wheth
;   285     0379  3     ! have room in the segment to add another file entry.  First, find the end of segment marker.
;   286     0380  3     !
;   287     0381  3     eos = .emp;                                         ! Point to the empty entry
;   288     0382  3     WHILE 1
;   289     0383  3     DO
;   290     0384  4         BEGIN
;   291     0385  4         eos = .eos + .ent_len;                          ! Advance to the next entry
;   292     0386  4         $logic_check (2, .eos LSSU (.seg + rt11$k_dirseglen)), 224);
;   293     0387  4         IF .eos [rt11ent$v_type] EQL rt11ent$m_typ_end_segment
;   294     0388  4         THEN
;   295     0389  4             EXITLOOP
```

```
;  296     0390  3          END;
;  297     0391  3
;  298     0392  3      ! Make sure that there is room to add one more entry to this segment.  If not, we will have to add a big
;  299     0393  3      !
;  300     0394  4      IF ((.eos+2 + .ent_len) GEQU (.seg + rt11$k_dirseglen))
;  301     0395  3      THEN
;  302     0396  4          BEGIN
;  303     0397  4
;  304     0398  4          ! Make the tentative file include all the blocks before the bad one
;  305     0399  4          !
;  306     0400  4          ent [rt11ent$w_blocks] = .blks_before;
;  307     0401  4
;  308     0402  4          ! Move the empty pointer to the ent pointer, where the common code expects to find the bad entry
;  309     0403  4          !
;  310     0404  4          ent = .emp;
;  311     0405  4
;  312     0406  4          ! Put the rest in a permanent FILE.BAD in the empty entry
;  313     0407  4          !
;  314     0408  4          ent [rt11ent$w_blocks] = .blks_after + 1;
;  315     0409  4
;  316     0410  4          $exch_signal (exch$_rt11_badfile, 1, .bad_pbn, exch$_rt11_bigbadfile);  ! Tell the guy the bad news
;  317     0411  4
;  318     0412  4          EXITLOOP;
;  319     0413  4          END
;  320     0414  4
;  321     0415  4      ! Room for another entry, make it <TENT> <BAD> <EMPTY>
;  322     0416  4      !
;  323     0417  3      ELSE
;  324     0418  4          BEGIN
;  325     0419  4          LOCAL
;  326     0420  4              sl;
;  327     0421  4
;  328     0422  4          ! Slide the rest of the segment up one entry to make room for the new entry
;  329     0423  4          !
;  330     0424  4          sl = .eos+2 - .emp;                                 ! Length of segment between empty and end
;  331     0425  4          CH$MOVE (.sl, .emp, .emp + .ent_len);   ! Slide rest of segment up
;  332     0426  4
;  333     0427  4          ! Finish up the tentative entry
;  334     0428  4          !
;  335     0429  4          ent [rt11ent$w_blocks] = .blks_before;
;  336     0430  4
;  337     0431  4          ! Point "ent" at the bad entry and "emp" at the new empty
;  338     0432  4          !
;  339     0433  4          ent = .emp;
;  340     0434  4          emp = .emp + .ent_len;
;  341     0435  4
;  342     0436  4          ! Finish up the new empty entry.  Since we slid the old empty up, all we need to do is set the lengt
;  343     0437  4          !
;  344     0438  4          emp [rt11ent$w_blocks] = .blks_after;
;  345     0439  4          $logic_check (3, (.emp [rt11ent$b_type_byte] EQL rt11ent$m_typ_empty), 225);
;  346     0440  4
;  347     0441  4          ! Create a one block permanent FILE.BAD in the middle entry.
;  348     0442  4          !
;  349     0443  4          ent [rt11ent$w_blocks] = 1;
;  350     0444  4
;  351     0445  4          $exch_signal (exch$_rt11_badfile, 1, .bad_pbn);         ! Tell the guy
;  352     0446  4
```

```
;  353          0447  4          EXITLOOP;
;  354          0448  3          END;
;  355          0449  3      $logic_check (0, (false), 216);          ! We should have hit an exitloop before here
;  356          0450  2      END;
;  357          0451  2
;  358          0452  2  ! "ent" points to the bad entry, fill in the info common to all three cases
;  359          0453  2  !
;  360          0454  2  ent [rt11ent$b_type_byte] = rt11ent$m_typ_permanent;
;  361          0455  2  ent [rt11ent$l_filename]  = r50_file;                    ! "FILE  " in radix 50
;  362          0456  2  ent [rt11ent$w_filetype]  = r50_bad;                     ! "BAD"    in radix 50
;  363          0457  2  exch$rt11_format_current_date (.ent);
;  364          0458  2
;  365          0459  2  exch$rt11_dirseg_put (.volb, .ctx [rt11ctx$l_seg_number]);      ! Flush the modified segment
;  366          0460  2
;  367          0461  2  ! Force a directory update on disk if caching is active
;  368          0462  2  !
;  369          0463  2  IF .volb [volb$v_dircache_active]
;  370          0464  2  THEN
;  371          0465  3      BEGIN
;  372          0466  3      exch$rt11_dircache_stop (.volb);            ! Do the I/O
;  373          0467  3      exch$rt11_dircache_start (.volb);           ! Renable caching
;  374          0468  2      END;
;  375          0469  2
;  376          0470  2  ! Set a flag so that the copy command will attempt to retry the current file
;  377          0471  2  !
;  378          0472  2  copy [copy$v_reopen_input] = true;
;  379          0473  2
;  380          0474  2  RETURN;
;  381          0475  2
;  382          0476  1  END;


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

                                .EXTRN  EXCH$CMD_FETCH_RECFMT_IMPLIED
                                .EXTRN  EXCH$CMD_MATCH_FILENAME
                                .EXTRN  EXCH$CMD_RELATED_FILE_PARSE
                                .EXTRN  EXCH$IO_RT11_READ
                                .EXTRN  EXCH$IO_RT11_WRITE
                                .EXTRN  EXCH$PDP_FILTER_FILENAME
                                .EXTRN  EXCH$PDP_FLUSH_WRITE_BUFFER
                                .EXTRN  EXCH$PDP_GET, EXCH$PDP_PUT
                                .EXTRN  EXCH$RTACP_CHECK_POSITION
                                .EXTRN  EXCH$RTACP_CLEAN_DIRECTORY
                                .EXTRN  EXCH$RTACP_FIND_EMPTY_AREA
                                .EXTRN  EXCH$RTACP_FIND_FILE
                                .EXTRN  EXCH$RTACP_VERIFY_DIRECTORY
                                .EXTRN  EXCH$UTIL_FAO_BUFFER
                                .EXTRN  EXCH$UTIL_FILE_ERROR
                                .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$A_GBL, EXCH$UTIL_BLOCK_CHECK
                                        .EXTRN  EXCH$_BADLOGIC, EXCH$_RT11_BADFILE
                                        .EXTRN  EXCH$_RT11_BIGBADFILE

                                        .PSECT  EXCH$RT11_CODE,NOWRT,2

                                0FFC 00000      .ENTRY  EXCH$RT11_BAD_FILE, Save R2,R3,R4,R5,R6,R7,-; 0219
                                                        R8,R9,R10,R11
                          5E            10  C2 00002      SUBL2   #16, SP
                7E 00000000G  EF            04  C1 00005      ADDL3   #4, EXCH$A_GBL, -(SP)                   0282
                          53        04    AC  D0 0000D      MOVL    FILB, R3                                    0283
                          59        1C    A3  D0 00011      MOVL    28(R3), R9                                  0286
                          58        20    A3  D0 00015      MOVL    32(R3), R8                                  0287
                          5A        7E    A8  9E 00019      MOVAB   126(R8), R10                                0288
                          52  035B00FA    8F  D0 0001D      MOVL    #56295674, R2                               0293
                          51      0235    8F  3C 00024      MOVZWL  #565, R1
                          50             53  D0 00029      MOVL    R3, R0
                             00000000G   EF  16 0002C      JSB     EXCH$UTIL_BLOCK_CHECK
                          52  010A00F7    8F  D0 00032      MOVL    #17432823, R2                               0294
                          51      0236    8F  3C 00039      MOVZWL  #566, R1
                          50        18    A3  D0 0003E      MOVL    24(R3), R0
                             00000000G   EF  16 00042      JSB     EXCH$UTIL_BLOCK_CHECK
                          52  041B00F3    8F  D0 00048      MOVL    #68878579, R2                               0295
                          51      0237    8F  3C 0004F      MOVZWL  #567, R1
                          50             59  D0 00054      MOVL    R9, R0
                             00000000G   EF  16 00057      JSB     EXCH$UTIL_BLOCK_CHECK
                          52  008200F4    8F  D0 0005D      MOVL    #8519924, R2                                0296
                          51      0238    8F  3C 00064      MOVZWL  #568, R1
                          50             58  D0 00069      MOVL    R8, R0
                             00000000G   EF  16 0006C      JSB     EXCH$UTIL_BLOCK_CHECK
                                   53        10  A8  D1 00072      CMPL    16(R8), R3                          0297
                                             13  13 00076      BEQL    1$
                                   7E        D9  8F  9A 00078      MOVZBL  #217, -(SP)
                                             01  DD 0007C      PUSHL   #1
                                      00000000G  8F  DD 0007E      PUSHL   #EXCH$_BADLOGIC
                             00000000G    00        03  FB 00084      CALLS   #3, LIB$STOP
                                   59        14  A8  D1 0008B 1$:    CMPL    20(R8), R9                          0298
                                             13  13 0008F      BEQL    2$
                                   7E        DA  8F  9A 00091      MOVZBL  #218, -(SP)
                                             01  DD 00095      PUSHL   #1
                                      00000000G  8F  DD 00097      PUSHL   #EXCH$_BADLOGIC
                             00000000G    00        03  FB 0009D      CALLS   #3, LIB$STOP
                          13        28    A8        01  E0 000A4 2$:    BBS     #1, 40(R8), 3$                  0299
                                   7E        DB  8F  9A 000A9      MOVZBL  #219, -(SP)
                                             01  DD 000AD      PUSHL   #1
                                      00000000G  8F  DD 000AF      PUSHL   #EXCH$_BADLOGIC
                             00000000G    00        03  FB 000B5      CALLS   #3, LIB$STOP
                                   50        14  A9  D0 000BC 3$:    MOVL    20(R9), R0                          0304
                                   5B        38  A0  D0 000C0      MOVL    56(R0), BAD_PBN
                          02        48    A9        04  E1 000C4      BBC     #4, 72(R9), 4$                      0305
                                                    5B  D7 000C9      DECL    BAD_PBN                             0307
                                   72    A8        5B  D1 000CB 4$:    CMPL    BAD_PBN, 114(R8)                    0308
                                                    06  1F 000CF      BLSSU   5$
                                   20    A8        5B  D1 000D1      CMPL    BAD_PBN, 32(R8)
                                                    13  1B 000D5      BLEQU   6$
                                         7E        DD  8F  9A 000D7 5$:    MOVZBL  #221, -(SP)
                                                    01  DD 000DB      PUSHL   #1
```

```
                                00000000G  8F  DD 000DD          PUSHL   #EXCH$_BADLOGIC                          ;
                     00000000G  00         03  FB 000E3          CALLS   #3, LIB$STOP                            ;
                           2B   A3         04  88 000EA 6$:      BISB2   #4, 43(R3)                              ; 0313
                           08   AE    7A   A8  D0 000EE          MOVL    122(R8), 8(SP)                          ; 0317
                50         08   AE         06  C1 000F3          ADDL3   #6, 8(SP), R0                           ;
                           04   AE         60  3C 000F8          MOVZWL  (R0), ENT_LEN                           ;
                           04   AE         0E  C0 000FC          ADDL2   #14, ENT_LEN                            ;
                57               6A   04   AE  C1 00100          ADDL3   ENT_LEN, (R10), EMP                     ; 0318
          0C    AE               5B   72   A8  C3 00105          SUBL3   114(R8), BAD_PBN, BLKS_BEFORE           ; 0330
                                         52  D4 0010B          CLRL    R2                                      ; 0334
                                    0C   AE  D5 0010D          TSTL    BLKS_BEFORE                             ;
                                         0E  12 00110          BNEQ    7$                                      ;
                                         52  D6 00112          INCL    R2                                      ;
                               50        6A  D0 00114          MOVL    (R10), R0                               ; 0340
          08    A7         08   A0         01  A3 00117          SUBW3   #1, 8(R0), 8(EMP)                       ;
                                      00AD  31 0011D          BRW     13$                                     ; 0344
          10    AE         20   A8         5B  C3 00120 7$:      SUBL3   BAD_PBN, 32(R8), BLKS_AFTER             ; 0353
                                    0C   52  E9 00126          BLBC    R2, 8$                                  ; 0357
                               50        6A  D0 00129          MOVL    (R10), R0                               ; 0363
                                    08   A0  B7 0012C          DECW    8(R0)                                   ;
                               6A             57  D0 0012F          MOVL    EMP, (R10)                              ; 0367
                                      0095  31 00132          BRW     12$                                     ; 0371
                               56             57  D0 00135 8$:      MOVL    EMP, EOS                                ; 0381
                               56        04   AE  C0 00138 9$:      ADDL2   ENT_LEN, EOS                            ; 0385
                50         08   AE 00000400   8F  C1 0013C          ADDL3   #1024, 8(SP), R0                        ; 0386
                               50             56  D1 00145          CMPL    EOS, R0                                 ;
                                              13  1F 00148          BLSSU   10$                                     ;
                               7E        E0   8F  9A 0014A          MOVZBL  #224, -(SP)                             ;
                                              01  DD 0014E          PUSHL   #1                                      ;
                                00000000G     8F  DD 00150          PUSHL   #EXCH$_BADLOGIC                         ;
                     00000000G  00         03  FB 00156          CALLS   #3, LIB$STOP                            ;
    08          01   A6                    04  00  ED 0015D 10$:      CMPZV   #0, #4, 1(EOS), #8                      ; 0387
                                              D3  12 00163          BNEQ    9$                                      ;
                50         04   AE         02  C1 00165          ADDL3   #2, ENT_LEN, R0                         ; 0394
                51              56         50  C1 0016A          ADDL3   R0, EOS, R1                             ;
                50         08   AE 00000400   8F  C1 0016E          ADDL3   #1024, 8(SP), R0                        ;
                               50             51  D1 00177          CMPL    R1, R0                                  ;
                                              2D  1F 0017A          BLSSU   11$                                     ;
                               50             6A  D0 0017C          MOVL    (R10), R0                               ; 0400
                                    08   A0        0C   AE  B0 0017F          MOVW    BLKS_BEFORE, 8(R0)                      ;
                               6A             57  D0 00184          MOVL    EMP, (R10)                              ; 0404
                               50             6A  D0 00187          MOVL    (R10), R0                               ; 0408
          08    A0         10   AE         01  A1 0018A          ADDW3   #1, BLKS_AFTER, 8(R0)                   ;
                                00000000G     8F  DD 00190          PUSHL   #EXCH$_RT11_BIGBADFILE                  ; 0410
                                              5B  DD 00196          PUSHL   BAD_PBN                                 ;
                                              01  DD 00198          PUSHL   #1                                      ;
                                00000000G     8F  DD 0019A          PUSHL   #EXCH$_RT11_BADFILE                     ;
                     00000000G  00         04  FB 001A0          CALLS   #4, LIB$SIGNAL                          ;
                                              39  11 001A7          BRB     14$                                     ; 0396
                50             56             57  C3 001A9 11$:      SUBL3   EMP, EOS, R0                            ; 0424
                               50             02  C0 001AD          ADDL2   #2, SL                                  ;
          04 BE47              67             50  28 001B0          MOVC3   SL, (EMP), @ENT_LEN[EMP]                ; 0425
                               50             6A  D0 001B6          MOVL    (R10), R0                               ; 0429
                                    08   A0        0C   AE  B0 001B9          MOVW    BLKS_BEFORE, 8(R0)                      ;
                               6A             57  D0 001BE          MOVL    EMP, (R10)                              ; 0433
                               57        04   AE  C0 001C1          ADDL2   ENT_LEN, EMP                            ; 0434
                                    08   A7        10   AE  B0 001C5          MOVW    BLKS_AFTER, 8(EMP)              ; 0438
```

```
                                50              6A  D0 001CA 12$:    MOVL    (R10), R0                               ; 0443
                             08 A0              01  B0 001CD 13$:    MOVW    #1, 8(R0)
                                                5B  DD 001D1         PUSHL   BAD_PBN                                 ; 0445
                                                01  DD 001D3         PUSHL   #1
                                    00000000G   8F  DD 001D5         PUSHL   #EXCH$_RT11_BADFILE
                         00000000G  00          03  FB 001DB         CALLS   #3, LIB$SIGNAL
                                51              6A  D0 001E2 14$:    MOVL    (R10), R1                               ; 0454
                             01 A1              04  90 001E5         MOVB    #4, 1(R1)
                             02 A1 1F4026F4     8F  D0 001E9         MOVL    #524297972, 2(R1)                       ; 0455
                             06 A1      0CAC    8F  B0 001F1         MOVW    #3244, 6(R1)                            ; 0456
                                        0000V   30  001F7         BSBW    EXCH$RT11_FORMAT_CURRENT_DATE           ; 0457
                                     76 A8      DD  001FA         PUSHL   118(R8)                                 ; 0459
                                                59  DD 001FD         PUSHL   R9
                              0000V CF          02  FB 001FF         CALLS   #2, EXCH$RT11_DIRSEG_PUT
                                    0E      50  A9  E9 00204         BLBC    80(R9), 15$                             ; 0463
                                                59  DD 00208         PUSHL   R9                                      ; 0466
                              0000V CF          01  FB 0020A         CALLS   #1, EXCH$RT11_DIRCACHE_STOP
                                                59  DD 0020F         PUSHL   R9                                      ; 0467
                              0000V CF          01  FB 00211         CALLS   #1, EXCH$RT11_DIRCACHE_START
                                50          00  BE  D0 00216 15$:    MOVL    @0(SP), R0                              ; 0472
                             34 A0              04  88 0021A         BISB2   #4, 52(R0)
                                                04  0021E         RET                                             ; 0476

; Routine Size:  543 bytes,    Routine Base:  EXCH$RT11_CODE + 0000
```

```
;  384          0477  1 GLOBAL ROUTINE exch$rt11_close_file (filb : $ref_bblock) =      %SBTTL 'exch$rt11_close_file (filb)'
;  385          0478  2 BEGIN
;  386          0479  2 !++
;  387          0480  2 !
;  388          0481  2 ! FUNCTIONAL DESCRIPTION:
;  389          0482  2 !
;  390          0483  2 !       Perform RT-11 volume specific close processing
;  391          0484  2 !
;  392          0485  2 ! INPUT/OUTPUT:
;  393          0486  2 !
;  394          0487  2 !       filb - pointer to block describing the file
;  395          0488  2 !
;  396          0489  2 ! IMPLICIT INPUTS:
;  397          0490  2 !
;  398          0491  2 !       none
;  399          0492  2 !
;  400          0493  2 ! OUTPUTS:
;  401          0494  2 !
;  402          0495  2 !       filb - receive info pertaining to the file to be closed
;  403          0496  2 !
;  404          0497  2 ! IMPLICIT OUTPUTS:
;  405          0498  2 !
;  406          0499  2 !       none
;  407          0500  2 !
;  408          0501  2 ! ROUTINE VALUE:
;  409          0502  2 !
;  410          0503  2 !       true if able to close the file, false otherwise
;  411          0504  2 !
;  412          0505  2 ! SIDE EFFECTS:
;  413          0506  2 !
;  414          0507  2 !       none
;  415          0508  2 !--
;  416          0509  2
;  417          0510  2 $dbgtrc_prefix ('exch$rt11_close_file> ');
;  418          0511  2
;  419          0512  2 LOCAL
;  420          0513  2     status
;  421          0514  2     ;
;  422          0515  2
;  423          0516  2 BIND
;  424          0517  2     ctx  = filb [filb$a_context]        : $ref_bblock,
;  425          0518  2     namb = filb [filb$a_assoc_namb]     : $ref_bblock,
;  426          0519  2     volb = filb [filb$a_assoc_volb]     : $ref_bblock
;  427          0520  2     ;
;  428          0521  2
;  429          0522  2 $debug_print_lit ('entry');
;  430          0523  2
;  431          0524  2 $block_check (2, .filb, filb, 447);                  !?? definitely over-zealous checking
;  432          0525  2 $block_check (2, .namb, namb, 448);
;  433          0526  2 $block_check (2, .volb, volb, 449);
;  434          0527  2 $block_check (2, .ctx, rt11ctx, 451);
;  435          0528  2 $logic_check (2, (.ctx [rt11ctx$a_assoc_filb] EQL .filb), 122);
;  436          0529  2 $logic_check (2, (.ctx [rt11ctx$a_assoc_volb] EQL .volb), 123);
;  437          0530  2 $logic_check (5, (exch$rtacp_verify_directory (.volb)), 186);
```

```
;  439          0531  2 ! Output files need some directory tweaks and a buffer flush
;  440          0532  2 !
;  441          0533  2 IF .ctx [rt11ctx$v_output_file]
;  442          0534  2 THEN
;  443          0535  3     BEGIN
;  444          0536  3     LOCAL
;  445          0537  3         blks_used,
;  446          0538  3         emp : $ref_bblock;                          ! pointer to the empty entry after this one
;  447          0539  3     BIND
;  448          0540  3         seg = ctx [rt11ctx$a_seg_address] : $ref_bblock,        ! pointer to a directory segment
;  449          0541  3         ent = ctx [rt11ctx$a_ent_address] : $ref_bblock;        ! and the directory entry for this file
;  450          0542  3
;  451          0543  3     ! Flush any blocks that are sitting in the output buffer
;  452          0544  3     !
;  453          0545  4     IF NOT (status = exch$pdp_flush_write_buffer (.ctx))
;  454          0546  3     THEN
;  455          0547  3         RETURN .status;
;  456          0548  3
;  457          0549  3     ! How many blocks were actually used in the file
;  458          0550  3     !
;  459          0551  3     blks_used = 1 + .ctx [rt11ctx$l_high_block_written] - .ctx [rt11ctx$l_start_block];
;  460          0552  3
;  461          0553  3     ! If an allocation was requested, and the allocation was more than was used, then increase the size to t
;  462          0554  3     ! allocation request.  The extra blocks will be filled with nulls.
;  463          0555  3     !
;  464          0556  5     IF   ((.filb [filb$l_q_allocation] NEQ 0)
;  465          0557  4         AND
;  466          0558  4           (.filb [filb$l_q_allocation] GTRU .blks_used))
;  467          0559  3     THEN
;  468          0560  4         BEGIN
;  469          0561  4         LOCAL
;  470          0562  4             blk_cnt,
;  471          0563  4             blks_to_clear,
;  472          0564  4             cur_blk;
;  473          0565  4
;  474          0566  4         ! Figure out how many blocks to clear and the pbn of the first block to clear
;  475          0567  4         !
;  476          0568  4         blks_to_clear = .filb [filb$l_q_allocation] - .blks_used;            ! Number of null blocks to w
;  477          0569  4         cur_blk = .ctx [rt11ctx$l_high_block_written] + 1;                  ! Block at which to write nu
;  478          0570  4         $logic_check (3, (.ctx [rt11ctx$a_buffer] NEQ 0), 195);
;  479          0571  4         CH$FILL (0, ctx$k_buffer_length, .ctx [rt11ctx$a_buffer]);          ! Fill the buffer with nulls
;  480          0572  4
;  481          0573  4         ! Write the null blocks
;  482          0574  4         !
;  483          0575  4         blk_cnt = .blks_to_clear;
;  484          0576  4         WHILE .blk_cnt GTR 0                        ! Note the signed compare
;  485          0577  4         DO
;  486          0578  5             BEGIN
;  487          0579  6             IF NOT (status = exch$io_rt11_write (.volb,                  ! All the rms stuff hangs fr
;  488          0580  6                                 .cur_blk,                               ! First block to write
;  489          0581  6                                 MINU (.blk_cnt, ctx$k_buffer_blocks),   ! Number of blocks
;  490          0582  6                                 .ctx [rt11ctx$a_buffer]))               ! Address of the I/O buffer
;  491          0583  5             THEN
;  492          0584  6                 BEGIN
;  493          0585  6                 exch$rt11_bad_file (.filb);
;  494          0586  6                 RETURN .status;
;  495          0587  5                 END;
```

```
;  496          0588  5
;  497          0589  5              ! Point at the next chunk to write
;  498          0590  5              !
;  499          0591  5              blk_cnt = .blk_cnt - ctx$k_buffer_blocks;           ! Number of blocks
;  500          0592  5              cur_blk = .cur_blk + ctx$k_buffer_blocks;           ! Number of blocks
;  501          0593  4              END;
;  502          0594  4
;  503          0595  4          ! Update the pointers to the new highest block written
;  504          0596  4          !
;  505          0597  4          ctx [rt11ctx$l_high_block_written] = .ctx [rt11ctx$l_high_block_written] + .blks_to_clear;
;  506          0598  4          blks_used = .blks_used + .blks_to_clear;
;  507          0599  4
;  508          0600  3          END;
;  509          0601  3
;  510          0602  3      ! Truncate the file by moving any unused blocks to the empty directory entry which immediately follows t
;  511          0603  3      ! entry.
;  512          0604  3      !
;  513          0605  3      emp = .ent + rt11ent$k_length + .seg [rt11hdr$w_extra_bytes];              ! Pointer to the empty entry
;  514          0606  3      $logic_check (3, ((.emp [rt11ent$b_type_byte] EQL rt11ent$m_typ_empty) AND (.emp [rt11ent$w_blocks] EQL
;  515          0607  3      emp [rt11ent$w_blocks] = .ctx [rt11ctx$l_eof_block] -                   ! Count of leftovers (0 is o
;  516          0608  3                               .ctx [rt11ctx$l_high_block_written];
;  517          0609  3      ent [rt11ent$w_blocks] = .blks_used;
;  518        P 0610  3      $debug_print_fao ('used !UL,  left !UL,  eof !UL,  high !UL', .ent [rt11ent$w_blocks], .emp [rt11ent$w_b
;  519          0611  3                       .ctx [rt11ctx$l_eof_block], .ctx [rt11ctx$l_high_block_written]);
;  520          0612  3
;  521          0613  3      ! If there is another file with the same name around, we need to delete it now
;  522          0614  3      !
;  523          0615  3      IF .filb [filb$v_delete_previous]
;  524          0616  3      THEN
;  525          0617  4          BEGIN
;  526          0618  4          LOCAL
;  527          0619  4              ctx2 : $ref_bblock;
;  528          0620  4
;  529          0621  4          ctx2 = exch$util_rt11ctx_allocate (.volb, 0);   ! Get an RT11 context block
;  530          0622  4
;  531          0623  4          IF exch$rtacp_find_file (.ctx2, ctx [rt11ctx$t_exp_fullname], .ctx [rt11ctx$l_exp_fullname_len])
;  532          0624  4          THEN
;  533          0625  5              BEGIN
;  534          0626  5              BIND
;  535          0627  5                  copy = exch$a_gbl [excg$a_copy_work] : $ref_bblock,
;  536          0628  5                  ent2 = ctx2 [rt11ctx$a_ent_address] : $ref_bblock;
;  537        P 0629  5              $debug_print_fao ('deleting previous copy of ''AF''',
;  538          0630  5                                .filb [filb$l_result_name_len], filb [filb$t_result_name]);
;  539          0631  5              $logic_check (2, (NOT .ctx2 [rt11ctx$v_typ_protected]), 172);       ! Must be able to delete
;  540          0632  5              ent2 [rt11ent$b_type_byte] = rt11ent$m_typ_empty;           ! It is gone
;  541          0633  5              exch$rt11_dirseg_put (.volb, .ctx2 [rt11ctx$l_seg_number]);
;  542          0634  5              IF .copy [copy$v_q_log]
;  543          0635  5              THEN
;  544          0636  5                  $exch_signal (exch$_deleteprev, 2, .filb [filb$l_result_name_len], filb [filb$t_result_name]
;  545          0637  5              END
;  546          0638  4          ELSE
;  547          0639  4              $logic_check (3, (false), 171);
;  548          0640  4
;  549          0641  4          ! Return the context block
;  550          0642  4          !
;  551          0643  4          exch$util_rt11ctx_release (.ctx2);
;  552          0644  4
```

```
;  553          0645  3          END;
;  554          0646  3
;  555          0647  3      ! And finally, mark our new file as permanent and current, and write the directory segment
;  556          0648  3      !
;  557          0649  3      ent [rt11ent$v_type] = rt11ent$m_typ_permanent;     ! Mark only type field, protected bit might be set
;  558          0650  3      ent [rt11ent$b_job] = 1;                            ! Mark the entry as current
;  559          0651  3      exch$rt11_dirseg_put (.volb, .ctx [rt11ctx$l_seg_number]);
;  560          0652  3
;  561          0653  3      $logic_check (4, (exch$rtacp_verify_directory (.volb)), 187);
;  562          0654  2      END;
;  563          0655  2
;  564          0656  2  ! Clear the stream active bit and all other context flags
;  565          0657  2  !
;  566          0658  2  ctx [rt11ctx$l_flags] = 0;
;  567          0659  2
;  568          0660  2  RETURN true;
;  569          0661  1  END;


                                                .EXTRN  EXCH$_DELETEPREV

                                 0FFC 00000     .ENTRY  EXCH$RT11_CLOSE_FILE, Save R2,R3,R4,R5,R6,- ; 0477
                                                        R7,R8,R9,R10,R11
                        5E          04  C2 00002        SUBL2   #4, SP                                  ;
                        57          04  AC D0 00005     MOVL    FILB, R7                                ; 0517
                        52  035B00FA  8F D0 00009     MOVL    #56295674, R2                           ; 0524
                        51      01BF  8F 3C 00010     MOVZWL  #447, R1                                ;
                        50            57 D0 00015     MOVL    R7, R0                                  ;
                             00000000G  EF 16 00018     JSB     EXCH$UTIL_BLOCK_CHECK                   ;
                        52  010A00F7  8F D0 0001E     MOVL    #17432823, R2                           ; 0525
                        51      01C0  8F 3C 00025     MOVZWL  #448, R1                                ;
                        50        18  A7 D0 0002A     MOVL    24(R7), R0                              ;
                             00000000G  EF 16 0002E     JSB     EXCH$UTIL_BLOCK_CHECK                   ;
                        6E        1C  A7 D0 00034     MOVL    28(R7), (SP)                            ; 0526
                        52  041B00F3  8F D0 00038     MOVL    #68878579, R2                           ;
                        51      01C1  8F 3C 0003F     MOVZWL  #449, R1                                ;
                        50            6E D0 00044     MOVL    (SP), R0                                ;
                             00000000G  EF 16 00047     JSB     EXCH$UTIL_BLOCK_CHECK                   ;
                        56        20  A7 D0 0004D     MOVL    32(R7), R6                              ; 0527
                        52  008200F4  8F D0 00051     MOVL    #8519924, R2                            ;
                        51      01C3  8F 3C 00058     MOVZWL  #451, R1                                ;
                        50            56 D0 0005D     MOVL    R6, R0                                  ;
                             00000000G  EF 16 00060     JSB     EXCH$UTIL_BLOCK_CHECK                   ;
                        57        10  A6 D1 00066     CMPL    16(R6), R7                              ; 0528
                                      13 13 0006A     BEQL    1$                                      ;
                        7E        7A  8F 9A 0006C     MOVZBL  #122, -(SP)                             ;
                                      01 DD 00070     PUSHL   #1                                      ;
                             00000000G  8F DD 00072     PUSHL   #EXCH$_BADLOGIC                         ;
                  00000000G         00  03 FB 00078     CALLS   #3, LIB$STOP                            ;
                        6E        14  A6 D1 0007F 1$:  CMPL    20(R6), (SP)                            ; 0529
                                      13 13 00083     BEQL    2$                                      ;
                        7E        7B  8F 9A 00085     MOVZBL  #123, -(SP)                             ;
                                      01 DD 00089     PUSHL   #1                                      ;
                             00000000G  8F DD 0008B     PUSHL   #EXCH$_BADLOGIC                         ;
                  00000000G         00  03 FB 00091     CALLS   #3, LIB$STOP                            ;
                     03   28  A6        01  E0 00098 2$:  BBS     #1, 40(R6), 3$                          ; 0533
```

```
                                                013E  31  0009D          BRW     15$                                                     ;
                                                  56  DD  000A0  3$:     PUSHL   R6                                                      ; 0545
                                00000000G         EF                     01  FB  000A2          CALLS   #1, EXCH$PDP_FLUSH_WRITE_BUFFER                  ;
                                                  5A                     50  D0  000A9          MOVL    R0, STATUS                              ;
                                                  54                     5A  E9  000AC          BLBC    STATUS, 6$                              ;
                                     50     34    A6                 72  A6  C3  000AF          SUBL3   114(R6), 52(R6), R0                     ; 0551
                                                  5B                 01  A0  9E  000B5          MOVAB   1(R0), BLKS_USED                        ;
                                                  50                 2D  A7  D0  000B9          MOVL    45(R7), R0                              ; 0556
                                                  57                     13  BD  000BD          BEQL    9$                                      ;
                                                  5B                     50  D1  000BF          CMPL    R0, BLKS_USED                           ; 0558
                                                  52                     1B  C2  000C2          BLEQU   9$                                      ;
                                     59           50                     5B  C3  000C4          SUBL3   BLKS_USED, R0, BLKS_TO_CLEAR            ; 0568
                                     58     34    A6                     01  C1  000C8          ADDL3   #1, 52(R6), CUR_BLK                     ; 0569
                1800 8F              00           6E                     00  2C  000CD          MOVC5   #0, (SP), #0, #6144, @24(R6)            ; 0571
                                                                    18   B6  000D4                                                          ;
                                                  52                     59  D0  000D6          MOVL    BLKS_TO_CLEAR, BLK_CNT                  ; 0575
                                                                         52  D5  000D9  4$:     TSTL    BLK_CNT                                 ; 0576
                                                                         32  15  000DB          BLEQ    8$                                      ;
                                                                    18   A6  DD  000DD          PUSHL   24(R6)                                  ; 0582
                                                                         52  DD  000E0          PUSHL   BLK_CNT                                 ; 0581
                                                  0C                     6E  D1  000E2          CMPL    (SP), #12                               ;
                                                                         03  1B  000E5          BLEQU   5$                                      ;
                                                  6E                     0C  D0  000E7          MOVL    #12, (SP)                               ;
                                                                         58  DD  000EA  5$:     PUSHL   CUR_BLK                                 ; 0580
                                                                    0C   AE  DD  000EC          PUSHL   12(SP)                                  ; 0579
                                00000000G         EF                     04  FB  000EF          CALLS   #4, EXCH$IO_RT11_WRITE                  ;
                                                  5A                     50  D0  000F6          MOVL    R0, STATUS                              ;
                                                  0B                     5A  E8  000F9          BLBS    STATUS, 7$                              ;
                                                                         57  DD  000FC          PUSHL   R7                                      ; 0585
                                   FCDE           CF                     01  FB  000FE          CALLS   #1, EXCH$RT11_BAD_FILE                  ;
                                                  50                     5A  D0  00103  6$:     MOVL    STATUS, R0                              ; 0586
                                                                         04  00106            RET                                             ;
                                                  52                     0C  C2  00107  7$:     SUBL2   #12, BLK_CNT                            ; 0591
                                                  58                     0C  C0  0010A          ADDL2   #12, CUR_BLK                            ; 0592
                                                                         CA  11  0010D          BRB     4$                                      ; 0576
                                     34     A6                         59  C0  0010F  8$:     ADDL2   BLKS_TO_CLEAR, 52(R6)                   ; 0597
                                                  5B                     59  C0  00113          ADDL2   BLKS_TO_CLEAR, BLKS_USED                ; 0598
                                                  53                 7E  A6  D0  00116  9$:     MOVL    126(R6), R3                             ; 0605
                                                  50                 7A  A6  D0  0011A          MOVL    122(R6), R0                             ;
                                                  50                 06  A0  3C  0011E          MOVZWL  6(R0), R0                               ;
                                                  50                 0E  A340  9E  00122        MOVAB   14(R3)[R0], EMP                         ;
                08   A0         20     A6     34  A6  A3  00127          SUBW3   52(R6), 32(R6), 8(EMP)                                  ; 0608
                                 08     A3         5B  B0  0012E          MOVW    BLKS_USED, 8(R3)                                        ; 0609
                     03          2B     A7         06  E0  00132          BBS     #6, 43(R7), 10$                                         ; 0615
                                                 008F  31  00137          BRW     14$                                                     ;
                                                   7E  D4  0013A  10$:     CLRL    -(SP)                                                   ; 0621
                                                                    04   AE  DD  0013C          PUSHL   4(SP)                                   ;
                                00000000G         EF                     02  FB  0013F          CALLS   #2, EXCH$UTIL_RT11CTX_ALLOCATE          ;
                                                  52                     50  D0  00146          MOVL    R0, CTX2                                ;
                                                                    46   A6  DD  00149          PUSHL   70(R6)                                  ; 0623
                                                                    54   A6  9F  0014C          PUSHAB  84(R6)                                  ;
                                                                         52  DD  0014F          PUSHL   CTX2                                    ;
                                00000000G         EF                     03  FB  00151          CALLS   #3, EXCH$RTACP_FIND_FILE                ;
                                                  52                     50  E9  00158          BLBC    R0, 12$                                 ;
                             54 00000000G         EF                     04  C1  0015B          ADDL3   #4, EXCH$A_GBL, R4                      ; 0627
                                                                    39   A2  95  00163          TSTB    57(CTX2)                                ; 0631
                                                                         13  18  00166          BGEQ    11$                                     ;
```

```
                                7E        AC   8F  9A 00168          MOVZBL  #172, -(SP)                     ;
                                            01  DD 0016C            PUSHL   #1                              ;
                                00000000G   8F  DD 0016E            PUSHL   #EXCH$_BADLOGIC                 ;
                     00000000G  00            03  FB 00174          CALLS   #3, LIB$STOP                    ;
                                50        7E  A2  D0 0017B 11$:     MOVL    126(CTX2), R0                   ; 0632
                           01   A0            02  90 0017F          MOVB    #2, 1(R0)                       ;
                                          76  A2  DD 00183          PUSHL   118(CTX2)                       ; 0633
                                          04  AE  DD 00186          PUSHL   4(SP)                           ;
                          0000V CF            02  FB 00189          CALLS   #2, EXCH$RT11_DIRSEG_PUT        ;
                                50            64  D0 0018E          MOVL    (R4), R0                        ; 0634
                     2A      30   A0            03  E1 00191          BBC     #3, 48(R0), 13$                 ;
                                             5A  A7  9F 00196          PUSHAB  90(R7)                          ; 0636
                                             3A  A7  DD 00199          PUSHL   58(R7)                          ;
                                             02  DD 0019C            PUSHL   #2                              ;
                                00000000G    8F  DD 0019E            PUSHL   #EXCH$_DELETEPREV               ;
                     00000000G  00            04  FB 001A4          CALLS   #4, LIB$SIGNAL                  ;
                                            13  11  001AB          BRB     13$                             ; 0623
                                7E        AB  8F  9A 001AD 12$:     MOVZBL  #171, -(SP)                     ; 0639
                                            01  DD 001B1            PUSHL   #1                              ;
                                00000000G   8F  DD 001B3            PUSHL   #EXCH$_BADLOGIC                 ;
                     00000000G  00            03  FB 001B9          CALLS   #3, LIB$STOP                    ;
                                            52  DD 001C0 13$:     PUSHL   CTX2                            ; 0643
                     00000000G  EF            01  FB 001C2          CALLS   #1, EXCH$UTIL_RT11CTX_RELEASE   ;
     01   A3             04      00            04  F0 001C9 14$:     INSV    #4, #0, #4, 1(R3)               ; 0649
                           0B   A3            01  90 001CF          MOVB    #1, 11(R3)                      ; 0650
                                          76  A6  DD 001D3          PUSHL   118(R6)                         ; 0651
                                          04  AE  DD 001D6          PUSHL   4(SP)                           ;
                          0000V CF            02  FB 001D9          CALLS   #2, EXCH$RT11_DIRSEG_PUT        ;
                                          28  A6  D4 001DE 15$:     CLRL    40(R6)                          ; 0658
                                50            01  D0 001E1          MOVL    #1, R0                          ; 0660
                                                04 001E4          RET                                     ; 0661

; Routine Size:  485 bytes,    Routine Base:  EXCH$RT11_CODE + 021F
```

```
;   571     0662  1 GLOBAL ROUTINE exch$rt11_create_file =  %SBTTL 'exch$rt11_create_file'
;   572     0663  2 BEGIN
;   573     0664  2 !++
;   574     0665  2 !
;   575     0666  2 ! FUNCTIONAL DESCRIPTION:
;   576     0667  2 !
;   577     0668  2 !       Perform RT-11 volume specific create processing
;   578     0669  2 !
;   579     0670  2 ! INPUT:
;   580     0671  2 !
;   581     0672  2 !       none
;   582     0673  2 !
;   583     0674  2 ! IMPLICIT INPUTS:
;   584     0675  2 !
;   585     0676  2 !       copy [copy$a_out_filb] - pointer to the filb for the output file
;   586     0677  2 !       copy [copy$a_inp_filb] - pointer to the filb for the input file
;   587     0678  2 !
;   588     0679  2 ! OUTPUTS:
;   589     0680  2 !
;   590     0681  2 !       none
;   591     0682  2 !
;   592     0683  2 ! IMPLICIT OUTPUTS:
;   593     0684  2 !
;   594     0685  2 !       copy [copy$a_out_filb] - block receives info pertaining to the created file
;   595     0686  2 !
;   596     0687  2 !
;   597     0688  2 ! ROUTINE VALUE:
;   598     0689  2 !
;   599     0690  2 !       true if able to create a file, false otherwise
;   600     0691  2 !
;   601     0692  2 ! SIDE EFFECTS:
;   602     0693  2 !
;   603     0694  2 !       none
;   604     0695  2 !--
;   605     0696  2
;   606     0697  2 $dbgtrc_prefix ('rt11_create_file> ');
;   607     0698  2
;   608     0699  2 LOCAL
;   609     0700  2     rfp : $bblock [nam$c_bln+nam$c_maxrss],      ! output file parse - an RMS NAM block plus expanded string
;   610     0701  2     nam_len,                                    ! temporary to hold length of name
;   611     0702  2     typ_len,                                    ! temporary to hold length of type
;   612     0703  2     tot_len,                                    ! temporary to hold length of name + type
;   613     0704  2     ent : $ref_bblock,                          ! a pointer to the entry we are adding
;   614     0705  2     start_block,                                ! the pbn where this file will start
;   615     0706  2     blocks,
;   616     0707  2     physical,
;   617     0708  2     status
;   618     0709  2     ;
;   619     0710  2
;   620     0711  2 BIND
;   621     0712  2     copy     = exch$a_gbl [excg$a_copy_work]    : $ref_bblock,
;   622     0713  2     out_name = copy [copy$q_output_filename]    : $desc_block,
;   623     0714  2     inp_filb = copy [copy$a_inp_filb]           : $ref_bblock,
;   624     0715  2     inp_namb = inp_filb [filb$a_assoc_namb]     : $ref_bblock,
;   625     0716  2     inp_ctx  = inp_filb [filb$a_context]        : $ref_bblock,
;   626     0717  2     out_filb = copy [copy$a_out_filb]           : $ref_bblock,
;   627     0718  2     out_namb = out_filb [filb$a_assoc_namb]     : $ref_bblock,
```

```
;  628          0719  2      out_ctx  = out_filb [filb$a_context]        : $ref_bblock,
;  629          0720  2      volb     = out_filb [filb$a_assoc_volb]     : $ref_bblock
;  630          0721  2      ;
;  631          0722  2
;  632          0723  2 $debug_print_lit ('entry');
;  633          0724  2
;  634          0725  2 $block_check (1, .out_filb, filb, 426);
;  635          0726  2 $block_check (1, .inp_filb, filb, 536);
;  636          0727  2 $block_check (2, .out_namb, namb, 491);
;  637          0728  2 $block_check (2, .inp_namb, namb, 492);
;  638          0729  2 $block_check (2, .volb, volb, 531);
;  639          0730  2 $logic_check (4, (exch$rtacp_verify_directory (.volb)), 188);
```

```
;  641     0731  2 ! Make certain that write access is permitted, this should be checked long before we get here
;  642     0732  2 !
;  643     0733  2 $logic_check (1, (.volb [volb$v_write]), 166);
;  644     0734  2
;  645     0735  2 ! If the context pointer is null, then allocate and initialize it.
;  646     0736  2 !
;  647     0737  2 IF .out_ctx EQL 0
;  648     0738  2 THEN
;  649     0739  2     out_ctx = exch$util_rt11ctx_allocate (.volb, .out_filb)     ! Get an RT11 context block
;  650     0740  2
;  651     0741  2 ELSE
;  652     0742  2     $block_check (2, .out_ctx, rt11ctx, 534);                   ! Make sure that it is what we think it is
;  653     0743  2
;  654     0744  2 ! Make sure that we haven't crossed signals someplace
;  655     0745  2 !
;  656     0746  2 $logic_check (4, (.out_ctx [rt11ctx$a_assoc_filb] EQL .out_filb), 162);
;  657     0747  2 $logic_check (4, (.out_ctx [rt11ctx$a_assoc_volb] EQL .volb), 163);
;  658     0748  2
;  659     0749  2 ! Set the rest of the block to nulls, nothing carries over from one output file to the next
;  660     0750  2 !
;  661     0751  2 CH$FILL (0, rt11ctx$k_end_zero - rt11ctx$k_start_zero,  ! Set rest of block to nulls
;  662     0752  2                 .out_ctx + rt11ctx$k_start_zero);
;  663     0753  2
;  664     0754  2 ! Perform an RMS output file parse on the related name (the result name for the input file) and the
;  665     0755  2 ! requested output name from the command line.
;  666     0756  2 !
;  667     0757  3 IF NOT (status = exch$cmd_related_file_parse (
;  668     0758  3                 .out_name [dsc$w_length], .out_name [dsc$a_pointer],                ! Command line out p
;  669     0759  3                 .inp_filb [filb$l_result_name_len], inp_filb [filb$t_result_name],  ! Related name
;  670     0760  3                 rfp))                                                               ! Gets new name
;  671     0761  2 THEN
;  672     0762  3     BEGIN
;  673     0763  3
;  674     0764  3     ! Move the raw name to where it is accessible for the outer signal
;  675     0765  3     !
;  676     0766  3     CH$MOVE (.out_name [dsc$w_length], .out_name [dsc$a_pointer], out_filb [filb$t_result_name]);
;  677     0767  3     RETURN .status;
;  678     0768  3
;  679     0769  2     END;
;  680     0770  2
;  681     0771  2 ! Create the result file name in the filb
;  682     0772  2 !
;  683     0773  2 out_filb [filb$v_name_change] = false;          ! Assume no name change
;  684     0774  2 tot_len = .rfp [nam$b_name];                   ! Remember starting length
;  685     0775  2 rfp [nam$b_name] = exch$pdp_filter_filename (.rfp [nam$b_name], .rfp [nam$l_name]);     ! Remove invalid cha
;  686     0776  2 nam_len = MINU (.rfp [nam$b_name], 6);         ! Maximum name is six letters
;  687     0777  2 IF .tot_len NEQ .nam_len                       ! If final length not same as initial, then it has changed
;  688     0778  2 THEN
;  689     0779  2     out_filb [filb$v_name_change] = true;
;  690     0780  2
;  691     0781  2 tot_len = .rfp [nam$b_type];                   ! Remember starting length of type field
;  692     0782  2 rfp [nam$b_type] = 1 + exch$pdp_filter_filename (.rfp [nam$b_type] - 1, .rfp [nam$l_type] + 1);
;  693     0783  2 typ_len = MINU (.rfp [nam$b_type], 4);         ! Maximum type is three (plus the separating dot)
;  694     0784  2 IF .tot_len NEQ .typ_len
;  695     0785  2 THEN
;  696     0786  2     out_filb [filb$v_name_change] = true;
;  697     0787  2
```

```
;  698          0788  2 tot_len = .nam_len + .typ_len;                      ! Final length of both
;  699          0789  2 out_filb [filb$l_result_name_len] = .volb [volb$l_vol_ident_len] + .tot_len;    ! Length of volume ident
;  700          0790  2 $logic_check (2, (.out_filb [filb$l_result_name_len] EQU filb$s_result_name), 164);
;  701          0791  2 CH$COPY (.volb [volb$l_vol_ident_len], volb [volb$t_vol_ident],                  ! Volume name
;  702          0792  2                 .nam_len, .rfp [nam$l_name], .typ_len, .rfp [nam$l_type],
;  703          0793  2                 0, filb$s_result_name, out_filb [filb$t_result_name]);
;  704          0794  2
;  705          0795  2 ! Do not create a .BAD file unless this is an explicit copy
;  706          0796  2 !
;  707          0797  2 IF CH$EQL (4, UPLIT BYTE ('.BAD'), .typ_len, .rfp [nam$l_type])
;  708          0798  2 THEN
;  709          0799  2     BEGIN
;  710          0800  2     IF  .inp_namb [namb$v_wild_name] OR .inp_namb [namb$v_wild_type]    ! Wild in input?
;  711          0801  2       OR
;  712          0802  2         .out_namb [namb$v_wild_name] OR .out_namb [namb$v_wild_type]    ! Wild in output?
;  713          0803  2     THEN
;  714          0804  2         RETURN exch$_nocopbad;
;  715          0805  2     END;
;  716          0806  2
;  717          0807  2 $debug_print_fao ('Looking for "!AF"', .out_filb [filb$l_result_name_len], out_filb [filb$t_result_name]);
;  718          0808  2
;  719          0809  2 ! See if we will have to delete a same-named file later on
;  720          0810  2 !
;  721          0811  2 out_filb [filb$v_delete_previous] = false;       ! Assume we won't have to delete
;  722          0812  2 IF exch$rtacp_find_file (.out_ctx, out_filb [filb$t_result_name] + .volb [volb$l_vol_ident_len], .tot_len)
;  723          0813  2 THEN
;  724          0814  2     BEGIN
;  725          0815  2     LOCAL
;  726          0816  2         retstat,                                  ! Help BLISS figure out the two signals are the same
;  727          0817  2         sigstat;
;  728          0818  2
;  729          0819  3     $debug_print_fao ('file "!AF" exists', .out_filb [filb$l_result_name_len], out_filb [filb$t_result_name]
;  730          0820  3     $logic_check (2, (.inp_ctx NEQ 0), 209);
;  731          0821  3
;  732          0822  3     ! If the input and output files are identical during a wildcard copy, this is illegal
;  733          0823  3     !
;  734          0824  3     IF .inp_ctx [rt11ctx$b_type] EQL exchblk$k_rt11ctx          ! Input must also be RT-11
;  735          0825  3     THEN
;  736          0826  3
;  737          0827  3         ! If the directory entry addresses are the same, this is in fact the same file.  Note that we are
;  738          0828  3         ! assuming that nothing has happened which might have restructured the directory.
;  739          0829  3         !
;  740          0830  3         IF .inp_ctx [rt11ctx$a_ent_address] EQL .out_ctx [rt11ctx$a_ent_address]
;  741          0831  3         THEN
;  742          0832  3             IF  .inp_namb [namb$v_wild_name] OR .inp_namb [namb$v_wild_type]    ! Wild in input?
;  743          0833  3               OR
;  744          0834  3                 .copy [copy$v_q_replace]
;  745          0835  3             THEN
;  746          0836  3                 RETURN exch$_nocopsamdev;
;  747          0837  3
;  748          0838  3     ! Verify that it is ok to delete the existing file
;  749          0839  3     !
;  750          0840  3     IF .out_ctx [rt11ctx$b_job] NEQ 0           ! Can't delete protected files
;  751          0841  3     THEN
;  752          0842  3         RETURN exch$_nocopdup;
;  753          0843  3
;  754          0844  3     IF .out_ctx [rt11ctx$v_typ_protected]       ! Can't delete protected files
```

```
;  755     0845  3     THEN
;  756     0846  3         RETURN exch$_nocopprot;
;  757     0847  3
;  758     0848  3     IF NOT .copy [copy$v_q_delete]              ! /NODELETE has been requested, don't do it
;  759     0849  3     THEN
;  760     0850  3         RETURN exch$_nocopnodel;
;  761     0851  3
;  762     0852  3     IF .out_ctx [rt11ctx$w_filetype] EQL r50_bad        ! Cannot delete a file with .BAD extension during a
;  763     0853  3     THEN
;  764     0854  3         RETURN exch$_nocopbaddel;
;  765     0855  3
;  766     0856  3     IF  .out_ctx [rt11ctx$w_filetype] EQL r50_sys       ! Cannot delete a file with .SYS extension during a
;  767     0857  3       AND                                               !  unless
;  768     0858  3         NOT .copy [copy$v_q_system]                     !  /SYSTEM has been specified
;  769     0859  3     THEN
;  770     0860  3         RETURN exch$_nocopsysdel;
;  771     0861  3
;  772     0862  3     ! If a delete-before-write operation is requested, delete this file now
;  773     0863  3     !
;  774     0864  3     IF .copy [copy$v_q_replace]
;  775     0865  3     THEN
;  776     0866  4         BEGIN
;  777     0867  4         BIND
;  778     0868  4             ent = out_ctx [rt11ctx$a_ent_address] : $ref_bblock;
;  779   P 0869  4         $debug_print_fao ('deleting previous copy of "!AF"
;  780     0870  4                                 .out_filb [filb$l_result_name_len], out_filb [filb$t_result_name
;  781     0871  4         $logic_check (2, (NOT .out_ctx [rt11ctx$v_typ_protected]), 179);        ! Must be able to delete
;  782     0872  4         ent [rt11ent$b_type_byte] = rt11ent$m_typ_empty;                        ! It is gone
;  783     0873  4         exch$rt11_dirseg_put (.volb, .out_ctx [rt11ctx$l_seg_number]);  ! Write the directory segment
;  784     0874  4         IF .copy [copy$v_q_log]
;  785     0875  4         THEN
;  786     0876  4             $exch_signal (exch$_deleteprev, 2, .out_filb [filb$l_result_name_len], out_filb [filb$t_result_n
;  787     0877  4         END
;  788     0878  4
;  789     0879  4     ! otherwise remember that we have some extra work to do when we close the file
;  790     0880  4     !
;  791     0881  3     ELSE
;  792     0882  3         out_filb [filb$v_delete_previous] = true;
;  793     0883  3
;  794     0884  2     END;
;  795     0885  2
;  796     0886  2 ! Reset the rest of the block to nulls, nothing carries over from before
;  797     0887  2 !
;  798     0888  2 CH$FILL (0, rt11ctx$k_end_zero - rt11ctx$k_start_zero,  ! Set rest of block to nulls
;  799     0889  2                 .out_ctx + rt11ctx$k_start_zero);
;  800     0890  2
;  801     0891  2 ! If a /ALLOCATION qualifier has been seen, use that value.  If BLOCKS ends up with the value 0, then
;  802     0892  2 ! we will get the largest area on the volume.
;  803     0893  2 !
;  804     0894  3 blocks = (IF .inp_filb [filb$l_q_allocation] NEQ 0       ! If specified on the input
;  805     0895  3           THEN                                          !  then
;  806     0896  3                .inp_filb [filb$l_q_allocation]          !   use that quantity
;  807     0897  3           ELSE                                          !  otherwise
;  808     0898  2                .copy [copy$l_q_allocation]);            !   use the quantity from the output
;  809     0899  2
;  810     0900  2 out_filb [filb$l_q_allocation] = .blocks;        ! Save the value so that we can look at it during the close
;  811     0901  2
```

```
;   812     0902  2 ! Make sure that the record format in the filb is correct
;   813     0903  2 !
;   814     0904  2 exch$cmd_fetch_recfmt_implied (.out_filb, .rfp [nam$l_type]+1); ! Pass it the type from the parse
;   815     0905  2
;   816     0906  2 ! Save the addresses of our routines for this volume and record format.
;   817     0907  2 !
;   818     0908  2 out_filb [filb$a_close_routine] = exch$rt11_close_file;
;   819     0909  2 out_filb [filb$a_delete_routine] = exch$rt11_delete_file;
;   820     0910  2 out_filb [filb$a_get_routine] = 0;
;   821     0911  2 out_filb [filb$a_put_routine] = exch$pdp_put;
;   822     0912  2
;   823     0913  2 ! Carriage control doesn't mean anything for RT-11 output, tell him we are ignoring
;   824     0914  2 !
;   825     0915  2 IF .inp_filb [filb$v_cctl_explicit]
;   826     0916  2  OR
;   827     0917  2     .out_filb [filb$v_cctl_explicit]
;   828     0918  2 THEN
;   829     0919  2     $exch_signal (exch$_nocarriage);
;   830     0920  2
;   831     0921  2 physical = false;                               ! Assume not using physical transfers
;   832     0922  2
;   833     0923  2 ! For RT-11 we can treat block transfer mode as fixed 512, physical
;   834     0924  2 !
;   835     0925  2 IF .out_filb [filb$b_transfer_mode] EQL filb$k_xfrm_block
;   836     0926  2   OR
;   837     0927  2     .inp_filb [filb$b_transfer_mode] EQL filb$k_xfrm_block
;   838     0928  2 THEN
;   839     0929  3     BEGIN
;   840     0930  3     physical = true;
;   841     0931  3     out_filb [filb$b_rec_format]  = filb$k_rfmt_fixed;
;   842     0932  3     out_filb [filb$l_fixed_len]   = 512;
;   843     0933  2     END;
;   844     0934  2
;   845     0935  2 ! If an explicit record format was given on the input but none on the output, carry the input to the output
;   846     0936  2 !
;   847     0937  2 IF .inp_filb [filb$v_rfmt_explicit]             ! The input file has explicit format
;   848     0938  3    AND (NOT .out_filb [filb$v_rfmt_explicit])   !  but the output has implied record format
;   849     0939  2 THEN
;   850     0940  3     BEGIN
;   851     0941  3     out_filb [filb$b_rec_format]  = .inp_filb [filb$b_rec_format];
;   852     0942  3     out_filb [filb$l_fixed_len]   = .inp_filb [filb$l_fixed_len];
;   853     0943  2     END;
;   854     0944  2
;   855     0945  2 ! In some circumstances we can do block mode I/O rather than record mode
;   856     0946  2 !
;   857     0947  3 IF  (.inp_filb [filb$b_transfer_mode] EQL filb$k_xfrm_automatic ! Both input and output must be automatic tr
;   858     0948  3      AND .out_filb [filb$b_transfer_mode] EQL filb$k_xfrm_automatic)
;   859     0949  2   AND
;   860     0950  4     ( NOT (.inp_filb [filb$v_rfmt_explicit]                    ! Both the input and output files must have
;   861     0951  3          OR .out_filb [filb$v_rfmt_explicit]))                 !  implied record formats
;   862     0952  2   AND
;   863     0953  3     (.inp_namb [namb$b_vol_format] EQL volb$k_vfmt_rt11        ! The input must be RT-11
;   864     0954  3       OR .inp_namb [namb$b_vol_format] EQL volb$k_vfmt_dos11)   !  or DOS-11
;   865     0955  2 THEN
;   866     0956  3     BEGIN
;   867     0957  3     inp_filb [filb$b_rec_format]  = filb$k_rfmt_fixed;
;   868     0958  3     inp_filb [filb$l_fixed_len]   = 512;
```

```
;  869     0959  3     out_filb [filb$b_rec_format]  = filb$k_rfmt_fixed;
;  870     0960  3     out_filb [filb$l_fixed_len]   = 512;
;  871     0961  2     END;
;  872     0962  2
;  873     0963  2 ! A block request of zero means grab the largest space on the volume.  Let's see if we can determine the exa
;  874     0964  2 ! block count we will need.
;  875     0965  2 !
;  876     0966  2 IF .blocks EQL 0
;  877     0967  2 THEN
;  878     0968  2     IF .physical                                                                ! Physical works fine
;  879     0969  2       OR
;  880     0970  3         (
;  881     0971  4             (.inp_filb [filb$b_rec_format]  EQL filb$k_rfmt_fixed)            ! If both input and output a
;  882     0972  3          AND
;  883     0973  4             (.out_filb [filb$b_rec_format]  EQL filb$k_rfmt_fixed)            !  fixed-record files and th
;  884     0974  3          AND
;  885     0975  4             (.inp_filb [filb$l_fixed_len] EQL .out_filb [filb$l_fixed_len])   !  record lengths are identi
;  886     0976  3         )
;  887     0977  2     THEN
;  888     0978  2         blocks = .inp_filb [filb$l_block_count];
;  889     0979  2
;  890     0980  2 ! Get some empty area on the volume
;  891     0981  2 !
;  892     0982  3 IF NOT (status = exch$rtacp_find_empty_area (.out_ctx, .blocks, .copy [copy$l_q_start_block]))
;  893     0983  2 THEN
;  894     0984  2     RETURN .status;
;  895     0985  2
;  896     0986  2 ! Set the entry up as a tentative file
;  897     0987  2 !
;  898     0988  2 ent = .out_ctx [rt11ctx$a_ent_address];                 ! Get the entry pointer into a place where we can us
;  899     0989  2 ent [rt11ent$b_type_byte] = rt11ent$m_typ_tentative;    ! Tentative entry
;  900     0990  2
;  901     0991  2 ! Set the protection attribute of the file.  If specified, use that value.  Otherwise, if input file is RT-1
;  902     0992  2 ! use the attribute of the input.
;  903     0993  2 !
;  904     0994  2 IF .copy [copy$v_q_protect_explicit]            ! If /PROTECT or /NOPROTECT was explicitly specified
;  905     0995  2 THEN
;  906     0996  2     ent [rt11ent$v_typ_protected] = .copy [copy$v_q_protect]
;  907     0997  2 ELSE
;  908     0998  3     BEGIN
;  909     0999  3     IF .inp_ctx NEQ 0
;  910     1000  3     THEN
;  911     1001  3         IF .inp_ctx [rt11ctx$b_type] EQL exchblk$k_rt11ctx              ! Input must also be RT-11
;  912     1002  3         THEN
;  913     1003  3             ent [rt11ent$v_typ_protected] = .inp_ctx [rt11ctx$v_typ_protected];
;  914     1004  2     END;
;  915     1005  2
;  916     1006  2 ! Get the date into RT11 format
;  917     1007  2 !
;  918     1008  2 exch$rt11_format_current_date (.ent);
;  919     1009  2
;  920     1010  2 ! Convert the file name to radix 50 and store in the entry
;  921     1011  2 !
;  922     1012  2 exch$util_radix50_from_ascii (.nam_len, .rfp [nam$l_name],
;  923     1013  2                                             rt11ctx$s_exp_name, ent [rt11ent$l_filename]);
;  924     1014  2 exch$util_radix50_from_ascii (.typ_len-1, .rfp [nam$l_type]+1,
;  925     1015  2                                             rt11ctx$s_exp_type, ent [rt11ent$w_filetype]);
```

```
;  926          1016  2
;  927          1017  2 ! Now force the modified entry to disk
;  928          1018  2 !
;  929          1019  2 exch$rt11_dirseg_put (.volb, .out_ctx [rt11ctx$l_seg_number]);
;  930          1020  2 CH$MOVE (rt11ent$k_length, .ent, out_ctx [rt11ctx$t_entry]);    ! Put a fresh copy into the context block
;  931          1021  2
;  932          1022  2 ! Define a record stream for this file
;  933          1023  2 !
;  934          1024  2 out_ctx [rt11ctx$l_cur_byte]    = 0;                            ! Context is the first byte in
;  935          1025  2 out_ctx [rt11ctx$l_cur_block]   = .out_ctx [rt11ctx$l_start_block];     !  the first block of the file
;  936          1026  2 out_ctx [rt11ctx$l_eof_block]   = .out_ctx [rt11ctx$l_start_block] + .out_ctx [rt11ctx$w_blocks] - 1;
;  937          1027  2 out_filb [filb$a_record]        = 0;                            ! No valid record or length
;  938          1028  2 out_filb [filb$l_record_len]    = 0;
;  939          1029  2
;  940          1030  2 ! Expand the radix-50 filename into the standard ascii text fields
;  941          1031  2 !
;  942          1032  2 exch$rt11_expand_filename (.out_ctx);
;  943          1033  2
;  944          1034  2 ! Clear all the flags except the ones we want by writing the masks into the longword
;  945          1035  2 !
;  946          1036  2 out_ctx [rt11ctx$l_flags] =    rt11ctx$m_stream_active  ! A record stream is currently active
;  947          1037  2                           OR rt11ctx$m_output_file;   !  and it is an output file
;  948          1038  2
;  949          1039  2 ! Set up the i/o and record buffer
;  950          1040  2 !
;  951          1041  2 IF .out_ctx [rt11ctx$a_buffer] EQL 0
;  952          1042  2 THEN
;  953          1043  2     out_ctx [rt11ctx$a_buffer] = exch$util_vm_allocate (ctx$k_buffer_length);
;  954          1044  2
;  955          1045  2 ! Set the block pointers to the chunk we are ready to write (i.e. nothing, 'cuz we've done no puts)
;  956          1046  2 !
;  957          1047  2 blocks = MINU (.out_ctx [rt11ctx$w_blocks], ctx$k_buffer_blocks);
;  958          1048  2 out_ctx [rt11ctx$l_buf_base_block] = .out_ctx [rt11ctx$l_start_block];
;  959          1049  2 out_ctx [rt11ctx$l_buf_high_block] = .out_ctx [rt11ctx$l_start_block] + .blocks - 1;
;  960          1050  2 out_ctx [rt11ctx$l_high_block_written] = .out_ctx [rt11ctx$l_start_block] - 1;
;  961          1051  2
;  962          1052  2 $logic_check (3, (exch$rtacp_verify_directory (.volb)), 189);
;  963          1053  2
;  964          1054  2 RETURN true;
;  965          1055  1 END;


                                                        .PSECT  EXCH$RT11_PLIT,NOWRT,2

                                    44  41  42  2E  00000 P.AAA:  .ASCII  \.BAD\                                        ;

                                                        .EXTRN  EXCH$_NOCOPBAD, EXCH$_NOCOPSAMDEV
                                                        .EXTRN  EXCH$_NOCOPDUP, EXCH$_NOCOPPROT
                                                        .EXTRN  EXCH$_NOCOPNODEL
                                                        .EXTRN  EXCH$_NOCOPBADDEL
                                                        .EXTRN  EXCH$_NOCOPSYSDEL
                                                        .EXTRN  EXCH$_NOCARRIAGE

                                                        .PSECT  EXCH$RT11_CODE,NOWRT,2

                                            0FFC 00000  .ENTRY  EXCH$RT11_CREATE_FILE, Save R2,R3,R4,R5,R6,-; 0662
```

```
                                                                   R7,R8,R9,R10,R11                      ;
                      5E        FE7C   CE  9E 00002         MOVAB  -388(SP), SP                          ;
          50 00000000G  EF             04  C1 00007         ADDL3  #4, EXCH$A_GBL, R0                    ; 0712
                      59               60  D0 0000F         MOVL   (R0), R9                              ; 0713
                      5A          14   A9  9E 00012         MOVAB  20(R9), R10                           ;
                      58          3C   A9  D0 00016         MOVL   60(R9), R8                            ; 0715
                      56          44   A9  D0 0001A         MOVL   68(R9), R6                            ; 0718
                      52   035B00FA    8F  D0 0001E         MOVL   #56295674, R2                         ; 0725
                      51        01AA   8F  3C 00025         MOVZWL #426, R1                              ;
                      50               56  D0 0002A         MOVL   R6, R0                                ;
                         00000000G     EF  16 0002D         JSB    EXCH$UTIL_BLOCK_CHECK                 ;
                      52   035B00FA    8F  D0 00033         MOVL   #56295674, R2                         ; 0726
                      51        0218   8F  3C 0003A         MOVZWL #536, R1                              ;
                      50               58  D0 0003F         MOVL   R8, R0                                ;
                         00000000G     EF  16 00042         JSB    EXCH$UTIL_BLOCK_CHECK                 ;
                04    AE          18   A6  D0 00048         MOVL   24(R6), 4(SP)                         ; 0727
                      52   010A00F7    8F  D0 0004D         MOVL   #17432823, R2                         ;
                      51        01EB   8F  3C 00054         MOVZWL #491, R1                              ;
                      50          04   AE  D0 00059         MOVL   4(SP), R0                             ;
                         00000000G     EF  16 0005D         JSB    EXCH$UTIL_BLOCK_CHECK                 ;
                      5B          18   A8  D0 00063         MOVL   24(R8), R11                           ; 0728
                      52   010A00F7    8F  D0 00067         MOVL   #17432823, R2                         ;
                      51        01EC   8F  3C 0006E         MOVZWL #492, R1                              ;
                      50               5B  D0 00073         MOVL   R11, R0                               ;
                         00000000G     EF  16 00076         JSB    EXCH$UTIL_BLOCK_CHECK                 ;
                      6E          1C   A6  D0 0007C         MOVL   28(R6), (SP)                          ; 0729
                      52   041B00F3    8F  D0 00080         MOVL   #68878579, R2                         ;
                      51        0213   8F  3C 00087         MOVZWL #531, R1                              ;
                      50               6E  D0 0008C         MOVL   (SP), R0                              ;
                         00000000G     EF  16 0008F         JSB    EXCH$UTIL_BLOCK_CHECK                 ;
          50          6E   00000048    8F  C1 00095         ADDL3  #72, (SP), R0                         ; 0733
          13          60               05  E0 0009D         BBS    #5, (R0), 1$                          ;
                      7E          A6   8F  9A 000A1         MOVZBL #166, -(SP)                           ;
                                       01  DD 000A5         PUSHL  #1                                    ;
                         00000000G     8F  DD 000A7         PUSHL  #EXCH$_BADLOGIC                       ;
         00000000G    00               03  FB 000AD         CALLS  #3, LIB$STOP                          ;
                                  20   A6  D5 000B4  1$:    TSTL   32(R6)                                ; 0737
                                  12   12 000B7             BNEQ   2$                                    ;
                                       56  DD 000B9         PUSHL  R6                                    ; 0739
                                  04   AE  DD 000BB         PUSHL  4(SP)                                 ;
         00000000G    EF               02  FB 000BE         CALLS  #2, EXCH$UTIL_RT11CTX_ALLOCATE        ;
                20    A6               50  D0 000C5         MOVL   R0, 32(R6)                            ;
                                       16  11 000C9         BRB    3$                                    ;
                      52   008200F4    8F  D0 000CB  2$:    MOVL   #8519924, R2                          ; 0742
                      51        0216   8F  3C 000D2         MOVZWL #534, R1                              ;
                      50          20   A6  D0 000D7         MOVL   32(R6), R0                            ;
                         00000000G     EF  16 000DB         JSB    EXCH$UTIL_BLOCK_CHECK                 ;
                      57          20   A6  D0 000E1  3$:    MOVL   32(R6), R7                            ; 0752
  0066   8F       00  6E               00  2C 000E5         MOVC5  #0, (SP), #0, #102, 28(R7)            ;
                                  1C   A7     000EC                                                      ;
                                  24   AE  9F 000EE         PUSHAB RFP                                   ; 0759
                                  5A   A8  9F 000F1         PUSHAB 90(R8)                                ;
                                  3A   A8  DD 000F4         PUSHL  58(R8)                                ;
                                  04   AA  DD 000F7         PUSHL  4(R10)                                ;
                      7E               6A  3C 000FA         MOVZWL (R10), -(SP)                          ;
         00000000G    EF               05  FB 000FD         CALLS  #5, EXCH$CMD_RELATED_FILE_PARSE       ;
                20    AE               50  D0 00104         MOVL   R0, STATUS                            ;
```

```
                                09    20   AE   E8 00108          BLBS    STATUS, 4$                                                          ;
                5A   A6         04    BA        6A 28 0010C          MOVC3   (R10), @4(R10), 90(R6)                                        ; 0766
                                     032A        31 00112          BRW     36$                                                         ; 0767
                                14    AE   2B   A6 9E 00115 4$:       MOVAB   43(R6), 20(SP)                                              ; 0773
                                14    BE   80   8F 8A 0011A          BICB2   #128, @20(SP)                                               ;
                                5A         5F   AE 9A 0011F          MOVZBL  RFP+59, TOT_LEN                                             ; 0774
                                           70   AE DD 00123          PUSHL   RFP+76                                                      ; 0775
                                7E         63   AE 9A 00126          MOVZBL  RFP+59, -(SP)                                               ;
                     00000000G  EF              02 FB 0012A          CALLS   #2, EXCH$PDP_FILTER_FILENAME                                ;
                           5F   AE              50 90 00131          MOVB    R0, RFP+59                                                  ;
                                50         5F   AE 9A 00135          MOVZBL  RFP+59, R0                                                  ; 0776
                                06              50 91 00139          CMPB    R0, #6                                                      ;
                                                03 1B 0013C          BLEQU   5$                                                          ;
                                50              06 D0 0013E          MOVL    #6, R0                                                      ;
                                1C    AE        50 D0 00141 5$:       MOVL    R0, NAM_LEN                                                 ;
                                1C    AE             5A D1 00145          CMPL    TOT_LEN, NAM_LEN                                            ; 0777
                                                05 13 00149          BEQL    6$                                                          ;
                                14    BE   80   8F 88 0014B          BISB2   #128, @20(SP)                                               ; 0779
                                5A         60   AE 9A 00150 6$:       MOVZBL  RFP+60, TOT_LEN                                             ; 0781
                           7E   74    AE         01 C1 00154          ADDL3   #1, RFP+80, -(SP)                                           ; 0782
                                7E         64   AE 9A 00159          MOVZBL  RFP+60, -(SP)                                               ;
                                                6E D7 0015D          DECL    (SP)                                                        ;
                     00000000G  EF              02 FB 0015F          CALLS   #2, EXCH$PDP_FILTER_FILENAME                                ;
                60   AE              50         01 81 00166          ADDB3   #1, R0, RFP+60                                              ;
                                50         60   AE 9A 0016B          MOVZBL  RFP+60, R0                                                  ; 0783
                                04              50 91 0016F          CMPB    R0, #4                                                      ;
                                                03 1B 00172          BLEQU   7$                                                          ;
                                50              04 D0 00174          MOVL    #4, R0                                                      ;
                                18    AE        50 D0 00177 7$:       MOVL    R0, TYP_LEN                                                 ;
                                18    AE             5A D1 0017B          CMPL    TOT_LEN, TYP_LEN                                            ; 0784
                                                05 13 0017F          BEQL    8$                                                          ;
                                14    BE        80   8F 88 00181          BISB2   #128, @20(SP)                                               ; 0786
                           5A   1C    AE        18   AE C1 00186 8$:       ADDL3   TYP_LEN, NAM_LEN, TOT_LEN                           ; 0788
                           50              6E 00000065   8F C1 0018C          ADDL3   #101, (SP), R0                                      ; 0789
                     3A    A6              60         5A C1 00194          ADDL3   TOT_LEN, (R0), 58(R6)                               ;
                     00000100   8F         3A    A6   D1 00199          CMPL    58(R6), #256                                        ; 0790
                                                13 1B 001A1          BLEQU   9$                                                          ;
                                7E         A4   8F 9A 001A3          MOVZBL  #164, -(SP)                                                 ;
                                                01 DD 001A7          PUSHL   #1                                                          ;
                                     00000000G  8F DD 001A9          PUSHL   #EXCH$_BADLOGIC                                             ;
                           00000000G  00              03 FB 001AF          CALLS   #3, LIB$STOP                                        ;
                                0C    AE   0100   8F 3C 001B6 9$:       MOVZWL  #256, 12(SP)                                        ; 0791
                                10    AE        5A   A6 9E 001BC          MOVAB   90(R6), 16(SP)                                      ; 0793
                                08    AE        10   AE D0 001C1          MOVL    16(SP), 8(SP)                                       ;
                           7E              6E 00000069   8F C1 001C6          ADDL3   #105, (SP), -(SP)                                   ;
                           7E         04   AE 00000065   8F C1 001CE          ADDL3   #101, 4(SP), -(SP)                                  ;
          0C   AE          00              9E         9E 2C 001D7          MOVC5   @(SP)+, @(SP)+, #0, 12(SP), @8(SP)                  ;
                                                08 BE    001DD                                                                          ;
                                                38 18 001DF          BGEQ    10$                                                         ;
                           7E              6E 00000065   8F C1 001E1          ADDL3   #101, (SP), -(SP)                                   ;
                                08    AE             9E C0 001E9          ADDL2   @(SP)+, 8(SP)                                       ;
                           7E              6E 00000065   8F C1 001ED          ADDL3   #101, (SP), -(SP)                                   ;
                                0C    AE             9E C2 001F5          SUBL2   @(SP)+, 12(SP)                                      ;
          0C   AE          00         70   BE   1C   AE 2C 001F9          MOVC5   NAM_LEN, @RFP+76, #0, 12(SP), @8(SP)                ;
                                                08 BE    00201                                                                          ;
                                                14 18 00203          BGEQ    10$                                                         ;
                                08    AE        1C   AE C0 00205          ADDL2   NAM_LEN, 8(SP)                                      ;
```

```
                        0C  AE        1C  AE  C2 0020A          SUBL2   NAM_LEN, 12(SP)                                                    ;
    0C  AE          00  74  BE        18  AE  2C 0020F          MOVC5   TYP_LEN, @RFP+80, #0, 12(SP), @8(SP)                               ;
                                      08  BE     00217                                                                                   ;
    18  AE          00  0000'  CF         04  2D 00219 10$:     CMPC5   #4, P.AAA, #0, TYP_LEN, @RFP+80                            ; 0797
                                      74  BE     00221                                                                                   ;
                                          2C 12 00223          BNEQ    12$                                                                ;
                    1F  6C  AB            01  E0 00225          BBS     #1, 108(R11), 11$                                          ; 0800
                    1A  6C  AB            02  E0 0022A          BBS     #2, 108(R11), 11$                                          ; 0802
                    50  04  AE 0000006C   8F  C1 0022F          ADDL3   #108, 4(SP), R0                                                    ;
                    0D  60                01  E0 00238          BBS     #1, (R0), 11$                                              ; 0804
                    51  04  AE 0000006C   8F  C1 0023C          ADDL3   #108, 4(SP), R1                                                    ;
                    08  61                02  E1 00245          BBC     #2, (R1), 12$                                              ; 0811
                        50 00000000G      8F  D0 00249 11$:     MOVL    #EXCH$_NOCOPBAD, R0                                        ; 0812
                                          04  00250              RET                                                                        ;
                        14  BE        40  8F  8A 00251 12$:     BICB2   #64, @20(SP)                                               ; 0820
                                          5A  DD 00256          PUSHL   TOT_LEN                                                    ; 0821
                    51  04  AE 00000065   8F  C1 00258          ADDL3   #101, 4(SP), R1                                                    ;
                    50      56            61  C1 00261          ADDL3   (R1), R6, R0                                                       ;
                                      5A  A0  9F 00265          PUSHAB  90(R0)                                                             ;
                                          57  DD 00268          PUSHL   R7                                                                 ;
                        00000000G     EF        03  FB 0026A          CALLS   #3, EXCH$RTACP_FIND_FILE                                           ;
                            03                50  E8 00271          BLBS    R0, 13$                                                    ;
                                          00DB  31 00274          BRW     24$                                                        ;
                        52  20        A8  D0 00277 13$:     MOVL    32(R8), R2                                                 ; 0836
                                          13  12 0027B          BNEQ    14$                                                        ;
                        7E  D1        8F  9A 0027D          MOVZBL  #209, -(SP)                                                        ;
                                          01  DD 00281          PUSHL   #1                                                                 ;
                                  00000000G  8F  DD 00283          PUSHL   #EXCH$_BADLOGIC                                                    ;
                        00000000G     00        03  FB 00289          CALLS   #3, LIB$STOP                                                       ;
                            F4  8F        0A  A2  91 00290 14$:     CMPB    10(R2), #244                                               ; 0840
                                          1E  12 00295          BNEQ    16$                                                        ;
                        7E  A7        7E  A2  D1 00297          CMPL    126(R2), 126(R7)                                           ; 0844
                                          17  12 0029C          BNEQ    16$                                                        ;
                0A      6C  AB            01  E0 0029E          BBS     #1, 108(R11), 15$                                          ; 0846
                05      6C  AB            02  E0 002A3          BBS     #2, 108(R11), 15$                                          ; 0848
                                      30  A9  95 002A8          TSTB    48(R9)                                                     ; 0850
                                          08  18 002AB          BGEQ    16$                                                        ;
                        50 00000000G      8F  D0 002AD 15$:     MOVL    #EXCH$_NOCOPSAMDEV, R0                                     ; 0852
                                          04  002B4              RET                                                                        ;
                                      43  A7  95 002B5 16$:     TSTB    67(R7)                                                     ; 0856
                                          08  13 002B8          BEQL    17$                                                        ;
                        50 00000000G      8F  D0 002BA          MOVL    #EXCH$_NOCOPDUP, R0                                        ; 0858
                                          04  002C1              RET                                                                        ;
                                      39  A7  95 002C2 17$:     TSTB    57(R7)                                                     ; 0860
                                          08  18 002C5          BGEQ    18$                                                        ;
                        50 00000000G      8F  D0 002C7          MOVL    #EXCH$_NOCOPPROT, R0                                       ; 0862
                                          04  002CE              RET                                                                        ;
                0B      30  A9            02  E0 002CF 18$:     BBS     #2, 48(R9), 19$                                            ; 0864
                        50 00000000G      8F  D0 002D4          MOVL    #EXCH$_NOCOPNODEL, R0                                      ; 0866
                                          04  002DB              RET                                                                        ;
                    0CAC  8F      3E  A7  B1 002DC 19$:     CMPW    62(R7), #3244                                              ; 0868
                                          08  12 002E2          BNEQ    20$                                                        ;
                        50 00000000G      8F  D0 002E4          MOVL    #EXCH$_NOCOPBADDEL, R0                                     ; 0870
                                          04  002EB              RET                                                                        ;
                    7ABB  8F      3E  A7  B1 002EC 20$:     CMPW    62(R7), #31419                                             ; 0872
                                          0D  12 002F2          BNEQ    21$                                                        ;
                0B      31  A9            01  E0 002F4          BBS     #1, 49(R9), 21$                                            ; 0874
```

```
                                50 00000000G   8F   D0 002F9          MOVL    #EXCH$_NOCOPSYSDEL, R0                          ; 0860
                                               04 00300          RET
                                      30       A9   95 00301 21$:    TSTB    48(R9)                                          ; 0864
                                               47   18 00304          BGEQ    23$
                                      39       A7   95 00306          TSTB    57(R7)                                          ; 0871
                                               13   18 00309          BGEQ    22$
                                7E             B3   8F   9A 0030B          MOVZBL  #179, -(SP)
                                               01   DD 0030F          PUSHL   #1
                                   00000000G   8F   DD 00311          PUSHL   #EXCH$_BADLOGIC
                             00000000G   00        03   FB 00317          CALLS   #3, LIB$STOP
                                50             7E   A7   D0 0031E 22$:    MOVL    126(R7), R0                             ; 0872
                                01   A0              02   90 00322          MOVB    #2, 1(R0)
                                               76   A7   DD 00326          PUSHL   118(R7)                                 ; 0873
                                               04   AE   DD 00329          PUSHL   4(SP)
                                0000V  CF              02   FB 0032C          CALLS   #2, EXCH$RT11_DIRSEG_PUT
                          1C         30   A9              03   E1 00331          BBC     #3, 48(R9), 24$                         ; 0874
                                               10   AE   DD 00336          PUSHL   16(SP)                                  ; 0876
                                               3A   A6   DD 00339          PUSHL   58(R6)
                                               02   DD 0033C          PUSHL   #2
                                   00000000G   8F   DD 0033E          PUSHL   #EXCH$_DELETEPREV
                             00000000G   00        04   FB 00344          CALLS   #4, LIB$SIGNAL
                                               05   11 0034B          BRB     24$                                     ; 0864
                                14             BE   40   8F   88 0034D 23$:    BISB2   #64, @20(SP)                            ; 0882
                          0066  8F         00    6E   00   2C 00352 24$:    MOVC5   #0, (SP), #0, #102, 28(R7)              ; 0889
                                1C             A7        00359
                                               2D   A8   D5 0035B          TSTL    45(R8)                                  ; 0894
                                               06   13   0035E          BEQL    25$
                                5A             2D   A8   D0 00360          MOVL    45(R8), BLOCKS                          ; 0896
                                               04   11   00364          BRB     26$
                                5A             24   A9   D0 00366 25$:    MOVL    36(R9), BLOCKS                          ; 0898
                          2D         A6              5A   D0 0036A 26$:    MOVL    BLOCKS, 45(R6)                          ; 0900
                    53    74         AE              01   C1 0036E          ADDL3   #1, RFP+80, R3                          ; 0904
                                               53   DD 00373          PUSHL   R3
                                               56   DD 00375          PUSHL   R6
                          00000000G  EF              02   FB 00377          CALLS   #2, EXCH$CMD_FETCH_RECFMT_IMPLIED
                                4A   A6   FA99        CF   9E 0037E          MOVAB   EXCH$RT11_CLOSE_FILE, 74(R6)            ; 0908
                                4E   A6   0000V       CF   9E 00384          MOVAB   EXCH$RT11_DELETE_FILE, 78(R6)           ; 0909
                                     52        A6   D4 0038A          CLRL    82(R6)                                  ; 0910
                                56   A6   00000000G  EF   9E 0038D          MOVAB   EXCH$PDP_PUT, 86(R6)                    ; 0911
                          05    2B   A8              01   E0 00395          BBS     #1, 43(R8), 27$                         ; 0915
                          0D    14   BE              01   E1 0039A          BBC     #1, @20(SP), 28$                        ; 0917
                                   00000000G   8F   DD 0039F 27$:    PUSHL   #EXCH$_NOCARRIAGE                       ; 0919
                             00000000G   00        01   FB 003A5          CALLS   #1, LIB$SIGNAL
                                               50   D4 003AC 28$:    CLRL    PHYSICAL                                ; 0921
                                01             29   A6   91 003AE          CMPB    41(R6), #1                              ; 0925
                                               06   13   003B2          BEQL    29$
                                01             29   A8   91 003B4          CMPB    41(R8), #1                              ; 0927
                                               0D   12   003B8          BNEQ    30$
                                50             01   D0 003BA 29$:    MOVL    #1, PHYSICAL                            ; 0930
                                28   A6              02   90 003BD          MOVB    #2, 40(R6)                              ; 0931
                                35   A6        0200  8F   3C 003C1          MOVZWL  #512, 53(R6)                            ; 0932
                                0E             2B   A8   E9 003C7 30$:    BLBC    43(R8), 31$                             ; 0937
                                0A             14   BE   E8 003CB          BLBS    @20(SP), 31$                            ; 0938
                                28   A6        28   A8   90 003CF          MOVB    40(R8), 40(R6)                          ; 0941
                                35   A6        35   A8   D0 003D4          MOVL    53(R8), 53(R6)                          ; 0942
                                               29   A8   95 003D9 31$:    TSTB    41(R8)                                  ; 0947
                                               2D   12   003DC          BNEQ    33$
```

```
                                29   A6  95  003DE          TSTB    41(R6)                                  ; 0948
                                28   12  003E1          BNEQ    33$
                          24    2B   A8  E8  003E3          BLBS    43(R8), 33$                             ; 0950
                          20    14   BE  E8  003E7          BLBS    @20(SP), 33$                            ; 0951
                          03    7A   AB  91  003EB          CMPB    122(R11), #3                            ; 0953
                                06   13  003EF          BEQL    32$
                          01    7A   AB  91  003F1          CMPB    122(R11), #1                            ; 0954
                                14   12  003F5          BNEQ    33$
                    28    A8          02  90  003F7  32$:   MOVB    #2, 40(R8)                              ; 0957
                    35    A8    0200  8F  3C  003FB          MOVZWL  #512, 53(R8)                            ; 0958
                    28    A6          02  90  00401          MOVB    #2, 40(R6)                              ; 0959
                    35    A6    0200  8F  3C  00405          MOVZWL  #512, 53(R6)                            ; 0960
                                5A   D5  0040B  33$:   TSTL    BLOCKS                                  ; 0966
                                1A   12  0040D          BNEQ    35$
                          13    50   E8  0040F          BLBS    PHYSICAL, 34$                           ; 0968
                          02    28   A8  91  00412          CMPB    40(R8), #2                              ; 0971
                                11   12  00416          BNEQ    35$
                          02    28   A6  91  00418          CMPB    40(R6), #2                              ; 0973
                                0B   12  0041C          BNEQ    35$
                    35    A6    35   A8  D1  0041E          CMPL    53(R8), 53(R6)                          ; 0975
                                04   12  00423          BNEQ    35$
                          5A    3E   A8  D0  00425  34$:   MOVL    62(R8), BLOCKS                          ; 0978
                                2C   A9  DD  00429  35$:   PUSHL   44(R9)                                  ; 0982
                                0480 8F  BB  0042C          PUSHR   #^M<R7,R10>
              00000000G         EF       03  FB  00430          CALLS   #3, EXCH$RTACP_FIND_EMPTY_AREA
                    20          AE       50  D0  00437          MOVL    R0, STATUS
                                05   20   AE  E8  0043B          BLBS    STATUS, 37$
                                50   20   AE  D0  0043F  36$:   MOVL    STATUS, R0                              ; 0984
                                     04   00443          RET
                          52    7E   A7  D0  00444  37$:   MOVL    126(R7), ENT                            ; 0988
                    01    A2          01  90  00448          MOVB    #1, 1(ENT)                              ; 0989
                    0E    30   A9          06  E1  0044C          BBC     #6, 48(R9), 38$                         ; 0994
          50        30   A9          01  05  EF  00451          EXTZV   #5, #1, 48(R9), R0                      ; 0996
01  A2              01             07  50  F0  00457          INSV    R0, #7, #1, 1(ENT)
                                     19  11  0045D          BRB     39$
                          50   20   A8  D0  0045F  38$:   MOVL    32(R8), R0                              ; 0999
                                     13  13  00463          BEQL    39$
                    F4    8F   0A   A0  91  00465          CMPB    10(R0), #244                            ; 1001
                                     0C  12  0046A          BNEQ    39$
          51        39   A0          01  07  EF  0046C          EXTZV   #7, #1, 57(R0), R1                      ; 1003
01  A2              01             07  51  F0  00472          INSV    R1, #7, #1, 1(ENT)
                          51   52   D0  00478  39$:   MOVL    ENT, R1                                 ; 1008
                    0000V 30   0047B          BSBW    EXCH$RT11_FORMAT_CURRENT_DATE
                    02    A2   9F  0047E          PUSHAB  2(ENT)                                  ; 1013
                                06   DD  00481          PUSHL   #6
                                78   AE  DD  00483          PUSHL   RFP+76
                                28   AE  DD  00486          PUSHL   NAM_LEN
                    00000000G   EF       04  FB  00489          CALLS   #4, EXCH$UTIL_RADIX50_FROM_ASCII
                                06   A2   9F  00490          PUSHAB  6(ENT)                                  ; 1015
                                03   DD  00493          PUSHL   #3
                                53   DD  00495          PUSHL   R3
                    50          24   AE       01  C3  00497          SUBL3   #1, TYP_LEN, R0                 ; 1014
                                50   DD  0049C          PUSHL   R0
                    00000000G   EF       04  FB  0049E          CALLS   #4, EXCH$UTIL_RADIX50_FROM_ASCII        ; 1015
                                76   A7   DD  004A5          PUSHL   118(R7)                                 ; 1019
                                04   AE   DD  004A8          PUSHL   4(SP)
                    0000V       CF       02  FB  004AB          CALLS   #2, EXCH$RT11_DIRSEG_PUT
```

```
                38   A7              62              0E  28 004B0          MOVC3   #14, (ENT), 56(R7)                    ; 1020
                                         24   A7     D4  CLRL 004B5          CLRL    36(R7)                            ; 1024
                                     52  72   A7     9E  MOVAB 004B8         MOVAB   114(R7), R2                       ; 1025
                                 1C  A7          62  D0  MOVL 004BC          MOVL    (R2), 28(R7)                      ;
                                     50  40   A7     3C  MOVZWL 004C0        MOVZWL  64(R7), R0                        ; 1026
                                     50          62  C0  ADDL2 004C4        ADDL2   (R2), R0                          ;
                                 20  A7      FF  A0  9E  MOVAB 004C7         MOVAB   -1(R0), 32(R7)                    ;
                                         42   A6     7C  CLRQ 004CC          CLRQ    66(R6)                            ; 1028
                                              57     DD  PUSHL 004CF         PUSHL   R7                                ; 1032
                            0000V CF              01  FB  CALLS 004D1       CALLS   #1, EXCH$RT11_EXPAND_FILENAME     ;
                                28  A7              03  D0  MOVL 004D6       MOVL    #3, 40(R7)                        ; 1037
                                         18   A7     D5  TSTL 004DA          TSTL    24(R7)                            ; 1041
                                              10     12  BNEQ 004DD          BNEQ    40$                               ;
                                     7E  1800     8F  3C  MOVZWL 004DF       MOVZWL  #6144, -(SP)                      ; 1043
                        00000000G    EF              01  FB  CALLS 004E4       CALLS   #1, EXCH$UTIL_VM_ALLOCATE         ;
                                18   A7              50  D0  MOVL 004EB       MOVL    R0, 24(R7)                        ;
                                     50  40   A7     3C  MOVZWL 004EF  40$:    MOVZWL  64(R7), R0                        ; 1047
                                     0C              50  B1  CMPW 004F3        CMPW    R0, #12                           ;
                                                     03  1B  BLEQU 004F6       BLEQU   41$                               ;
                                     50              0C  D0  MOVL 004F8        MOVL    #12, R0                           ;
                                     5A              50  D0  MOVL 004FB  41$:  MOVL    R0, BLOCKS                        ;
                                 2C  A7              62  D0  MOVL 004FE        MOVL    (R2), 44(R7)                      ; 1048
                            50       62              5A  C1  ADDL3 00502       ADDL3   BLOCKS, (R2), R0                  ; 1049
                                 30  A7          FF  A0  9E  MOVAB 00506       MOVAB   -1(R0), 48(R7)                    ;
                            34   A7              62              01  C3 0050B  SUBL3   #1, (R2), 52(R7)                 ; 1050
                                                 50              01  D0 00510  MOVL    #1, R0                            ; 1054
                                                                 04  00513      RET                                       ; 1055

; Routine Size:  1300 bytes,    Routine Base:  EXCH$RT11_CODE + 0404
```

```
;  967          1056  1 GLOBAL ROUTINE exch$rt11_delete_file (filb : $ref_bblock) =     %SBTTL 'exch$rt11_delete_file (filb)'
;  968          1057  2 BEGIN
;  969          1058  2 !++
;  970          1059  2 !
;  971          1060  2 ! FUNCTIONAL DESCRIPTION:
;  972          1061  2 !
;  973          1062  2 !       Perform RT-11 volume specific delete processing.  This is only used to delete output files which we
;  974          1063  2 !       have created, but decide to delete.  For example, if the input file were totally unreadable we might
;  975          1064  2 !       delete the output file.
;  976          1065  2 !
;  977          1066  2 ! INPUT/OUTPUT:
;  978          1067  2 !
;  979          1068  2 !       filb - pointer to block describing the file
;  980          1069  2 !
;  981          1070  2 ! IMPLICIT INPUTS:
;  982          1071  2 !
;  983          1072  2 !       none
;  984          1073  2 !
;  985          1074  2 ! OUTPUTS:
;  986          1075  2 !
;  987          1076  2 !       filb - receive info pertaining to the file to be deleted
;  988          1077  2 !
;  989          1078  2 ! IMPLICIT OUTPUTS:
;  990          1079  2 !
;  991          1080  2 !       none
;  992          1081  2 !
;  993          1082  2 ! ROUTINE VALUE:
;  994          1083  2 !
;  995          1084  2 !       true if able to delete the file, false otherwise
;  996          1085  2 !
;  997          1086  2 ! SIDE EFFECTS:
;  998          1087  2 !
;  999          1088  2 !       none
; 1000          1089  2 !--
; 1001          1090  2
; 1002          1091  2 $dbgtrc_prefix ('exch$rt11_delete_file> ');
; 1003          1092  2
; 1004          1093  2 LOCAL
; 1005          1094  2     status
; 1006          1095  2     ;
; 1007          1096  2
; 1008          1097  2 BIND
; 1009          1098  2     ctx  = filb [filb$a_context]         : $ref_bblock,
; 1010          1099  2     namb = filb [filb$a_assoc_namb]      : $ref_bblock,
; 1011          1100  2     volb = filb [filb$a_assoc_volb]      : $ref_bblock
; 1012          1101  2     ;
; 1013          1102  2
; 1014          1103  2 $debug_print_lit ('entry');
; 1015          1104  2
; 1016          1105  2 $block_check (2, .filb, filb, 560);
; 1017          1106  2 $block_check (2, .ctx, rt11ctx, 561);
; 1018          1107  2 $logic_check (3, (.ctx [rt11ctx$v_output_file]), 149);
; 1019          1108  2
; 1020          1109  2 ! Not much to do, simply leave the file marked as tentative
; 1021          1110  2 !
; 1022          1111  2 ctx [rt11ctx$l_flags] = 0;
; 1023          1112  2
```

```
; 1024          1113  2 RETURN true;
; 1025          1114  1 END;


                                                001C 00000      .ENTRY  EXCH$RT11_DELETE_FILE, Save R2,R3,R4    ; 1056
                                    54 00000000G   EF   9E 00002    MOVAB   EXCH$UTIL_BLOCK_CHECK, R4               ;
                          53        04     AC      20   C1 00009    ADDL3   #32, FILB, R3                           ; 1098
                                    52 035B00FA    8F   D0 0000E    MOVL    #56295674, R2                           ; 1105
                                    51     0230    8F   3C 00015    MOVZWL  #560, R1                                ;
                                    50       04    AC   D0 0001A    MOVL    FILB, R0                                ;
                                                   64   16 0001E    JSB     EXCH$UTIL_BLOCK_CHECK                   ;
                                    53             63   D0 00020    MOVL    (R3), R3                                ; 1106
                                    52 008200F4    8F   D0 00023    MOVL    #8519924, R2                            ;
                                    51     0231    8F   3C 0002A    MOVZWL  #561, R1                                ;
                                    50             53   D0 0002F    MOVL    R3, R0                                  ;
                                                   64   16 00032    JSB     EXCH$UTIL_BLOCK_CHECK                   ;
                                              28   A3   D4 00034    CLRL    40(R3)                                  ; 1111
                                    50             01   D0 00037    MOVL    #1, R0                                  ; 1113
                                                   04   00039      RET                                             ; 1114

; Routine Size:  59 bytes,    Routine Base:  EXCH$RT11_CODE + 0918
```

```
;  1027     1115  1 GLOBAL ROUTINE exch$rt11_dircache_exit_handler (status, %SBTTL 'exch$rt11_dircache_exit_handler (status, vol
;  1028     1116  1                                                 volb : $ref_bblock) : NOVALUE =
;  1029     1117  2 BEGIN
;  1030     1118  2 !++
;  1031     1119  2 !
;  1032     1120  2 ! FUNCTIONAL DESCRIPTION:
;  1033     1121  2 !
;  1034     1122  2 !       Flush the write cache on the directory.
;  1035     1123  2 !
;  1036     1124  2 ! INPUTS:
;  1037     1125  2 !
;  1038     1126  2 !       status - pointer to status code
;  1039     1127  2 !       volb - pointer to volb which has been connected to the RT-11 device
;  1040     1128  2 !
;  1041     1129  2 ! IMPLICIT INPUTS:
;  1042     1130  2 !
;  1043     1131  2 !       none
;  1044     1132  2 !
;  1045     1133  2 ! OUTPUTS:
;  1046     1134  2 !
;  1047     1135  2 !       none
;  1048     1136  2 !
;  1049     1137  2 ! IMPLICIT OUTPUTS:
;  1050     1138  2 !
;  1051     1139  2 !       none
;  1052     1140  2 !
;  1053     1141  2 ! ROUTINE VALUE:
;  1054     1142  2 !
;  1055     1143  2 !       none
;  1056     1144  2 !
;  1057     1145  2 ! SIDE EFFECTS:
;  1058     1146  2 !
;  1059     1147  2 !       any modified directory segments will be written
;  1060     1148  2 !--
;  1061     1149  2
;  1062     1150  2 $dbgtrc_prefix ('rt11_dircache_exit_handler> ');
;  1063     1151  2
;  1064     1152  2 BIND
;  1065     1153  2     fab = volb [volb$a_fab] : $ref_bblock,
;  1066     1154  2     rab = volb [volb$a_rab] : $ref_bblock
;  1067     1155  2     ;
;  1068     1156  2
;  1069     1157  2 $trace_print_fao ('entry - volb !XL, dircache !XL', .volb, .volb [volb$l_dircache]);
;  1070     1158  2
;  1071     1159  2 ! If there are any modified segments signal and flush
;  1072     1160  2 !
;  1073     1161  2 IF .volb [volb$l_dircache] NEQ volb$m_dircache_active
;  1074     1162  2 THEN
;  1075     1163  3     BEGIN
;  1076     1164  3
;  1077     1165  3     ! Tell we are flushing the directory of a slow device, it might be a while before it finishes
;  1078     1166  3     !
;  1079     1167  3     IF .volb [volb$l_devtype] EQL dt$_tu58      ! If it is any kind of TU58
;  1080     1168  3     THEN
;  1081     1169  4         BEGIN
;  1082     1170  4         LOCAL
;  1083     1171  4             msgvec : VECTOR [5, LONG],
```

```
;  1084          1172  4              status;
;  1085          1173  4
;  1086          1174  4          ! We use the $putmsg service to print this message.  If we signalled it, we could exit the image if
;  1087          1175  4          ! another signal was active in the catch-all condition handler.  This is extremely likely to happen
;  1088          1176  4          ! if the control/Y was hit during a command with a /LOG in effect, since the catch-all handler ends
;  1089          1177  4          ! up printing EXCHANGE log messages.
;  1090          1178  4          !
;  1091          1179  4          msgvec [0] = 4;
;  1092          1180  4          msgvec [1] = exch$_writecache;
;  1093          1181  4          msgvec [2] = 2;
;  1094          1182  4          msgvec [3] = .volb [volb$l_vol_ident_len];
;  1095          1183  4          msgvec [4] = volb [volb$t_vol_ident];
;  1096          1184  5          IF NOT (status = $putmsg (msgvec=msgvec))
;  1097          1185  4          THEN
;  1098          1186  4              $exch_signal_stop (.status);
;  1099          1187  3          END;
;  1100          1188  3
;  1101          1189  3      ! It is possible that I/O is active (likely if the device is a TU58), so wait for it to complete
;  1102          1190  3      !
;  1103          1191  4      IF NOT (status = $wait (rab = .rab))
;  1104          1192  3      THEN
;  1105          1193  3          exch$util_file_error (exch$_waiterr, .status, .fab, .rab [rab$l_stv]);
;  1106          1194  3
;  1107          1195  3      ! Call the normal cache stop routine
;  1108          1196  3      !
;  1109          1197  3      exch$rt11_dircache_stop (.volb);
;  1110          1198  3
;  1111          1199  2      END;
;  1112          1200  2
;  1113          1201  2 RETURN;
;  1114          1202  1 END;


                                                                .EXTRN  EXCH$_WRITECACHE
                                                                .EXTRN  SYS$PUTMSG, LIB$STOP
                                                                .EXTRN  SYS$WAIT, EXCH$_WAITERR

                                                                .ENTRY  EXCH$RT11_DIRCACHE_EXIT_HANDLER, Save R2,R3 ; 1115
                                                        5E      SUBL2   #20, SP                                     ;
                                                        53      MOVL    VOLB, R3                                    ; 1153
                                                        01      CMPL    80(R3), #1                                  ; 1161
                                                                BEQL    3$                                          ;
                                                        0E      CMPL    60(R3), #14                                 ; 1167
                                                                BNEQ    1$                                          ;
                                                        6E      MOVL    #4, MSGVEC                                  ; 1179
                                                04      AE      MOVL    #EXCH$_WRITECACHE, MSGVEC+4                 ; 1180
                                                08      AE      MOVL    #2, MSGVEC+8                                ; 1181
                                                0C      AE      MOVL    101(R3), MSGVEC+12                          ; 1182
                                                10      AE      MOVAB   105(R3), MSGVEC+16                          ; 1183
                                                        CLRQ    -(SP)                                               ; 1184
                                                        CLRL    -(SP)                                               ;
                                                        PUSHAB  MSGVEC                                              ;
                                                00000000G       CALLS   #4, SYS$PUTMSG                              ;
                                                        BLBS    STATUS, 1$                                          ;
                                                        PUSHL   STATUS                                              ; 1186
                                                00000000G       CALLS   #1, LIB$STOP                                ;
```

```
                                                04 00048          RET                                                     ;
                                52       14    A3 DO 00049 1$:   MOVL    20(R3), R2                                    ; 1191
                                                52 DD 0004D       PUSHL   R2                                            ;
                      00000000G 00              01 FB 0004F       CALLS   #1, SYS$WAIT                                  ;
                             04 AC              50 DO 00056       MOVL    R0, STATUS                                    ;
                                16              50 E8 0005A       BLBS    R0, 2$                                        ;
                                         0C     A2 DD 0005D       PUSHL   12(R2)                                        ; 1193
                                         10     A3 DD 00060       PUSHL   16(R3)                                        ;
                                         04     AC DD 00063       PUSHL   STATUS                                        ;
                                  00000000G     8F DD 00066       PUSHL   #EXCH$_WAITERR                                ;
                      00000000G EF              04 FB 0006C       CALLS   #4, EXCH$UTIL_FILE_ERROR                      ;
                                                53 DD 00073 2$:   PUSHL   R3                                            ; 1197
                           0000V CF             01 FB 00075       CALLS   #1, EXCH$RT11_DIRCACHE_STOP                   ;
                                                04 0007A 3$:      RET                                                   ; 1202

; Routine Size:  123 bytes,    Routine Base:  EXCH$RT11_CODE + 0953
```

```
;  1116          1203  1 GLOBAL ROUTINE exch$rt11_dircache_start (volb : $ref_bblock) : NOVALUE =        %SBTTL 'exch$rt11_dircache_s
;  1117          1204  2 BEGIN
;  1118          1205  2 !++
;  1119          1206  2 !
;  1120          1207  2 ! FUNCTIONAL DESCRIPTION:
;  1121          1208  2 !
;  1122          1209  2 !       Set up the write cache on the directory.
;  1123          1210  2 !
;  1124          1211  2 ! INPUTS:
;  1125          1212  2 !
;  1126          1213  2 !       volb - pointer to volb which has been connected to the RT-11 device
;  1127          1214  2 !
;  1128          1215  2 ! IMPLICIT INPUTS:
;  1129          1216  2 !
;  1130          1217  2 !       none
;  1131          1218  2 !
;  1132          1219  2 ! OUTPUTS:
;  1133          1220  2 !
;  1134          1221  2 !       none
;  1135          1222  2 !
;  1136          1223  2 ! IMPLICIT OUTPUTS:
;  1137          1224  2 !
;  1138          1225  2 !       none
;  1139          1226  2 !
;  1140          1227  2 ! ROUTINE VALUE:
;  1141          1228  2 !
;  1142          1229  2 !       none
;  1143          1230  2 !
;  1144          1231  2 ! SIDE EFFECTS:
;  1145          1232  2 !
;  1146          1233  2 !       error conditions will be signaled
;  1147          1234  2 !--
;  1148          1235  2
;  1149          1236  2 $dbgtrc_prefix ('rt11_dircache_start> ');
;  1150          1237  2
;  1151          1238  2 LOCAL
;  1152          1239  2     status
;  1153          1240  2     ;
;  1154          1241  2
;  1155          1242  2 $block_check (2, .volb, volb, 461);
;  1156          1243  2 $logic_check (2, (.volb [volb$v_write]), 203);  ! We shouldn't get this far if we aren't supposed to write t
;  1157          1244  2
;  1158          1245  2 ! If global caching is in effect, ignore this call
;  1159          1246  2 !
;  1160          1247  2 IF .exch$a_gbl [excg$v_q_cache]
;  1161          1248  2 THEN
;  1162          1249  2     RETURN;
```

```
;  1164          1250  2 ! Check some conditions before we proceed
;  1165          1251  2 !
;  1166          1252  2 $trace_print_fao ('entry - volb !XL', .volb);
;  1167          1253  2 $logic_check (4, (exch$rtacp_verify_directory (.volb)), 204);
;  1168          1254  2 $logic_check (2, (NOT .volb [volb$v_dircache_active]), 131);    ! If it is already on we are confused
;  1169        L 1255  2 $logic_check (4, (volb$m_dircache_active EQL 1), 132);
;  %PRINT:       assumption 132 verified during compilation
;  1170          1256  2
;  1171          1257  2 ! Engage directory write caching.  Clear all 31 segment flags and activate caching by putting a 1 in the
;  1172          1258  2 ! caching longword
;  1173          1259  2 !
;  1174          1260  2 volb [volb$l_dircache] = volb$m_dircache_active;
;  1175          1261  2
;  1176          1262  2 ! Declare an exit handler, so that we can flush the cache if the image is run down
;  1177          1263  2 !
;  1178          1264  2 $logic_check (1, (.exch$a_gbl [excg$a_exh_routine] EQL 0), 313);                ! There had better not be on
;  1179          1265  2 exch$a_gbl [excg$a_exh_routine]          = exch$rt11_dircache_exit_handler;     ! Routine to flush the cache
;  1180          1266  2 exch$a_gbl [excg$l_exh_arg_count]       = 2;                                   ! Status and volb
;  1181          1267  2 exch$a_gbl [excg$a_exh_status]          = exch$a_gbl [excg$l_exh_condvalu];    ! Address to store status
;  1182          1268  2 exch$a_gbl [excg$a_exh_volb]            = .volb;                               ! Pass address of volb
;  1183          1269  2
;  1184          1270  3 IF NOT (status = $dclexh (desblk=exch$a_gbl [excg$r_exit_block]))
;  1185          1271  2 THEN
;  1186          1272  2     $exch_signal_stop (.status);
;  1187          1273  2
;  1188          1274  2 RETURN;
;  1189          1275  1 END;


                                                                .EXTRN  SYS$DCLEXH

                                                007C 00000      .ENTRY  EXCH$RT11_DIRCACHE_START, Save R2,R3,R4,R5,-    ; 1203
                                                                        R6
                                    56 00000000G   EF   9E 00002        MOVAB   EXCH$A_GBL, R6                          ;
                                    55 00000000G   8F   D0 00009        MOVL    #EXCH$_BADLOGIC, R5                     ;
                                    54 00000000G   00   9E 00010        MOVAB   LIB$STOP, R4                            ;
                                    53       04    AC   D0 00017        MOVL    VOLB, R3                                ; 1242
                                    52 041B00F3    8F   D0 0001B        MOVL    #68878579, R2                           ;
                                    51     01CD    8F   3C 00022        MOVZWL  #461, R1                                ;
                                    50             53   D0 00027        MOVL    R3, R0                                  ;
                                           00000000G   EF   16 0002A    JSB     EXCH$UTIL_BLOCK_CHECK                   ;
                              0B    48       A3         05   E0 00030    BBS     #5, 72(R3), 1$                          ; 1243
                                    7E       CB   8F   9A 00035        MOVZBL  #203, -(SP)                             ;
                                                  01   DD 00039        PUSHL   #1                                      ;
                                                  55   DD 0003B        PUSHL   R5                                      ;
                                    64                 03   FB 0003D    CALLS   #3, LIB$STOP                            ;
                              4F    00       B6         01   E0 00040 1$: BBS     #1, @EXCH$A_GBL, 4$                     ; 1247
                                    0B       50   A3    E9 00045        BLBC    80(R3), 2$                              ; 1254
                                    7E       83   8F    9A 00049        MOVZBL  #131, -(SP)                             ;
                                                  01    DD 0004D        PUSHL   #1                                      ;
                                                  55    DD 0004F        PUSHL   R5                                      ;
                                    64                  03    FB 00051  CALLS   #3, LIB$STOP                            ;
                                    50   A3             01    D0 00054 2$: MOVL    #1, 80(R3)                         ; 1260
                                    50                  66    D0 00058  MOVL    EXCH$A_GBL, R0                        ; 1264
                                               30   A0   D5 0005B        TSTL    48(R0)                                  ;
                                                    0C   13 0005E        BEQL    3$                                      ;
```

```
                            7E      0139    8F  3C 00060          MOVZWL  #313, -(SP)                                         ;
                                            01  DD 00065          PUSHL   #1                                                  ;
                                            55  DD 00067          PUSHL   R5                                                  ;
                            64              03  FB 00069          CALLS   #3, LIB$STOP                                        ;
                            50              66  D0 0006C 3$:      MOVL    EXCH$A_GBL, R0                                      ; 1265
                      30    A0      FF12    CF  9E 0006F          MOVAB   EXCH$RT11_DIRCACHE_EXIT_HANDLER, 48(R0)             ;
                      34    A0              02  D0 00075          MOVL    #2, 52(R0)                                          ; 1266
                      38    A0        40    A0  9E 00079          MOVAB   64(R0), 56(R0)                                      ; 1267
                      3C    A0              53  D0 0007E          MOVL    R3, 60(R0)                                          ; 1268
                                      2C    A0  9F 00082          PUSHAB  44(R0)                                              ; 1270
                00000000G   00              01  FB 00085          CALLS   #1, SYS$DCLEXH                                      ;
                            05              50  E8 0008C          BLBS    STATUS, 4$                                          ;
                                            50  DD 0008F          PUSHL   STATUS                                              ; 1272
                            64              01  FB 00091          CALLS   #1, LIB$STOP                                        ;
                                                04  00094 4$:      RET                                                         ; 1275

; Routine Size:  149 bytes,    Routine Base:  EXCH$RT11_CODE + 09CE
```

```
;  1191          1276  1 GLOBAL ROUTINE exch$rt11_dircache_stop (volb : $ref_bblock) : NOVALUE = %SBTTL 'exch$rt11_dircache_stop (vol
;  1192          1277  2 BEGIN
;  1193          1278  2 !++
;  1194          1279  2 !
;  1195          1280  2 ! FUNCTIONAL DESCRIPTION:
;  1196          1281  2 !
;  1197          1282  2 !       Clear and flush caches.
;  1198          1283  2 !
;  1199          1284  2 ! INPUTS:
;  1200          1285  2 !
;  1201          1286  2 !       volb - pointer to volb which has been connected to the RT-11 device
;  1202          1287  2 !
;  1203          1288  2 ! IMPLICIT INPUTS:
;  1204          1289  2 !
;  1205          1290  2 !       none
;  1206          1291  2 !
;  1207          1292  2 ! OUTPUTS:
;  1208          1293  2 !
;  1209          1294  2 !       none
;  1210          1295  2 !
;  1211          1296  2 ! IMPLICIT OUTPUTS:
;  1212          1297  2 !
;  1213          1298  2 !       none
;  1214          1299  2 !
;  1215          1300  2 ! ROUTINE VALUE:
;  1216          1301  2 !
;  1217          1302  2 !       none
;  1218          1303  2 !
;  1219          1304  2 ! SIDE EFFECTS:
;  1220          1305  2 !
;  1221          1306  2 !       error conditions will be signaled
;  1222          1307  2 !--
;  1223          1308  2
;  1224          1309  2 $dbgtrc_prefix ('rt11_dircache_stop> ');
;  1225          1310  2
;  1226          1311  2 $block_check (2, .volb, volb, 457);
```

```
;  1228          1312  2 ! If global caching is in effect, ignore this call
;  1229          1313  2 !
;  1230          1314  2 IF .exch$a_gbl [excg$v_q_cache]
;  1231          1315  2 THEN
;  1232          1316  2     RETURN;
;  1233          1317  2
;  1234          1318  2 $trace_print_fao ('entry - volb !XL, dircache !XL', .volb, .volb [volb$l_dircache]);
;  1235          1319  2 $logic_check (2, (.volb [volb$v_write]), 175);  ! We shouldn't get this far if we aren't supposed to write t
;  1236          1320  2
;  1237          1321  2 ! Verify that the directory is valid before we allow it to be written.  Since a corrupted directory would ha
;  1238          1322  2 ! been write-locked when we mounted, this means that EXCHANGE has corrupted the directory
;  1239          1323  2 !
;  1240          1324  2 $logic_check (0, (exch$rtacp_verify_directory (.volb)), 178);
;  1241          1325  2
;  1242          1326  2 ! Clear the cache bit so that we will really write, then flush the directory
;  1243          1327  2 !
;  1244          1328  2 volb [volb$v_dircache_active] = false;
;  1245          1329  2 exch$rt11_dirseg_flush (.volb, .volb [volb$l_dircache]);
;  1246          1330  2
;  1247          1331  2 ! Cancel the exit handler which was declared to flush this cache
;  1248          1332  2 !
;  1249          1333  2 $canexh (desblk=exch$a_gbl [excg$r_exit_block]);
;  1250          1334  2 exch$a_gbl [excg$a_exh_routine] = 0;             ! Mark that no exit handler is active
;  1251          1335  2
;  1252          1336  2 RETURN;
;  1253          1337  1 END;


                                                                .EXTRN  SYS$CANEXH

                                        007C 00000              .ENTRY  EXCH$RT11_DIRCACHE_STOP, Save R2,R3,R4,R5,-     ; 1276
                                                                R6                                                      ;
                    56 00000000G    00  9E 00002              MOVAB   LIB$STOP, R6                                    ;
                    55 00000000G    8F  D0 00009              MOVL    #EXCH$_BADLOGIC, R5                             ;
                    54 00000000G    EF  9E 00010              MOVAB   EXCH$A_GBL, R4                                  ;
                    53       04     AC  D0 00017              MOVL    VOLB, R3                                        ; 1311
                    52 041B00F3     8F  D0 0001B              MOVL    #68878579, R2                                   ;
                    51     01C9     8F  3C 00022              MOVZWL  #457, R1                                        ;
                    50              53  D0 00027              MOVL    R3, R0                                          ;
                       00000000G    EF  16 0002A              JSB     EXCH$UTIL_BLOCK_CHECK                           ;
                46     00    B4        01  E0 00030              BBS     #1, @EXCH$A_GBL, 3$                             ; 1314
                0B     48    A3        05  E0 00035              BBS     #5, 72(R3), 1$                                  ; 1319
                          7E     AF    8F  9A 0003A              MOVZBL  #175, -(SP)                                     ;
                                       01  DD 0003E              PUSHL   #1                                              ;
                                       55  DD 00040              PUSHL   R5                                              ;
                          66           03  FB 00042              CALLS   #3, LIB$STOP                                    ;
                                       53  DD 00045  1$:         PUSHL   R3                                              ; 1324
                00000000G EF           01  FB 00047              CALLS   #1, EXCH$RTACP_VERIFY_DIRECTORY                 ;
                          0B           50  E8 0004E              BLBS    R0, 2$                                          ;
                          7E     B2    8F  9A 00051              MOVZBL  #178, -(SP)                                     ;
                                       01  DD 00055              PUSHL   #1                                              ;
                                       55  DD 00057              PUSHL   R5                                              ;
                          66           03  FB 00059              CALLS   #3, LIB$STOP                                    ;
                50        A3           01  8A 0005C  2$:         BICB2   #1, 80(R3)                                      ; 1328
                                 50    A3  DD 00060              PUSHL   80(R3)                                          ; 1329
                                       53  DD 00063              PUSHL   R3                                              ;
```

```
                 0000V  CF            02  FB 00065         CALLS   #2, EXCH$RT11_DIRSEG_FLUSH                ;
              7E        64            2C  C1 0006A         ADDL3   #44, EXCH$A_GBL, -(SP)                   ; 1333
          00000000G     00            01  FB 0006E         CALLS   #1, SYS$CANEXH                           ;
                        50            64  D0 00075         MOVL    EXCH$A_GBL, R0                           ; 1334
                                   30 A0  D4 00078         CLRL    48(R0)                                   ;
                                      04  0007B 3$:        RET                                              ; 1337

; Routine Size:  124 bytes,    Routine Base:  EXCH$RT11_CODE + 0A63
```

```
;  1255     1338  1 GLOBAL ROUTINE exch$rt11_dirseg_flush (volb : $ref_bblock,      %SBTTL 'exch$rt11_dirseg_flush (volb, mod)'
;  1256     1339  1                                                 segment_modified : BITVECTOR [32]) =
;  1257     1340  2 BEGIN
;  1258     1341  2 !++
;  1259     1342  2 !
;  1260     1343  2 ! FUNCTIONAL DESCRIPTION:
;  1261     1344  2 !
;  1262     1345  2 !       Write any directory segments which have been modified.  Whether any actual I/O will occur depends on
;  1263     1346  2 !       volb [volb$v_dircache_active] bit.  If this bit is set, actual I/O will be postponed until flush is
;  1264     1347  2 !       with the bit clear.
;  1265     1348  2 !
;  1266     1349  2 ! INPUTS:
;  1267     1350  2 !
;  1268     1351  2 !       volb    - pointer to volb which has been connected to the RT-11 device
;  1269     1352  2 !       modified_segments - bitvector, set means to write the segment, clear means don't write it
;  1270     1353  2 !
;  1271     1354  2 ! IMPLICIT INPUTS:
;  1272     1355  2 !
;  1273     1356  2 !       none
;  1274     1357  2 !
;  1275     1358  2 ! OUTPUTS:
;  1276     1359  2 !
;  1277     1360  2 !       none
;  1278     1361  2 !
;  1279     1362  2 ! IMPLICIT OUTPUTS:
;  1280     1363  2 !
;  1281     1364  2 !       none
;  1282     1365  2 !
;  1283     1366  2 ! ROUTINE VALUE:
;  1284     1367  2 !
;  1285     1368  2 !       true if success, false if failed
;  1286     1369  2 !
;  1287     1370  2 ! SIDE EFFECTS:
;  1288     1371  2 !
;  1289     1372  2 !       error conditions will be signaled
;  1290     1373  2 !--
;  1291     1374  2
;  1292     1375  2 $dbgtrc_prefix ('rt11_dirseg_flush> ');
;  1293     1376  2
;  1294     1377  2 LOCAL
;  1295     1378  2     seg : $ref_bblock,
;  1296     1379  2     status
;  1297     1380  2     ;
;  1298     1381  2
;  1299     1382  2 BIND
;  1300     1383  2     modified_segments = segment_modified        ! map a longword onto the bitvector
;  1301     1384  2     ;
;  1302     1385  2
;  1303     1386  2 $trace_print_fao ('entry - volb !XL, dircache !XL', .volb, .modified_segments);
;  1304     1387  2
;  1305     1388  2 $block_check (2, .volb, volb, 532);
;  1306     1389  2
;  1307     1390  2 ! Assume that all will go well
;  1308     1391  2 !
;  1309     1392  2 status = true;
;  1310     1393  2
;  1311     1394  2 ! A quick exit in case nothing has changed
```

```
;  1312          1395  2 !
;  1313          1396  2 IF .modified_segments EQL 0                    ! No directory segments have been modified, nothing to do
;  1314          1397  2 THEN
;  1315          1398  2     RETURN .status;
;  1316          1399  2
;  1317          1400  2 ! Find the high segment
;  1318          1401  2 !
;  1319          1402  2 seg = exch$rt11_dirseg_get (.volb, 1);
;  1320          1403  2 $logic_check (2, (.seg NEQ 0), 210);
;  1321          1404  2 $trace_print_fao ('high segment !UL', .seg [rt11hdr$w_high_seg]);
;  1322          1405  2
;  1323          1406  2 ! Look at each of the bits, writing those that are set
;  1324          1407  2 !
;  1325          1408  2 INCRU seg_num FROM 1 TO .seg [rt11hdr$w_high_seg]
;  1326          1409  2 DO
;  1327          1410  3     BEGIN
;  1328          1411  3     $trace_print_fao ('seg num !UL, modified !UL', .seg_num, .segment_modified [.seg_num]);
;  1329          1412  3     IF .segment_modified [.seg_num]
;  1330          1413  3     THEN
;  1331          1414  4         BEGIN
;  1332          1415  4         LOCAL                                   ! We will continue if error, but we want to remember the wor
;  1333          1416  4             temp;
;  1334          1417  4
;  1335          1418  4         temp = exch$rt11_dirseg_put (.volb, .seg_num);
;  1336          1419  4         IF NOT .temp
;  1337          1420  4         THEN
;  1338          1421  4             status = .temp;
;  1339          1422  3         END;
;  1340          1423  2     END;
;  1341          1424  2
;  1342          1425  2 RETURN .status;
;  1343          1426  1 END;


                                    001C 00000         .ENTRY  EXCH$RT11_DIRSEG_FLUSH, Save R2,R3,R4       ;  1338
                            52 041B00F3 8F   D0 00002  MOVL    #68878579, R2                                ;  1388
                            51      0214 8F   3C 00009  MOVZWL  #532, R1                                     ;
                            50        04 AC   D0 0000E  MOVL    VOLB, R0                                   ;
                         00000000G    EF   16 00012  JSB     EXCH$UTIL_BLOCK_CHECK                        ;
                            54        01   D0 00018  MOVL    #1, STATUS                                   ;  1392
                                    08 AC   D5 0001B  TSTL    MODIFIED_SEGMENTS                            ;  1396
                                       47   13 0001E  BEQL    5$                                           ;
                                       01   DD 00020  PUSHL   #1                                           ;  1402
                                    04 AC   DD 00022  PUSHL   VOLB                                         ;
                                0000V CF    02   FB 00025  CALLS   #2, EXCH$RT11_DIRSEG_GET                ;
                                      52    50   D0 0002A  MOVL    R0, SEG                                  ;
                                            13   12 0002D  BNEQ    1$                                       ;  1403
                                    7E      D2 8F   9A 0002F  MOVZBL  #210, -(SP)                          ;
                                            01   DD 00033  PUSHL   #1                                      ;
                                 00000000G  8F   DD 00035  PUSHL   #EXCH$_BADLOGIC                         ;
                            00000000G 00    03   FB 0003B  CALLS   #3, LIB$STOP                            ;
                                      53    A2   3C 00042 1$:  MOVZWL  4(SEG), R3                          ;  1408
                                      52    01   D0 00046  MOVL    #1, SEG_NUM                              ;
                                            17   11 00049  BRB     4$                                       ;
```

```
                10      08   AC              52  E1 0004B 2$:    BBC     SEG_NUM, SEGMENT_MODIFIED, 3$           ; 1412
                                             52  DD 00050        PUSHL   SEG_NUM                                 ; 1418
                                         04  AC  DD 00052        PUSHL   VOLB                                    ;
                        0000V CF             02  FB 00055        CALLS   #2, EXCH$RT11_DIRSEG_PUT                ;
                              03             50  E8 0005A        BLBS    TEMP, 3$                                ; 1419
                              54             50  D0 0005D        MOVL    TEMP, STATUS                            ; 1421
                                             52  D6 00060 3$:    INCL    SEG_NUM                                 ; 1408
                              53             52  D1 00062 4$:    CMPL    SEG_NUM, R3                             ;
                                             E4  1B 00065        BLEQU   2$                                      ;
                              50             54  D0 00067 5$:    MOVL    STATUS, R0                              ; 1425
                                                 04 0006A        RET                                             ; 1426

; Routine Size:  107 bytes,    Routine Base:  EXCH$RT11_CODE + 0ADF
```

```
;  1345          1427  1 GLOBAL ROUTINE exch$rt11_dirseg_get (volb : $ref_bblock, number) =      %SBTTL 'exch$rt11_dirseg_get (volb)'
;  1346          1428  2 BEGIN
;  1347          1429  2 !++
;  1348          1430  2 !
;  1349          1431  2 ! FUNCTIONAL DESCRIPTION:
;  1350          1432  2 !
;  1351          1433  2 !       Return a pointer to the requested directory segment
;  1352          1434  2 !
;  1353          1435  2 ! INPUTS:
;  1354          1436  2 !
;  1355          1437  2 !       volb - pointer to volb which has been connected to the RT-11 device
;  1356          1438  2 !       number - directory segment number in the range 1-31
;  1357          1439  2 !
;  1358          1440  2 ! IMPLICIT INPUTS:
;  1359          1441  2 !
;  1360          1442  2 !       none
;  1361          1443  2 !
;  1362          1444  2 ! OUTPUTS:
;  1363          1445  2 !
;  1364          1446  2 !       none
;  1365          1447  2 !
;  1366          1448  2 ! IMPLICIT OUTPUTS:
;  1367          1449  2 !
;  1368          1450  2 !       none
;  1369          1451  2 !
;  1370          1452  2 ! ROUTINE VALUE:
;  1371          1453  2 !
;  1372          1454  2 !       address of segment, or 0 if any error
;  1373          1455  2 !
;  1374          1456  2 ! SIDE EFFECTS:
;  1375          1457  2 !
;  1376          1458  2 !       error conditions will be signaled
;  1377          1459  2 !--
;  1378          1460  2
;  1379          1461  2 $dbgtrc_prefix ('rt11_dirseg_get> ');
;  1380          1462  2
;  1381          1463  2 LOCAL
;  1382          1464  2     rtv : $ref_bblock,                          ! a pointer to the rt11 volb extension
;  1383          1465  2     rot : $ref_bblock,                          ! a pointer to the root directory segment
;  1384          1466  2     seg : $ref_bblock                           ! a pointer to the desired segment
;  1385          1467  2     ;
;  1386          1468  2
;  1387          1469  2 $block_check (2, .volb, volb, 453);
;  1388          1470  2 $trace_print_fao ('  entry - volb !XL, seg !2UL, dircache !XL', .volb, .number, .volb [volb$l_dircache]);
;  1389          1471  2
;  1390          1472  2 ! Get the pointer to our volb extension and to the root segment
;  1391          1473  2 !
;  1392          1474  2 rtv = .volb [volb$a_vfmt_specific];
;  1393          1475  2 $block_check (2, .rtv, rt11, 454);
;  1394          1476  2 rot = rtv [rt11$t_block_0] + (512 * rt11$k_root_block);
;  1395          1477  2
;  1396          1478  2 ! We assume that the directory (thru high_seg not num_segs) is present in memory
;  1397          1479  2 !
;  1398          1480  2 $logic_check (2, .rtv [rt11$v_dir_present], 124);
;  1399          1481  2
;  1400          1482  2 ! Check the consistency of the root segment.  The following tests are order-dependent.  The BLISS optimizer
;  1401          1483  2 ! fold them all together into a single signal and return, but if we had all the tests inside a single IF sta
```

```
;  1402     1484  2 ! the optimizer might have executed them in any order it felt like.
;  1403     1485  2 !
;  1404     1486  3 IF    (.rot [rt11hdr$w_num_segs] EQL 0)
;  1405     1487  2   OR
;  1406     1488  3       (.rot [rt11hdr$w_num_segs] GTRU 31)
;  1407     1489  2 THEN
;  1408     1490  3     BEGIN
;  1409     1491  3     $exch_signal (exch$_rt11_baddirect, 2, .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident]);
;  1410     1492  3     RETURN 0;
;  1411     1493  3     END;
;  1412     1494  3 IF    (.rot [rt11hdr$w_high_seg] EQL 0)
;  1413     1495  2   OR
;  1414     1496  3       (.rot [rt11hdr$w_high_seg] GTRU .rot [rt11hdr$w_num_segs])
;  1415     1497  2 THEN
;  1416     1498  3     BEGIN
;  1417     1499  3     $exch_signal (exch$_rt11_baddirect, 2, .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident]);
;  1418     1500  3     RETURN 0;
;  1419     1501  3     END;
;  1420     1502  3 IF    (.rot [rt11hdr$w_next_seg] GTRU .rot [rt11hdr$w_high_seg])
;  1421     1503  2 THEN
;  1422     1504  3     BEGIN
;  1423     1505  3     $exch_signal (exch$_rt11_baddirect, 2, .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident]);
;  1424     1506  3     RETURN 0;
;  1425     1507  2     END;
;  1426     1508  2
;  1427     1509  2 ! The RT-11 Version 4 DUP objects if more than 119 extra words are specified in an initialize (/Z:120. fails
;  1428     1510  2 ! Since strange things can happen (like directories which can hold < 1 file) if this number is large, we are
;  1429     1511  2 ! going to complain if it exceeds this number too.
;  1430     1512  2 !
;  1431     1513  3 IF (.rot [rt11hdr$w_extra_bytes] GTRU 238)
;  1432     1514  2 THEN
;  1433     1515  3     BEGIN
;  1434     1516  3     $exch_signal (exch$_rt11_baddirect, 2, .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident]);
;  1435     1517  3     RETURN 0;
;  1436     1518  2     END;
;  1437     1519  3 IF ((.rot [rt11hdr$w_extra_bytes] AND 1) NEQ 0) ! It can't be odd either
;  1438     1520  2 THEN
;  1439     1521  3     BEGIN
;  1440     1522  3     $exch_signal (exch$_rt11_baddirect, 2, .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident]);
;  1441     1523  3     RETURN 0;
;  1442     1524  2     END;
;  1443     1525  2
;  1444     1526  2 ! Do a bounds check on the requested segment
;  1445     1527  2 !
;  1446     1528  3 IF    (.number EQL 0)
;  1447     1529  2   OR
;  1448     1530  3       (.number GTRU .rot [rt11hdr$w_high_seg])
;  1449     1531  2 THEN
;  1450     1532  3     BEGIN
;  1451     1533  3     $exch_signal (exch$_rt11_baddirect, 2, .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident]);
;  1452     1534  3     RETURN 0;
;  1453     1535  2     END;
;  1454     1536  2
;  1455     1537  2 ! Looks good, now compute the desired segment as an offset from the root
;  1456     1538  2 !
;  1457     1539  2 seg = .rot + ((.number-1) * rt11$k_dirseglen);
;  1458     1540  2
```

```
;  1459          1541  2 ! Now perform some consistency checks on the segment header
;  1460          1542  2 !
;  1461          1543  3 IF    (.seg [rt11hdr$w_num_segs] NEQ .rot [rt11hdr$w_num_segs])
;  1462          1544  2   OR
;  1463          1545  3       (.seg [rt11hdr$w_next_seg] GTRU .rot [rt11hdr$w_high_seg])
;  1464          1546  2   OR
;  1465          1547  3       (.seg [rt11hdr$w_extra_bytes] NEQ .rot [rt11hdr$w_extra_bytes])
;  1466          1548  2 THEN
;  1467          1549  3     BEGIN
;  1468          1550  3     $exch_signal (exch$_rt11_baddirect, 2, .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident]);
;  1469          1551  3     RETURN 0;
;  1470          1552  2     END;
;  1471          1553  2
;  1472          1554  2 RETURN .seg;
;  1473          1555  1 END;


                                                        .EXTRN  EXCH$_RT11_BADDIRECT

                                        007C 00000      .ENTRY  EXCH$RT11_DIRSEG_GET, Save R2,R3,R4,R5,R6    ; 1427
                    56 00000000G   EF   9E 00002        MOVAB   EXCH$UTIL_BLOCK_CHECK, R6                    ;
                    53       04    AC   D0 00009        MOVL    VOLB, R3                                     ; 1469
                    52 041B00F3    8F   D0 0000D        MOVL    #68878579, R2                                ;
                    51     01C5    8F   3C 00014        MOVZWL  #453, R1                                     ;
                    50             53   D0 00019        MOVL    R3, R0                                       ;
                                   66   16 0001C        JSB     EXCH$UTIL_BLOCK_CHECK                        ;
                    54       54    A3   D0 0001E        MOVL    84(R3), RTV                                  ; 1474
                    52 880E00F5    8F   D0 00022        MOVL    #-2012348171, R2                             ; 1475
                    51     01C6    8F   3C 00029        MOVZWL  #454, R1                                     ;
                    50             54   D0 0002E        MOVL    RTV, R0                                      ;
                                   66   16 00031        JSB     EXCH$UTIL_BLOCK_CHECK                        ;
                    52     0C0E    C4   9E 00033        MOVAB   3086(R4), ROT                                ; 1476
                    13       0C    A4   E8 00038        BLBS    12(RTV), 1$                                  ; 1480
                    7E       7C    8F   9A 0003C        MOVZBL  #124, -(SP)                                  ;
                                   01   DD 00040        PUSHL   #1                                           ;
                          00000000G  8F   DD 00042        PUSHL   #EXCH$_BADLOGIC                              ;
                00000000G  00            03   FB 00048        CALLS   #3, LIB$STOP                         ;
                                         62   B5 0004F  1$:    TSTW    (ROT)                                ; 1486
                                         49   13 00051        BEQL    2$                                   ;
                    1F                   62   B1 00053        CMPW    (ROT), #31                           ; 1488
                                         44   1A 00056        BGTRU   2$                                   ;
                    55       04    A2   3C 00058        MOVZWL  4(ROT), R5                                   ; 1494
                                         3E   13 0005C        BEQL    2$                                   ;
                    55                   62   B1 0005E        CMPW    (ROT), R5                            ; 1496
                                         39   1F 00061        BLSSU   2$                                   ;
                    55       02    A2   B1 00063        CMPW    2(ROT), R5                                   ; 1502
                                         33   1A 00067        BGTRU   2$                                   ;
                00EE    8F       06    A2   B1 00069        CMPW    6(ROT), #238                         ; 1513
                                         2B   1A 0006F        BGTRU   2$                                   ;
                    27       06    A2   E8 00071        BLBS    6(ROT), 2$                                   ; 1519
                    54       08    AC   D0 00075        MOVL    NUMBER, R4                                   ; 1528
                                         21   13 00079        BEQL    2$                                   ;
                    55                   54   D1 0007B        CMPL    R4, R5                               ; 1530
                                         1C   1A 0007E        BGTRU   2$                                   ;
            50      54                   0A   78 00080        ASHL    #10, R4, R0                          ; 1539
                    54     FC00 C042     9E   9E 00084        MOVAB   -1024(R0)[ROT], SEG                  ;
```

```
                          62              64  B1 0008A          CMPW    (SEG), (ROT)                    ; 1543
                                          0D  12 0008D          BNEQ    2$                              ;
                          55        02    A4  B1 0008F          CMPW    2(SEG), R5                      ; 1545
                                          07  1A 00093          BGTRU   2$                              ;
                      06  A2        06    A4  B1 00095          CMPW    6(SEG), 6(ROT)                  ; 1547
                                          17  13 0009A          BEQL    3$                              ;
                                    69    A3  9F 0009C 2$:      PUSHAB  105(R3)                         ; 1550
                                    65    A3  DD 0009F          PUSHL   101(R3)                         ;
                                    02    DD  000A2             PUSHL   #2                              ;
                              00000000G   8F  DD 000A4          PUSHL   #EXCH$_RT11_BADDIRECT           ;
                00000000G     00                04  FB 000AA   CALLS   #4, LIB$SIGNAL                  ;
                                                04  11 000B1   BRB     4$                              ; 1551
                          50                    54  D0 000B3 3$:  MOVL    SEG, R0                     ; 1554
                                                04  000B6       RET                                     ;
                                                50  D4 000B7 4$:  CLRL    R0                          ; 1555
                                                04  000B9       RET                                     ;

; Routine Size:  186 bytes,    Routine Base:  EXCH$RT11_CODE + 0B4A
```

```
;  1475          1556  1 GLOBAL ROUTINE exch$rt11_dirseg_get_nochk (volb : $ref_bblock, number) : jsb_r1r2 =     %SBTTL 'exch$rt11_di
;  1476          1557  2 BEGIN
;  1477          1558  2 !++
;  1478          1559  2 !
;  1479          1560  2 ! FUNCTIONAL DESCRIPTION:
;  1480          1561  2 !
;  1481          1562  2 !       Return a pointer to the requested directory segment without any checking
;  1482          1563  2 !
;  1483          1564  2 ! INPUTS:
;  1484          1565  2 !
;  1485          1566  2 !       volb - pointer to volb which has been connected to the RT-11 device
;  1486          1567  2 !       number - directory segment number in the range 1-31
;  1487          1568  2 !
;  1488          1569  2 ! IMPLICIT INPUTS:
;  1489          1570  2 !
;  1490          1571  2 !       none
;  1491          1572  2 !
;  1492          1573  2 ! OUTPUTS:
;  1493          1574  2 !
;  1494          1575  2 !       none
;  1495          1576  2 !
;  1496          1577  2 ! IMPLICIT OUTPUTS:
;  1497          1578  2 !
;  1498          1579  2 !       none
;  1499          1580  2 !
;  1500          1581  2 ! ROUTINE VALUE:
;  1501          1582  2 !
;  1502          1583  2 !       address of segment, or 0 if any error
;  1503          1584  2 !
;  1504          1585  2 ! SIDE EFFECTS:
;  1505          1586  2 !
;  1506          1587  2 !       error conditions will be signaled
;  1507          1588  2 !--
;  1508          1589  2
;  1509          1590  2 $dbgtrc_prefix ('rt11_dirseg_get_nochk> ');
;  1510          1591  2
;  1511          1592  2 BIND
;  1512          1593  2     rtv = volb [volb$a_vfmt_specific] : $ref_bblock             ! a pointer to the rt11 volb extension
;  1513          1594  2     ;
;  1514          1595  2
;  1515          1596  2 $debug_print_lit ('entry');
;  1516          1597  2
;  1517          1598  2 ! Get the pointer to our volb extension and to the root segment, then compute the
;  1518          1599  2 !
;  1519          1600  2 RETURN rtv [rt11$t_block_0] + (512 * rt11$k_root_block) + ((.number-1) * rt11$k_dirseglen);
;  1520          1601  1 END;


                 52            52            0A  78 00000 EXCH$RT11_DIRSEG_GET_NOCHK::
                                                         ASHL    #10, R2, R2                                   ;  1600
                               52      54    A1  C0 00004         ADDL2   84(VOLB), R2                          ;
                               52    080E    C2  9E 00008         MOVAB   2062(R2), R2                          ;
                               50            52  D0 0000D         MOVL    R2, R0                                ;
                                             05  00010         RSB                                           ;  1601
```

; Routine Size: 17 bytes,   Routine Base: EXCH$RT11_CODE + 0C04

```
;  1522          1602  1 GLOBAL ROUTINE exch$rt11_dirseg_put (volb : $ref_bblock, number) =      %SBTTL 'exch$rt11_dirseg_put (volb,
;  1523          1603  2 BEGIN
;  1524          1604  2 !++
;  1525          1605  2 !
;  1526          1606  2 ! FUNCTIONAL DESCRIPTION:
;  1527          1607  2 !
;  1528          1608  2 !       Write a directory segment back to disk
;  1529          1609  2 !
;  1530          1610  2 ! INPUTS:
;  1531          1611  2 !
;  1532          1612  2 !       volb - pointer to volb which has been connected to the RT-11 device
;  1533          1613  2 !       number - directory segment number in the range 1-31
;  1534          1614  2 !
;  1535          1615  2 ! IMPLICIT INPUTS:
;  1536          1616  2 !
;  1537          1617  2 !       none
;  1538          1618  2 !
;  1539          1619  2 ! OUTPUTS:
;  1540          1620  2 !
;  1541          1621  2 !       none
;  1542          1622  2 !
;  1543          1623  2 ! IMPLICIT OUTPUTS:
;  1544          1624  2 !
;  1545          1625  2 !       none
;  1546          1626  2 !
;  1547          1627  2 ! ROUTINE VALUE:
;  1548          1628  2 !
;  1549          1629  2 !       true if success, error code if problem arose
;  1550          1630  2 !
;  1551          1631  2 ! SIDE EFFECTS:
;  1552          1632  2 !
;  1553          1633  2 !       error conditions will be signaled
;  1554          1634  2 !--
;  1555          1635  2
;  1556          1636  2 $dbgtrc_prefix ('rt11_dirseg_put> ');
;  1557          1637  2
;  1558          1638  2 LOCAL
;  1559          1639  2     blk,                                          ! pbn of block to write
;  1560          1640  2     rtv : $ref_bblock,                            ! a pointer to the rt11 volb extension
;  1561          1641  2     rot : $ref_bblock,                            ! a pointer to the root directory segment
;  1562          1642  2     seg : $ref_bblock,                            ! a pointer to the desired segment
;  1563          1643  2     status
;  1564          1644  2     ;
;  1565          1645  2
;  1566          1646  2 $block_check (2, .volb, volb, 529);
;  1567          1647  2 $trace_print_fao ('* entry - volb !XL, seg !2UL, dircache !XL', .volb, .number, .volb [volb$l_dircache]);
;  1568          1648  2 $logic_check (2, (.volb [volb$v_write]), 146);  ! We shouldn't get this far if we aren't supposed to write t
;  1569          1649  2
;  1570          1650  2 ! Get the pointer to our volb extension and to the root segment
;  1571          1651  2 !
;  1572          1652  2 rtv = .volb [volb$a_vfmt_specific];
;  1573          1653  2 $block_check (2, .rtv, rt11, 528);
;  1574          1654  2 rot = rtv [rt11$t_block_0] + (512 * rt11$k_root_block);
;  1575          1655  2
;  1576          1656  2 ! We assume that the directory (thru high_seg not num_segs) is present in memory
;  1577          1657  2 !
;  1578          1658  2 $logic_check (2, .rtv [rt11$v_dir_present], 142);
```

```
;  1579     1659  2
;  1580     1660  2 ! Do a bounds check on the requested segment
;  1581     1661  2 !
;  1582     1662  3 IF    (.number EQL 0)
;  1583     1663  2   OR
;  1584     1664  3       (.number GTRU .rot [rt11hdr$w_high_seg])
;  1585     1665  2 THEN
;  1586     1666  2     $exch_signal_return (exch$_rt11_baddirect, 2, .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident]);
;  1587     1667  2
;  1588     1668  2 ! Segment number is valid, if caching is on we just set a bit in the cache bitvector    !?? interim cache st
;  1589     1669  2 !
;  1590     1670  2 IF .volb [volb$v_dircache_active]
;  1591     1671  2 THEN
;  1592     1672  3     BEGIN
;  1593     1673  3     BIND
;  1594     1674  3         segbit = volb [volb$l_dircache] : BITVECTOR [32];
;  1595     1675  3     segbit [.number] = true;
;  1596     1676  3     status = true;
;  1597     1677  3     END
;  1598     1678  3
;  1599     1679  3 ! Caching not active, write the segment immediately
;  1600     1680  3 !
;  1601     1681  2 ELSE
;  1602     1682  3     BEGIN
;  1603     1683  3
;  1604     1684  3     ! Looks good, now find the address of the desired segment, and the block number
;  1605     1685  3     !
;  1606     1686  3     seg = .rot + ((.number-1) * rt11$k_dirseglen);
;  1607     1687  3     blk = 2*(.number-1) + rt11$k_root_block;
;  1608     1688  3
;  1609     1689  3     ! Now perform some consistency checks on the segment header
;  1610     1690  3     !
;  1611     1691  4     IF    (.seg [rt11hdr$w_num_segs] NEQ .rot [rt11hdr$w_num_segs])
;  1612     1692  3       OR
;  1613     1693  4         (.seg [rt11hdr$w_next_seg] GTRU .rot [rt11hdr$w_high_seg])
;  1614     1694  3     THEN
;  1615     1695  3         $exch_signal_return (exch$_rt11_baddirect, 2, .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident])
;  1616     1696  3
;  1617     1697  3     ! Write the directory segment back to the disk
;  1618     1698  3     !
;  1619     1699  3     status = exch$io_rt11_write (.volb, .blk, 2, .seg);
;  1620     1700  3
;  1621     1701  3     ! If there was an error, we should give them extra warning that quick action can save the file
;  1622     1702  3     !
;  1623     1703  4     IF  (NOT .status)
;  1624     1704  3     THEN
;  1625     1705  3         $exch_signal (exch$_dire_error);
;  1626     1706  3
;  1627     1707  3 ! Following expression would print additional information to direct the user as to recovery
;  1628     1708  3 ! procedures, so that he could save all the information in the volume by using the correct
;  1629     1709  3 ! copy of the directory which is still in memory.
;  1630     1710  3 !
;  1631     1711  3 !^      IF .exch$a_gbl [excg$v_foreign_command] ! If single command, no hope of recovery, sigh...
;  1632     1712  3 !^      THEN
;  1633     1713  3 !^          $exch_signal (exch$_dire_error)
;  1634     1714  3 !^      ELSE
;  1635     1715  3 !^          $exch_signal (exch$_dire_error, 0, exch$_recover);
```

```
;  1636          1716  3
;  1637          1717  2     END;
;  1638          1718  2
;  1639          1719  2 RETURN .status;
;  1640          1720  1 END;


                                                                .EXTRN  EXCH$_DIRE_ERROR

                                                    0FFC 00000          .ENTRY  EXCH$RT11_DIRSEG_PUT, Save R2,R3,R4,R5,R6,-      ; 1602
                                                                                R7,R8,R9,R10,R11
                                5B 00000000G   8F   D0 00002            MOVL    #EXCH$_RT11_BADDIRECT, R11                       ;
                                5A 00000000G   00   9E 00009            MOVAB   LIB$STOP, R10                                    ;
                                59 00000000G   8F   D0 00010            MOVL    #EXCH$_BADLOGIC, R9                              ;
                                58 00000000G   00   9E 00017            MOVAB   LIB$SIGNAL, R8                                   ;
                                54       04    AC   D0 0001E            MOVL    VOLB, R4                                         ; 1646
                                52 041B00F3    8F   D0 00022            MOVL    #68878579, R2                                    ;
                                51     0211    8F   3C 00029            MOVZWL  #529, R1                                         ;
                                50             54   D0 0002E            MOVL    R4, R0                                           ;
                                   00000000G   EF   16 00031            JSB     EXCH$UTIL_BLOCK_CHECK                            ;
                          0B       48    A4         05 E0 00037            BBS     #5, 72(R4), 1$                                   ; 1648
                                7E         92   8F   9A 0003C            MOVZBL  #146, -(SP)                                      ;
                                              01   DD 00040            PUSHL   #1                                               ;
                                              59   DD 00042            PUSHL   R9                                               ;
                                6A                 03   FB 00044            CALLS   #3, LIB$STOP                                     ;
                                55         54    A4   D0 00047 1$:        MOVL    84(R4), RTV                                      ; 1652
                                52 880E00F5    8F   D0 0004B            MOVL    #-2012348171, R2                                 ; 1653
                                51     0210    8F   3C 00052            MOVZWL  #528, R1                                         ;
                                50             55   D0 00057            MOVL    RTV, R0                                          ;
                                   00000000G   EF   16 0005A            JSB     EXCH$UTIL_BLOCK_CHECK                            ;
                                53     0C0E    C5   9E 00060            MOVAB   3086(R5), ROT                                    ; 1654
                                0B       0C    A5   E8 00065            BLBS    12(RTV), 2$                                      ; 1658
                                7E       8E    8F   9A 00069            MOVZBL  #142, -(SP)                                      ;
                                              01   DD 0006D            PUSHL   #1                                               ;
                                              59   DD 0006F            PUSHL   R9                                               ;
                                6A                 03   FB 00071            CALLS   #3, LIB$STOP                                     ;
                                56       08    AC   D0 00074 2$:        MOVL    NUMBER, R6                                       ; 1662
                                              08   13 00078            BEQL    3$                                               ;
                          56       04    A3         10 00 ED 0007A            CMPZV   #0, #16, 4(ROT), R6                              ; 1664
                                              14   1E 00080            BGEQU   4$                                               ;
                                52                 5B   D0 00082 3$:        MOVL    R11, TEMP                                        ; 1666
                                              69   A4   9F 00085            PUSHAB  105(R4)                                          ;
                                              65   A4   DD 00088            PUSHL   101(R4)                                          ;
                                              02   DD 0008B            PUSHL   #2                                               ;
                                              52   DD 0008D            PUSHL   TEMP                                             ;
                                68                 04   FB 0008F            CALLS   #4, LIB$SIGNAL                                   ;
                                50                 52   D0 00092            MOVL    TEMP, R0                                         ;
                                                   04   00095            RET                                                      ;
                                0A       50    A4   E9 00096 4$:        BLBC    80(R4), 6$                                       ; 1670
                          00       50    A4         56 E2 0009A            BBSS    R6, 80(R4), 5$                                   ; 1675
                                57                 01   D0 0009F 5$:        MOVL    #1, STATUS                                       ; 1676
                                              4F   11 000A2            BRB     9$                                               ; 1670
                          52                 56   0A   78 000A4 6$:        ASHL    #10, R6, R2                                      ; 1686
                                55     FC00 C243   9E 000A8            MOVAB   -1024(R2)[ROT], SEG                              ;
                          52                 56   01   78 000AE            ASHL    #1, R6, BLK                                      ; 1687
                                52                 04   C0 000B2            ADDL2   #4, BLK                                          ;
```

```
                                63              65  B1 000B5          CMPW    (SEG), (ROT)                            ; 1691
                                                07  12 000B8          BNEQ    7$
                             04 A3         02   A5  B1 000BA          CMPW    2(SEG), 4(ROT)                          ; 1693
                                                14  1B 000BF          BLEQU   8$
                                56              5B  D0 000C1 7$:      MOVL    R11, TEMP                               ; 1695
                                            69  A4  9F 000C4          PUSHAB  105(R4)
                                            65  A4  DD 000C7          PUSHL   101(R4)
                                                02  DD 000CA          PUSHL   #2
                                                56  DD 000CC          PUSHL   TEMP
                                68                  04  FB 000CE          CALLS   #4, LIB$SIGNAL
                                50                  56  D0 000D1          MOVL    TEMP, R0
                                                        04  000D4          RET
                                                55  DD 000D5 8$:      PUSHL   SEG                                     ; 1699
                                                02  DD 000D7          PUSHL   #2
                                                52  DD 000D9          PUSHL   BLK
                                                54  DD 000DB          PUSHL   R4
                         00000000G              EF  04  FB 000DD          CALLS   #4, EXCH$IO_RT11_WRITE
                                57              50  D0 000E4          MOVL    R0, STATUS
                                09              57  E8 000E7          BLBS    STATUS, 9$                              ; 1703
                                    00000000G   8F  DD 000EA          PUSHL   #EXCH$_DIRE_ERROR                       ; 1705
                                68                  01  FB 000F0          CALLS   #1, LIB$SIGNAL
                                50                  57  D0 000F3 9$:      MOVL    STATUS, R0                              ; 1719
                                                        04  000F6          RET                                     ; 1720

; Routine Size:  247 bytes,    Routine Base:  EXCH$RT11_CODE + 0C15
```

```
;  1642     1721  1 GLOBAL ROUTINE exch$rt11_expand_filename (ctx : $ref_bblock) =  %SBTTL 'exch$rt11_expand_filename (ctx)'
;  1643     1722  2 BEGIN
;  1644     1723  2 !++
;  1645     1724  2 !
;  1646     1725  2 ! FUNCTIONAL DESCRIPTION:
;  1647     1726  2 !
;  1648     1727  2 !       Convert the information in a directory entry to ascii text.  This involves changing the RADIX-50
;  1649     1728  2 !       filename to ASCII and converting the date to ASCII.
;  1650     1729  2 !
;  1651     1730  2 ! INPUTS:
;  1652     1731  2 !
;  1653     1732  2 !       ctx - pointer to an rt11ctx structure which contains a copy of the directory entry
;  1654     1733  2 !
;  1655     1734  2 ! IMPLICIT INPUTS:
;  1656     1735  2 !
;  1657     1736  2 !       none
;  1658     1737  2 !
;  1659     1738  2 ! OUTPUTS:
;  1660     1739  2 !
;  1661     1740  2 !       ctx - receives ASCII filename info
;  1662     1741  2 !
;  1663     1742  2 ! IMPLICIT OUTPUTS:
;  1664     1743  2 !
;  1665     1744  2 !       none
;  1666     1745  2 !
;  1667     1746  2 ! ROUTINE VALUE:
;  1668     1747  2 !
;  1669     1748  2 !       success or failure if the entry is invalid
;  1670     1749  2 !
;  1671     1750  2 ! SIDE EFFECTS:
;  1672     1751  2 !
;  1673     1752  2 !       error conditions will be signaled
;  1674     1753  2 !--
;  1675     1754  2
;  1676     1755  2 $dbgtrc_prefix ('rt11_expand_filename> ');
;  1677     1756  2
;  1678     1757  2 OWN                                                     ! Read-only own
;  1679     1758  2     months : VECTOR [13, LONG] INITIAL
;  1680     1759  2                 ('Jan-','Feb-','Mar-','Apr-','May-','Jun-','Jul-','Aug-','Sep-','Oct-','Nov-','Dec-','***-')
;  1681     1760  2     ;
;  1682     1761  2
;  1683     1762  2 LOCAL
;  1684     1763  2     year,
;  1685     1764  2     mon,
;  1686     1765  2     day,
;  1687     1766  2     date_desc : VECTOR [2, LONG],
;  1688     1767  2     status,
;  1689     1768  2     ch
;  1690     1769  2     ;
;  1691     1770  2
;  1692     1771  2 $block_check (2, .ctx, rt11ctx, 452);
;  1693     1772  2
;  1694     1773  2 ! Convert the file name from 2 Radix-50 words to 'rt11ctx$s_exp_name' ASCII characters, type from 1 R50 word
;  1695     1774  2 ! 'RT11ctx$s_exp_type' chars
;  1696     1775  2 !
;  1697     1776  2 exch$util_radix50_to_ascii (rt11ctx$s_exp_name, ctx [rt11ctx$l_filename], ctx [rt11ctx$t_exp_name]);
;  1698     1777  2 ch = CH$FIND_CH (rt11ctx$s_exp_name, ctx [rt11ctx$t_exp_name], ' ');
```

```
;  1699     1778  2 ctx [rt11ctx$l_exp_name_len] = (IF .ch EQL 0 THEN rt11ctx$s_exp_name ELSE .ch - ctx [rt11ctx$t_exp_name]);
;  1700     1779  2
;  1701     1780  2 exch$util_radix50_to_ascii (rt11ctx$s_exp_type, ctx [rt11ctx$w_filetype], ctx [rt11ctx$t_exp_type]);
;  1702     1781  2 ch = CH$FIND_CH (rt11ctx$s_exp_type, ctx [rt11ctx$t_exp_type], ' ');
;  1703     1782  2 ctx [rt11ctx$l_exp_type_len] = (IF .ch EQL 0 THEN rt11ctx$s_exp_type ELSE .ch - ctx [rt11ctx$t_exp_type]);
;  1704     1783  2
;  1705     1784  2 ! If file is protected, set the P
;  1706     1785  2 !
;  1707     1786  2 IF .ctx [rt11ctx$v_typ_protected]
;  1708     1787  2 THEN
;  1709     1788  2     CH$MOVE (2, UPLIT BYTE ('p-'), ctx [rt11ctx$t_exp_protected])
;  1710     1789  2 ELSE
;  1711     1790  2     CH$MOVE (2, UPLIT BYTE ('  '), ctx [rt11ctx$t_exp_protected]);
;  1712     1791  2
;  1713     1792  2 ! Create a filename in the standard, non-embedded blank format by concatenating the name, a "." and the type
;  1714     1793  2 !
;  1715     1794  2 ctx [rt11ctx$l_exp_fullname_len] = .ctx [rt11ctx$l_exp_name_len] + 1 + .ctx [rt11ctx$l_exp_type_len];
;  1716     1795  2 CH$COPY (.ctx [rt11ctx$l_exp_name_len], ctx [rt11ctx$t_exp_name],       ! the file name
;  1717     1796  2           1, UPLIT BYTE ('.'),                                          ! the "." separator
;  1718     1797  2           .ctx [rt11ctx$l_exp_type_len], ctx [rt11ctx$t_exp_type],      ! the file type
;  1719     1798  2          %C ' ', rt11ctx$s_exp_fullname, ctx [rt11ctx$t_exp_fullname]); ! the blank-padded result
;  1720     1799  2
;  1721     1800  2 ! Create an ASCII representation of the date
;  1722     1801  2 !
;  1723     1802  2 IF .ctx [rt11ctx$w_date] EQL 0
;  1724     1803  2 THEN
;  1725     1804  2     CH$MOVE (rt11ctx$s_exp_date, UPLIT BYTE (' < nodate >'), ctx [rt11ctx$t_exp_date])
;  1726     1805  2 ELSE
;  1727     1806  3     BEGIN
;  1728     1807  3     year = 1972 + .ctx [rt11ctx$v_year];                  ! 1972 is stored as zero
;  1729     1808  3     day  = .ctx [rt11ctx$v_day];                          ! day is stored 1-31
;  1730     1809  3     mon  = .ctx [rt11ctx$v_month] - 1;                    ! month is 1-12, adjust for vector index
;  1731     1810  3     IF .mon GTRU 12 THEN mon = 12;                        ! point bad months at '***-'
;  1732     1811  3     date_desc [0] = rt11ctx$s_exp_date;                   ! set up a descriptor for FAO
;  1733     1812  3     date_desc [1] = ctx [rt11ctx$t_exp_date];
;  1734     1813  4     IF NOT (status = $fao (%ASCID '!2UC-!AF!4UL', 0, date_desc, .day, 4, months [.mon], .year))
;  1735     1814  3     THEN
;  1736     1815  3         $exch_signal_stop (.status);
;  1737     1816  2     END;
;  1738     1817  2
;  1739   L 1818  2 %IF switch_debug
;  1740   U 1819  2 %THEN
;  1741   U 1820  2         BEGIN
;  1742   U 1821  2         LOCAL
;  1743   U 1822  2             ent_typ;
;  1744   U 1823  2         BIND
;  1745   U 1824  2             a = ctx [rt11ctx$t_entry] : VECTOR [, WORD];
;  1746   U 1825  2         ! Show the entry type
;  1747   U 1826  2         !
;  1748   U 1827  2         ent_typ = (CASE .ctx [rt11ctx$v_type] FROM 0 TO rt11ctx$m_typ_end_segment OF
;  1749   U 1828  2                 SET
;  1750   U 1829  2                         [rt11ent$m_typ_tentative] :     %ASCID 'tent';
;  1751   U 1830  2                         [rt11ent$m_typ_empty] :         %ASCID 'empty';
;  1752   U 1831  2                         [rt11ent$m_typ_permanent] :     %ASCID 'perm';
;  1753   U 1832  2                         [rt11ent$m_typ_end_segment] :   %ASCID 'end';
;  1754   U 1833  2                         [INRANGE, OUTRANGE] :           %ASCID 'unknown';
;  1755   U 1834  2                 TES);
```

```
;  1756        U 1835  2
;  1757        U 1836  2          ! Show what we are returning
;  1758        U 1837  2          !
;  1759        U 1838  2          $debug_print_fao ('!7AS !10<!AF.!AF!> !6UL !AF   !XW !XW !XW !XW !XW !XW !XW',
;  1760        U 1839  2                  .ent_typ,
;  1761        U 1840  2                  .ctx [rt11ctx$l_exp_name_len], ctx [rt11ctx$t_exp_name],
;  1762        U 1841  2                  .ctx [rt11ctx$l_exp_type_len], ctx [rt11ctx$t_exp_type],
;  1763        U 1842  2                  .ctx [rt11ctx$w_blocks],
;  1764        U 1843  2                  rt11ctx$s_exp_date, ctx [rt11ctx$t_exp_date],
;  1765        U 1844  2                  .a[0], .a[1], .a[2], .a[3], .a[4], .a[5], .a[6]);
;  1766        U 1845  2          END;
;  1767          1846  2 %FI
;  1768          1847  2
;  1769          1848  2 RETURN true;
;  1770          1849  1 END;


                                                    .PSECT  EXCH$RT11_PLIT,NOWRT,2

                                    2D 6E 61 4A 00004 MONTHS: .ASCII  \Jan-\                    ;
                                    2D 62 65 46 00008         .ASCII  \Feb-\                    ;
                                    2D 72 61 4D 0000C         .ASCII  \Mar-\                    ;
                                    2D 72 70 41 00010         .ASCII  \Apr-\                    ;
                                    2D 79 61 4D 00014         .ASCII  \May-\                    ;
                                    2D 6E 75 4A 00018         .ASCII  \Jun-\                    ;
                                    2D 6C 75 4A 0001C         .ASCII  \Jul-\                    ;
                                    2D 67 75 41 00020         .ASCII  \Aug-\                    ;
                                    2D 70 65 53 00024         .ASCII  \Sep-\                    ;
                                    2D 74 63 4F 00028         .ASCII  \Oct-\                    ;
                                    2D 76 6F 4E 0002C         .ASCII  \Nov-\                    ;
                                    2D 63 65 44 00030         .ASCII  \Dec-\                    ;
                                    2D 2A 2A 2A 00034         .ASCII  \***-\                    ;
                                       2D 70    00038 P.AAB:  .ASCII  \p-\                      ;
                                       20 20    0003A P.AAC:  .ASCII  \  \                      ;
                                          2E    0003C P.AAD:  .ASCII  \.\                       ;
                3E 20 65 74 61 64 6F 6E 20 3C 20 0003D P.AAE:  .ASCII  \ < nodate >\            ;
4C 55 34 21 46 41 21 2D 4C 55 32 21    00048 P.AAG:  .ASCII  \!2UL-!AF!4UL\                    ;
                        010E0000C      00054 P.AAF:  .LONG   17694732                          ;
                        00000000'      00058         .ADDRESS P.AAG                            ;

                                                    .EXTRN  SYS$FAO

                                                    .PSECT  EXCH$RT11_CODE,NOWRT,2

                                          07FC 00000        .ENTRY  EXCH$RT11_EXPAND_FILENAME, Save R2,R3,R4,-  ; 1721
                                                            R5,R6,R7,R8,R9,R10                  ;
                                5A 00000000G   EF   9E 00002        MOVAB   EXCH$UTIL_RADIX50_TO_ASCII, R10     ;
                                59     0000'   CF   9E 00009        MOVAB   P.AAB, R9                           ;
                                5E             08   C2 0000E        SUBL2   #8, SP                              ;
                                56       04    AC   D0 00011        MOVL    CTX, R6                             ; 1771
                                52 008200F4    8F   D0 00015        MOVL    #8519924, R2                        ;
                                51     01C4    8F   3C 0001C        MOVZWL  #452, R1                            ;
                                50             56   D0 00021        MOVL    R6, R0                              ;
                                   00000000G   EF   16 00024        JSB     EXCH$UTIL_BLOCK_CHECK               ;
                                         5E    A6   9F 0002A        PUSHAB  94(R6)                              ; 1776
                                         3A    A6   9F 0002D        PUSHAB  58(R6)                              ;
```

```
                                                 06   DD 00030          PUSHL   #6
                                                 03   FB 00032          CALLS   #3, EXCH$UTIL_RADIX50_TO_ASCII
                                      6A         03   FB 00032
                                   5E A6         06   20 00035          LOCC    #32, #6, 94(R6)                                  ;1777
                                                 02   12 0003A          BNEQ    1$
                                                 51   D4 0003C          CLRL    R1
                                      52         51   D0 0003E 1$:      MOVL    R1, CH
                                                 05   12 00041          BNEQ    2$                                               ;1778
                                      51         06   D0 00043          MOVL    #6, R1
                                                 08   11 00046          BRB     3$
                                      50      5E A6   9E 00048 2$:      MOVAB   94(R6), R0
                                   51 52         50   C3 0004C          SUBL3   R0, CH, R1
                                4A A6            51   D0 00050 3$:      MOVL    R1, 74(R6)
                                                64 A6   9F 00054          PUSHAB  100(R6)                                          ;1780
                                                3E A6   9F 00057          PUSHAB  62(R6)
                                                 03   DD 0005A          PUSHL   #3
                                      6A         03   FB 0005C          CALLS   #3, EXCH$UTIL_RADIX50_TO_ASCII
                                   64 A6         03   20 0005F          LOCC    #32, #3, 100(R6)                                 ;1781
                                                 02   12 00064          BNEQ    4$
                                                 51   D4 00066          CLRL    R1
                                      52         51   D0 00068 4$:      MOVL    R1, CH
                                                 05   12 0006B          BNEQ    5$                                               ;1782
                                      51         03   D0 0006D          MOVL    #3, R1
                                                 08   11 00070          BRB     6$
                                      50      64 A6   9E 00072 5$:      MOVAB   100(R6), R0
                                   51 52         50   C3 00076          SUBL3   R0, CH, R1
                                4E A6            51   D0 0007A 6$:      MOVL    R1, 78(R6)
                                                39 A6   95 0007E          TSTB    57(R6)                                           ;1786
                                                 06   18 00081          BGEQ    7$
                                52 A6            69   B0 00083          MOVW    P.AAB, 82(R6)                                    ;1788
                                                 05   11 00087          BRB     8$
                                52 A6         02 A9   B0 00089 7$:      MOVW    P.AAC, 82(R6)                                    ;1790
                             50 4A A6         4E A6   C1 0008E 8$:      ADDL3   78(R6), 74(R6), R0                               ;1794
                                46 A6         01 A0   9E 00094          MOVAB   1(R0), 70(R6)
                                      58         0A   D0 00099          MOVL    #10, R8                                          ;1797
                                      57      54 A6   9E 0009C          MOVAB   84(R6), R7                                       ;1798
58                  20 5E A6         4A A6         2C 000A0          MOVC5   74(R6), 94(R6), #32, R8, (R7)
                                                 67   000A7
                                                 1D   18 000A8          BGEQ    9$
                                      57      4A A6   C0 000AA          ADDL2   74(R6), R7
                                      58      4A A6   C2 000AE          SUBL2   74(R6), R8
58                  20 04 A9                     01   2C 000B2          MOVC5   #1, P.AAD, #32, R8, (R7)
                                                 67   000B8
                                                 0C   18 000B9          BGEQ    9$
                                                 57   D6 000BB          INCL    R7
                                                 58   D7 000BD          DECL    R8
58                  20 64 A6         4E A6         2C 000BF          MOVC5   78(R6), 100(R6), #32, R8, (R7)
                                                 67   000C6
                                                44 A6   B5 000C7 9$:      TSTW    68(R6)                                           ;1802
                                                 08   12 000CA          BNEQ    10$
                67 A6 05 A9                      0B   28 000CC          MOVC3   #11, P.AAE, 103(R6)                              ;1804
                                                 4F   11 000D2          BRB     12$
52                  44 A6            05         00   EF 000D4 10$:      EXTZV   #0, #5, 68(R6), YEAR                             ;1807
                                      52   07B4   C2   9E 000DA          MOVAB   1972(R2), YEAR
51                  44 A6            05         05   EF 000DF          EXTZV   #5, #5, 68(R6), DAY                              ;1808
50                  45 A6            05         02   EF 000E5          EXTZV   #2, #5, 69(R6), MON                              ;1809
                                                 50   D7 000EB          DECL    MON
                                      0C         50   D1 000ED          CMPL    MON, #12                                         ;1810
```

```
                                    03  1B 000F0          BLEQU   11$                                                    ;
                                50  0C  D0 000F2          MOVL    #12, MON                                               ;
                                6E  0B  D0 000F5 11$:      MOVL    #11, DATE_DESC                                         ; 1811
                          04    AE  67  A6  9E 000F8       MOVAB   103(R6), DATE_DESC+4                                   ; 1812
                                    52  DD 000FD          PUSHL   YEAR                                                   ; 1813
                                CC  A940  DF 000FF        PUSHAL  MONTHS[MON]                                            ;
                                    04  DD 00103         PUSHL   #4                                                     ;
                                    51  DD 00105         PUSHL   DAY                                                    ;
                                10  AE  9F 00107         PUSHAB  DATE_DESC                                              ;
                                    7E  D4 0010A         CLRL    -(SP)                                                  ;
                                1C  A9  9F 0010C         PUSHAB  P.AAF                                                  ;
                  00000000G     00  07  FB 0010F         CALLS   #7, SYS$FAO                                            ;
                                0A  50  E8 00116         BLBS    STATUS, 12$                                            ;
                                    50  DD 00119         PUSHL   STATUS                                                 ; 1815
                  00000000G     00  01  FB 0011B         CALLS   #1, LIB$STOP                                           ;
                                    04  00122          RET                                                            ;
                                50      01  D0 00123 12$:  MOVL    #1, R0                                                 ; 1848
                                    04  00126          RET                                                            ; 1849

; Routine Size:  295 bytes,    Routine Base:  EXCH$RT11_CODE + 0D0C
```

```
;  1772     1850  1 GLOBAL ROUTINE exch$rt11_format_current_date (ent : $ref_bblock) : NOVALUE jsb_r1 =     %SBTTL 'exch$rt11_fo
;  1773     1851  2 BEGIN
;  1774     1852  2 !++
;  1775     1853  2 !
;  1776     1854  2 ! FUNCTIONAL DESCRIPTION:
;  1777     1855  2 !
;  1778     1856  2 !       Format the current date, placing it into the date field of an RT-11 directory entry
;  1779     1857  2 !
;  1780     1858  2 ! INPUT:
;  1781     1859  2 !
;  1782     1860  2 !       ent - pointer to the directory entry
;  1783     1861  2 !
;  1784     1862  2 ! IMPLICIT INPUTS:
;  1785     1863  2 !
;  1786     1864  2 !       none
;  1787     1865  2 !
;  1788     1866  2 ! OUTPUTS:
;  1789     1867  2 !
;  1790     1868  2 !       none
;  1791     1869  2 !
;  1792     1870  2 ! IMPLICIT OUTPUTS:
;  1793     1871  2 !
;  1794     1872  2 !       none
;  1795     1873  2 !
;  1796     1874  2 ! ROUTINE VALUE:
;  1797     1875  2 !
;  1798     1876  2 !       none
;  1799     1877  2 !
;  1800     1878  2 ! SIDE EFFECTS:
;  1801     1879  2 !
;  1802     1880  2 !       none
;  1803     1881  2 !--
;  1804     1882  2
;  1805     1883  2 $dbgtrc_prefix ('rt11_format_current_date> ');
;  1806     1884  2
;  1807     1885  2 LOCAL
;  1808     1886  2     timbuf : VECTOR [7, WORD]
;  1809     1887  2     ;
;  1810     1888  2
;  1811     1889  2 BIND
;  1812     1890  2     year = timbuf [0] : WORD, month = timbuf [1] : WORD, day = timbuf [2] : WORD;
;  1813     1891  2
;  1814     1892  2 $numtim (timbuf=timbuf);
;  1815     1893  2
;  1816     1894  2 ent [rt11ent$v_year]  = .year - 1972;
;  1817     1895  2 ent [rt11ent$v_month] = .month;
;  1818     1896  2 ent [rt11ent$v_day]   = .day;
;  1819     1897  2
;  1820     1898  2 RETURN;
;  1821     1899  1 END;


                                                .EXTRN  SYS$NUMTIM

                        5E              10  C2 00000 EXCH$RT11_FORMAT_CURRENT_DATE::
                                                SOBL2   #16, SP                                 ; 1850
```

```
                                                 51   DD 00003         PUSHL   R1                                                      ;
                                                 7E   D4 00005         CLRL    -(SP)                                                   ; 1892
                                           08    AE   9F 00007         PUSHAB  TIMBUF
                                 00000000G 00    02   FB 0000A         CALLS   #2, SYS$NUMTIM
                                           50    04   AE 3C 00011      MOVZWL  YEAR, R0                                                ; 1894
                                           50  F84C   C0 9E 00015      MOVAB   -1972(R0), R0
                                     51    6E         0C C1 0001A      ADDL3   #12, ENT, R1
                                61   05    00         50 F0 0001E      INSV    R0, #0, #5, (R1)
                                     50    6E         0D C1 00023      ADDL3   #13, ENT, R0                                            ; 1895
                                60   05    02    06   AE F0 00027      INSV    MONTH, #2, #5, (R0)
                                     50    8E         0C C1 0002D      ADDL3   #12, ENT, R0                                            ; 1896
                                60   05    05    04   AE F0 00031      INSV    DAY, #5, #5, (R0)
                                           5E         10 C0 00037      ADDL2   #16, SP                                                 ; 1899
                                                      05 0003A         RSB                                                             ;

; Routine Size:  59 bytes,    Routine Base:  EXCH$RT11_CODE + 0E33
```

```
;  1823     1900  1 GLOBAL ROUTINE exch$rt11_mount (volb : $ref_bblock) =   %SBTTL 'exch$rt11_mount (volb)'
;  1824     1901  2 BEGIN
;  1825     1902  2 !++
;  1826     1903  2 !
;  1827     1904  2 ! FUNCTIONAL DESCRIPTION:
;  1828     1905  2 !
;  1829     1906  2 !       Perform RT-11 volume specific mount processing
;  1830     1907  2 !
;  1831     1908  2 ! INPUTS:
;  1832     1909  2 !
;  1833     1910  2 !       volb - pointer to volb which has been connected to the RT-11 device
;  1834     1911  2 !
;  1835     1912  2 ! IMPLICIT INPUTS:
;  1836     1913  2 !
;  1837     1914  2 !       none
;  1838     1915  2 !
;  1839     1916  2 ! OUTPUTS:
;  1840     1917  2 !
;  1841     1918  2 !       none
;  1842     1919  2 !
;  1843     1920  2 ! IMPLICIT OUTPUTS:
;  1844     1921  2 !
;  1845     1922  2 !       none
;  1846     1923  2 !
;  1847     1924  2 ! ROUTINE VALUE:
;  1848     1925  2 !
;  1849     1926  2 !       none
;  1850     1927  2 !
;  1851     1928  2 ! SIDE EFFECTS:
;  1852     1929  2 !
;  1853     1930  2 !       none
;  1854     1931  2 !--
;  1855     1932  2
;  1856     1933  2 $dbgtrc_prefix ('rt11_mount> ');
;  1857     1934  2
;  1858     1935  2 LOCAL
;  1859     1936  2     rtv : $ref_bblock,                          ! a pointer to the rt11 volb extension
;  1860     1937  2     seg : $ref_bblock,                          ! a pointer to the current directory segment
;  1861     1938  2     blocks,
;  1862     1939  2     status
;  1863     1940  2     ;
;  1864     1941  2
;  1865     1942  2 $debug_print_lit ('entry');
;  1866     1943  2
;  1867     1944  2 $block_check (1, .volb, volb, 462);
;  1868     1945  2
;  1869     1946  2 ! Allocate and initialize our volb extension
;  1870     1947  2 !
;  1871     1948  2 $logic_check (2, (.volb [volb$a_vfmt_specific] EQL 0), 127);
;  1872     1949  2 rtv = exch$util_vm_allocate (exchblk$s_rt11);
;  1873     1950  2 CH$FILL (0, rt11$k_end_zero - rt11$k_start_zero, .rtv + rt11$k_start_zero);     ! Set part of block to nulls
;  1874     1951  2 volb [volb$a_vfmt_specific] = .rtv;
;  1875     1952  2 $block_init (.rtv, rt11);
;  1876     1953  2
;  1877     1954  2 ! Read the first part of the volume, the home block on pbn 1
;  1878     1955  2 !
;  1879     1956  3 IF NOT (status = exch$io_rt11_read (.volb, 1, 1, rtv [rt11$t_block_1]))
```

```
;  1880     1957  2 THEN
;  1881     1958  2     RETURN .status;
;  1882     1959  2
;  1883     1960  2 ! Read the the first directory segment, found on blocks 6 and 7.
;  1884     1961  2 !
;  1885     1962  2 IF NOT (status = exch$io_rt11_read (.volb, rt11$k_root_block, 2,
;  1886     1963  2                                     rtv [rt11$t_block_0] + (512 * rt11$k_root_block)))
;  1887     1964  2 THEN
;  1888     1965  2     RETURN .status;
;  1889     1966  2
;  1890     1967  2 ! Use the segment get routine to verify this first segment.  We temporarily set the present flag because the
;  1891     1968  2 ! get routine expects to see it.
;  1892     1969  2 !
;  1893     1970  2 rtv [rt11$v_dir_present] = true;
;  1894     1971  2 seg = exch$rt11_dirseg_get (.volb, 1);          ! Get a pointer to the root segment
;  1895     1972  2 rtv [rt11$v_dir_present] = false;
;  1896     1973  2 IF .seg EQL 0                                   ! DIRSEG_GET will have signalled any problems
;  1897     1974  2 THEN
;  1898     1975  2     RETURN exch$_rt11_baddirect;
;  1899     1976  2
;  1900     1977  2 ! Read in the rest of the directory if it is a multi-segment directory
;  1901     1978  2 !
;  1902     1979  2 IF .seg [rt11hdr$w_high_seg] GTRU 1
;  1903     1980  2 THEN
;  1904     1981  3     BEGIN
;  1905     1982  3     LOCAL
;  1906     1983  3         blk_cnt, addr;
;  1907     1984  3     blk_cnt = 2 * (.seg [rt11hdr$w_high_seg] - 1);      ! Segs are 2 blocks, but one is already in memory
;  1908     1985  3     addr = rt11$k_dirseglen + .seg;                     ! Get a pointer to space after the root segment
;  1909     1986  4     IF NOT (status = exch$io_rt11_read (.volb, rt11$k_root_block+2, .blk_cnt, .addr))
;  1910     1987  3     THEN
;  1911     1988  3         RETURN .status;
;  1912     1989  2     END;
;  1913     1990  2
;  1914     1991  2 ! Now we are ok, set the flag that it is present
;  1915     1992  2 !
;  1916     1993  2 rtv [rt11$v_dir_present] = true;                ! This means present through HIGH_SEG, not NUM_SEGS
;  1917     1994  2
;  1918     1995  2 ! Verify the directory
;  1919     1996  2 !
;  1920     1997  2 status = exch$rtacp_verify_directory (.volb);
;  1921     1998  2
;  1922     1999  2 ! Set the volume type string
;  1923     2000  2 !
;  1924     2001  2 CH$MOVE (5, UPLIT BYTE ('RT-11'), volb [volb$t_vol_type]);
;  1925     2002  2 volb [volb$l_vol_type_len] = 5;
;  1926     2003  2
;  1927     2004  2 ! Initialize the directory cache to the state of the global /CACHE qualifier
;  1928     2005  2 !
;  1929     2006  2 volb [volb$l_dircache] = .exch$a_gbl [excg$v_q_cache];
;  1930     2007  2
;  1931     2008  2 RETURN .status;
;  1932     2009  1 END;


                                                .PSECT  EXCH$RT11_PLIT,NOWRT,2
```

```
                        31  31  2D  54  52  0005C P.AAH:  .ASCII  \RT-11\                                    ;

                                                          .PSECT  EXCH$RT11_CODE,NOWRT,2

                                            01FC 00000    .ENTRY  EXCH$RT11_MOUNT, Save R2,R3,R4,R5,R6,R7,R8 ; 1900
                    58 00000000G    EF   9E 00002         MOVAB   EXCH$IO_RT11_READ, R8                      ;
                    56       04     AC   D0 00009         MOVL    VOLB, R6                                   ; 1944
                    52 041B00F3     8F   D0 0000D         MOVL    #68878579, R2                              ;
                    51     01CE     8F   3C 00014         MOVZWL  #462, R1                                   ;
                    50              56   D0 00019         MOVL    R6, R0                                     ;
                        00000000G   EF   16 0001C         JSB     EXCH$UTIL_BLOCK_CHECK                      ;
                              54    A6   D5 00022         TSTL    84(R6)                                     ; 1948
                                    13   13 00025         BEQL    1$                                         ;
                    7E        7F    8F   9A 00027         MOVZBL  #127, -(SP)                                ;
                                    01   DD 0002B         PUSHL   #1                                         ;
                        00000000G   8F   DD 0002D         PUSHL   #EXCH$_BADLOGIC                            ;
                00000000G   00      03   FB 00033         CALLS   #3, LIB$STOP                               ;
                    7E      880E    8F   3C 0003A 1$:     MOVZWL  #34830, -(SP)                              ; 1949
                00000000G   EF      01   FB 0003F         CALLS   #1, EXCH$UTIL_VM_ALLOCATE                  ;
                    52              50   D0 00046         MOVL    R0, RTV                                    ;
                    53        0C    A2   9E 00049         MOVAB   12(RTV), R3                                ; 1950
                                    63   B4 0004D         CLRW    (R3)                                       ;
                          54  A6            52   D0 0004F         MOVL    RTV, 84(R6)                        ; 1951
                          08  A2      880E  8F   B0 00053         MOVW    #-30706, 8(RTV)                    ; 1952
                          0A  A2            0B   8E 00059         MNEGB   #11, 10(RTV)                       ;
                                    020E    C2   9F 0005D         PUSHAB  526(RTV)                           ; 1956
                                            01   DD 00061         PUSHL   #1                                 ;
                                            01   DD 00063         PUSHL   #1                                 ;
                                            56   DD 00065         PUSHL   R6                                 ;
                                        68       04   FB 00067         CALLS   #4, EXCH$IO_RT11_READ          ;
                                        57       50   D0 0006A         MOVL    R0, STATUS                     ;
                                        77       57   E9 0006D         BLBC    STATUS, 4$                     ;
                                              0C0E    C2   9F 00070         PUSHAB  3086(RTV)                  ; 1963
                                                      02   DD 00074         PUSHL   #2                         ; 1962
                                                      06   DD 00076         PUSHL   #6                         ;
                                                      56   DD 00078         PUSHL   R6                         ;
                                        68            04   FB 0007A         CALLS   #4, EXCH$IO_RT11_READ      ;
                                        57            50   D0 0007D         MOVL    R0, STATUS                 ;
                                        64            57   E9 00080         BLBC    STATUS, 4$                 ;
                                        63            01   88 00083         BISB2   #1, (R3)                   ; 1970
                                                      01   DD 00086         PUSHL   #1                         ; 1971
                                                      56   DD 00088         PUSHL   R6                         ;
                                  FC4D  CF            02   FB 0008A         CALLS   #2, EXCH$RT11_DIRSEG_GET   ;
                                        63            01   8A 0008F         BICB2   #1, (R3)                   ; 1972
                                                      50   D5 00092         TSTL    SEG                        ; 1973
                                                      08   12 00094         BNEQ    2$                         ;
                                        50  00000000G  8F   D0 00096         MOVL    #EXCH$_RT11_BADDIRECT, R0  ; 1975
                                                      04   00 0009D         RET                                ;
                                        01        04  A0   B1 0009E 2$:     CMPW    4(SEG), #1                 ; 1979
                                                      1F   1B 000A2         BLEQU   3$                         ;
                                        51        04  A0   3C 000A4         MOVZWL  4(SEG), R1                 ; 1984
                                                      51   D7 000A8         DECL    R1                         ;
                                        51            02   C4 000AA         MULL2   #2, BLK_CNT                ;
                                        50      0400  C0   9E 000AD         MOVAB   1024(R0), ADDR             ; 1985
```

```
                                                 50  DD 000B2          PUSHL   ADDR                                                    ; 1986
                                                 51  DD 000B4          PUSHL   BLK_CNT                                                 ;
                                                 08  DD 000B6          PUSHL   #8                                                      ;
                                                 56  DD 000B8          PUSHL   R6                                                      ;
                                             68  04  FB 000BA          CALLS   #4, EXCH$IO_RT11_READ                                   ;
                                             57  50  D0 000BD          MOVL    R0, STATUS                                              ;
                                             24  57  E9 000C0          BLBC    STATUS, 4$                                              ;
                                             63  01  88 000C3 3$:      BISB2   #1, (R3)                                                ; 1993
                                                 56  DD 000C6          PUSHL   R6                                                      ; 1997
                                 00000000G   EF  01  FB 000C8          CALLS   #1, EXCH$RTACP_VERIFY_DIRECTORY                         ;
                                             57  50  D0 000CF          MOVL    R0, STATUS                                              ;
                         5D  A6      0000'   CF  05  28 000D2          MOVC3   #5, P.AAH, 93(R6)                                       ; 2001
                                       59    A6  05  D0 000D9          MOVL    #5, 89(R6)                                              ; 2002
              50  A6 00000000G   FF          01  01  EF 000DD          EXTZV   #1, #1, @EXCH$A_GBL, 80(R6)                             ; 2006
                                             50  57  D0 000E7 4$:      MOVL    STATUS, R0                                              ; 2008
                                                 04  000EA              RET                                                             ; 2009

; Routine Size:  235 bytes,    Routine Base:  EXCH$RT11_CODE + 0E6E
```

```
;  1934     2010  1 GLOBAL ROUTINE exch$rt11_open_file =    %SBTTL 'exch$rt11_open_file'
;  1935     2011  2 BEGIN
;  1936     2012  2 !++
;  1937     2013  2 !
;  1938     2014  2 ! FUNCTIONAL DESCRIPTION:
;  1939     2015  2 !
;  1940     2016  2 !       Perform RT-11 volume specific open processing
;  1941     2017  2 !
;  1942     2018  2 ! INPUT:
;  1943     2019  2 !
;  1944     2020  2 !       none
;  1945     2021  2 !
;  1946     2022  2 ! IMPLICIT INPUTS:
;  1947     2023  2 !
;  1948     2024  2 !       copy work area
;  1949     2025  2 !
;  1950     2026  2 ! OUTPUTS:
;  1951     2027  2 !
;  1952     2028  2 !       none
;  1953     2029  2 !
;  1954     2030  2 ! IMPLICIT OUTPUTS:
;  1955     2031  2 !
;  1956     2032  2 !       filb - receive info pertaining to the open file
;  1957     2033  2 !
;  1958     2034  2 ! ROUTINE VALUE:
;  1959     2035  2 !
;  1960     2036  2 !       true if able to open a file, false otherwise
;  1961     2037  2 !
;  1962     2038  2 ! SIDE EFFECTS:
;  1963     2039  2 !
;  1964     2040  2 !       none
;  1965     2041  2 !--
;  1966     2042  2
;  1967     2043  2 $dbgtrc_prefix ('rt11_open_file> ');
;  1968     2044  2
;  1969     2045  2 LOCAL
;  1970     2046  2     out_filb                                    : $ref_bblock,
;  1971     2047  2     status
;  1972     2048  2     ;
;  1973     2049  2
;  1974     2050  2 BIND
;  1975     2051  2     copy      = exch$a_gbl [excg$a_copy_work]   : $ref_bblock,
;  1976     2052  2     inp_filb = copy [copy$a_inp_filb]           : $ref_bblock,
;  1977     2053  2     ctx       = inp_filb [filb$a_context]       : $ref_bblock,
;  1978     2054  2     namb      = inp_filb [filb$a_assoc_namb]    : $ref_bblock,
;  1979     2055  2     nam_nam   = namb [namb$q_name]              : $desc_block,      ! Get name and type components from
;  1980     2056  2     nam_typ   = namb [namb$q_type]              : $desc_block,      !  the namb
;  1981     2057  2     volb      = inp_filb [filb$a_assoc_volb]    : $ref_bblock,
;  1982     2058  2     inp_namb = copy [copy$a_inp_namb]           : $ref_bblock
;  1983     2059  2     ;
;  1984     2060  2
;  1985     2061  2 $debug_print_lit ('entry');
;  1986     2062  2
;  1987     2063  2 $block_check (2, .inp_filb, filb, 463);
;  1988     2064  2 $block_check (2, .namb, namb, 464);
;  1989     2065  2 $block_check (2, .volb, volb, 465);
;  1990     2066  2
```

```
;  1991          2067  2 ! Make sure that the output filb points to a valid filb
;  1992          2068  2 !
;  1993          2069  2 IF (out_filb = .copy [copy$a_out_filb]) EQL 0
;  1994          2070  2 THEN
;  1995          2071  2     out_filb = .inp_filb;
;  1996          2072  2 $block_check (2, .out_filb, filb, 472);
```

```
;  1998     2073  2 ! If the context pointer is null, then allocate and initialize it.
;  1999     2074  2 !
;  2000     2075  2 IF .ctx EQL 0
;  2001     2076  2 THEN
;  2002     2077  2     ctx = exch$util_rt11ctx_allocate (.volb, .inp_filb) ! Create an RT11 context block
;  2003     2078  2
;  2004     2079  2 ! If non-null, we are doing a subsequent lookup in a wildcard search
;  2005     2080  2 !
;  2006     2081  2 ELSE
;  2007     2082  3     BEGIN
;  2008     2083  3
;  2009     2084  3     ! If not wildcard, then we must be done
;  2010     2085  3     !
;  2011     2086  4     IF NOT (.namb [namb$v_wild_name] OR .namb [namb$v_wild_type])
;  2012     2087  3     THEN
;  2013     2088  3         RETURN false;
;  2014     2089  3
;  2015     2090  3     $block_check (2, .ctx, rt11ctx, 446);
;  2016     2091  3
;  2017     2092  2     END;
;  2018     2093  2
;  2019     2094  2 ! Make sure that we haven't crossed signals someplace
;  2020     2095  2 !
;  2021     2096  2 $logic_check (4, (.ctx [rt11ctx$a_assoc_filb] EQL .inp_filb), 128);
;  2022     2097  2 $logic_check (4, (.ctx [rt11ctx$a_assoc_volb] EQL .volb), 129);
;  2023     2098  2
;  2024     2099  2 ! We assume that the file name and type fields in the namb are adjacent.  If they aren't, the next call to
;  2025     2100  2 ! exch$rtacp_find_file will choke.
;  2026     2101  2 !
;  2027     2102  2 $logic_check (3, (.nam_typ [dsc$a_pointer] EQL (.nam_nam [dsc$w_length] + .nam_nam [dsc$a_pointer])), 154);
```

```
;  2029     2103  3 IF (
;  2030     2104  3     IF .copy [copy$v_reopen_input]                ! If we are retrying, then reuse the context block
;  2031     2105  3     THEN
;  2032     2106  3         1
;  2033     2107  3     ELSE                                          ! Otherwise skip to the next file
;  2034     2108  3         exch$rtacp_find_file (.ctx, .nam_nam [dsc$a_pointer], .nam_nam [dsc$w_length] + .nam_typ [dsc$w_leng
;  2035     2109  3     )
;  2036     2110  3 THEN
;  2037     2111  3     BEGIN
;  2038     2112  3
;  2039     2113  3     ! Do not find a .BAD file unless it is explicitly specified
;  2040     2114  3     !
;  2041     2115  3     IF .ctx [rt11ctx$w_filetype] EQL r50_bad
;  2042     2116  3     THEN
;  2043     2117  3         IF  .inp_namb [namb$v_wild_name]              ! If the found file was not explicitly named
;  2044     2118  3            OR                                         !  then skip to the next file by calling
;  2045     2119  3             .inp_namb [namb$v_wild_type]              !  ourselves again
;  2046     2120  3         THEN
;  2047     2121  3             RETURN exch$rt11_open_file ();
;  2048     2122  3
;  2049     2123  3     ! Create the result name string in the filb
;  2050     2124  3     !
;  2051     2125  3     inp_filb [filb$l_result_name_len] = .volb [volb$l_vol_ident_len]    ! Length of volume ident
;  2052     2126  3                                       + .ctx [rt11ctx$l_exp_fullname_len];    !  plus rt fullname
;  2053     2127  3     $logic_check (3, (.inp_filb [filb$l_result_name_len] LEQU filb$s_result_name), 130);
;  2054     2128  3     CH$COPY (.volb [volb$l_vol_ident_len], volb [volb$t_vol_ident],     ! Volume name
;  2055     2129  3                 .ctx [rt11ctx$l_exp_fullname_len], ctx [rt11ctx$t_exp_fullname],
;  2056     2130  3                 0, filb$s_result_name, inp_filb [filb$t_result_name]);
;  2057     2131  3
;  2058     2132  3     $debug_print_fao ('Found "!AF"', .inp_filb [filb$l_result_name_len], inp_filb [filb$t_result_name]);
;  2059     2133  3
;  2060     2134  3     ! Define a record stream for this file
;  2061     2135  3     !
;  2062     2136  3     ctx [rt11ctx$l_cur_byte]    = 0;                                ! Context is the first byte in
;  2063     2137  3     ctx [rt11ctx$l_cur_block]   = .ctx [rt11ctx$l_start_block]; !  the first block of the file
;  2064     2138  3     ctx [rt11ctx$l_eof_block]   = .ctx [rt11ctx$l_start_block] + .ctx [rt11ctx$w_blocks] - 1;
;  2065     2139  3     inp_filb [filb$l_block_count]       = .ctx [rt11ctx$w_blocks];      ! Put the size in the filb
;  2066     2140  3     inp_filb [filb$a_record]    = 0;                                ! No valid record or length
;  2067     2141  3     inp_filb [filb$l_record_len]        = 0;
;  2068     2142  3
;  2069     2143  3     ! Make sure that the record format in the filb is correct
;  2070     2144  3     !
;  2071     2145  3     exch$cmd_fetch_recfmt_implied (.inp_filb, ctx [rt11ctx$t_exp_type]);
;  2072     2146  3
;  2073     2147  3     ! For RT-11 we can treat block transfer mode as fixed 512
;  2074     2148  3     !
;  2075     2149  3     IF .out_filb [filb$b_transfer_mode] EQL filb$k_xfrm_block
;  2076     2150  3       OR
;  2077     2151  3        .inp_filb [filb$b_transfer_mode] EQL filb$k_xfrm_block
;  2078     2152  3     THEN
;  2079     2153  4         BEGIN
;  2080     2154  4         inp_filb [filb$b_rec_format]  = filb$k_rfmt_fixed;
;  2081     2155  4         inp_filb [filb$l_fixed_len]   = 512;
;  2082     2156  3         END;
;  2083     2157  3
;  2084     2158  3     ! Clear all the flags except the ones we want by writing the masks into the longword
;  2085     2159  3     !
```

```
;  2086          2160  3     ctx [rt11ctx$l_flags] = rt11ctx$m_stream_active;    ! A record stream is currently active
;  2087          2161  3     inp_filb [filb$v_files_found] = true;                ! One or more files have been opened
;  2088          2162  3
;  2089          2163  3     ! Set up the i/o and record buffer (for when we can't use locate mode)
;  2090          2164  3     !
;  2091          2165  3     IF .ctx [rt11ctx$a_buffer] EQL 0            ! If doing wildcards buffer might be there
;  2092          2166  3     THEN
;  2093          2167  3         ctx [rt11ctx$a_buffer] = exch$util_vm_allocate (ctx$k_buffer_length);
;  2094          2168  3
;  2095          2169  3     ! Set the block pointers so that we will advance the buffer on the first get
;  2096          2170  3     !
;  2097          2171  3     ctx [rt11ctx$l_buf_base_block] = .ctx [rt11ctx$l_start_block];
;  2098          2172  3     ctx [rt11ctx$l_buf_high_block] = .ctx [rt11ctx$l_start_block] - 1;
;  2099          2173  3
;  2100          2174  3     ! Save the addresses of our routines for this volume and record format.
;  2101          2175  3     !
;  2102          2176  3     inp_filb [filb$a_close_routine] = exch$rt11_close_file;
;  2103          2177  3     inp_filb [filb$a_put_routine] = 0;
;  2104          2178  3     inp_filb [filb$a_get_routine] = exch$pdp_get;
;  2105          2179  3
;  2106          2180  3     RETURN true;                                 ! True means its open
;  2107          2181  3     END;
;  2108          2182  2
;  2109          2183  2 ! If no files were found, then scream and shout
;  2110          2184  2 !
;  2111          2185  2 IF NOT .inp_filb [filb$v_files_found]
;  2112          2186  2 THEN
;  2113          2187  3     BEGIN
;  2114          2188  3     REGISTER
;  2115          2189  3         fao_desc = 0 : $ref_bblock;
;  2116          2190  3
;  2117          2191  3     ! Concatenate the volume name to the file name and type fields
;  2118          2192  3     !
;  2119          2193  3     fao_desc = exch$util_fao_buffer (%ASCID '!AF!AF', .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident],
;  2120          2194  3                                      .nam_nam [dsc$w_length] + .nam_typ [dsc$w_length], .nam_nam [dsc$a_pointer]);
;  2121          2195  3     $exch_signal (exch$_filenotfound, 1, .fao_desc);
;  2122          2196  3     RETURN rms$_fnf;
;  2123          2197  3
;  2124          2198  2     END;
;  2125          2199  2
;  2126          2200  2 ! Normal exit, return a 0
;  2127          2201  2 !
;  2128          2202  2 RETURN 0;
;  2129          2203  1 END;


                                                                .PSECT  EXCH$RT11_PLIT,NOWRT,2

                                                        00061   .BLKB   3
                00  00  46  41  21  46  41  21  00064 P.AAJ:   .ASCII  \!AF!AF\<0><0>                                    ;
                                    010E0006    0006C P.AAI:   .LONG   17694726                                          ;
                                    00000000'   00070          .ADDRESS P.AAJ                                        ;

                                                                .EXTRN  EXCH$_FILENOTFOUND

                                                                .PSECT  EXCH$RT11_CODE,NOWRT,2
```

```
                                 0FFC 00000          .ENTRY  EXCH$RT11_OPEN_FILE, Save R2,R3,R4,R5,R6,-      ; 2010
                                                             R7,R8,R9,R10,R11
                          5E      04  C2 00002        SUBL2   #4, SP
               50 00000000G   EF      04  C1 00005        ADDL3   #4, EXCH$A_GBL, R0                          ; 2051
                          53          60  D0 0000D        MOVL    (R0), R3                                    ; 2052
                          57      3C  A3  D0 00010        MOVL    60(R3), R7                                  ; 2053
                          59      18  A7  D0 00014        MOVL    24(R7), R9                                  ; 2055
                          52      50  A9  9F 00018        PUSHAB  80(R9)
                          52 035B00FA  8F  D0 0001B        MOVL    #56295674, R2                               ; 2063
                          51    01CF  8F  3C 00022        MOVZWL  #463, R1
                          50          57  D0 00027        MOVL    R7, R0
                         00000000G    EF  16 0002A        JSB     EXCH$UTIL_BLOCK_CHECK
                          52 010A00F7  8F  D0 00030        MOVL    #17432823, R2                               ; 2064
                          51    01D0  8F  3C 00037        MOVZWL  #464, R1
                          50          59  D0 0003C        MOVL    R9, R0
                         00000000G    EF  16 0003F        JSB     EXCH$UTIL_BLOCK_CHECK
                          58      1C  A7  D0 00045        MOVL    28(R7), R8                                  ; 2065
                          52 041B00F3  8F  D0 00049        MOVL    #68878579, R2
                          51    01D1  8F  3C 00050        MOVZWL  #465, R1
                          50          58  D0 00055        MOVL    R8, R0
                         00000000G    EF  16 00058        JSB     EXCH$UTIL_BLOCK_CHECK
                          5B      44  A3  D0 0005E        MOVL    68(R3), OUT_FILB                            ; 2069
                                      03  12 00062        BNEQ    1$
                          5B          57  D0 00064        MOVL    R7, OUT_FILB                                ; 2071
                          52 035B00FA  8F  D0 00067 1$:    MOVL    #56295674, R2                               ; 2072
                          51    01D8  8F  3C 0006E        MOVZWL  #472, R1
                          50          5B  D0 00073        MOVL    OUT_FILB, R0
                         00000000G    EF  16 00076        JSB     EXCH$UTIL_BLOCK_CHECK
                              20      A7  D5 0007C        TSTL    32(R7)                                      ; 2075
                                      11  12 0007F        BNEQ    2$
                                      57  DD 00081        PUSHL   R7                                          ; 2077
                                      58  DD 00083        PUSHL   R8
                00000000G    EF          02  FB 00085        CALLS   #2, EXCH$UTIL_RT11CTX_ALLOCATE
                      20     A7          50  D0 0008C        MOVL    R0, 32(R7)
                                      23  11 00090        BRB     4$
             08       6C    A9          01  E0 00092 2$:    BBS     #1, 108(R9), 3$                             ; 2086
             03       6C    A9          02  E0 00097        BBS     #2, 108(R9), 3$
                                  0147    31 0009C        BRW     13$
                          52 008200F4  8F  D0 0009F 3$:    MOVL    #8519924, R2                                ; 2090
                          51    01BE  8F  3C 000A6        MOVZWL  #446, R1
                          50      20  A7  D0 000AB        MOVL    32(R7), R0
                         00000000G    EF  16 000AF        JSB     EXCH$UTIL_BLOCK_CHECK
             22       34    A3          02  E0 000B5 4$:    BBS     #2, 52(R3), 5$                              ; 2104
                          50      00  BE  3C 000BA        MOVZWL  @0(SP), R0                                  ; 2108
                          51      58  A9  3C 000BE        MOVZWL  88(R9), R1
                                  6140    9F 000C2        PUSHAB  (R1)[R0]
             52       04    AE          04  C1 000C5        ADDL3   #4, 4(SP), R2
                                      62  DD 000CA        PUSHL   (R2)
                              20      A7  DD 000CC        PUSHL   32(R7)
                00000000G    EF          03  FB 000CF        CALLS   #3, EXCH$RTACP_FIND_FILE
                          03              50  E8 000D6        BLBS    R0, 5$
                                  00CA    31 000D9        BRW     12$
                          56      20  A7  D0 000DC 5$:    MOVL    32(R7), R6                                  ; 2115
                    0CAC  8F      3E  A6  B1 000E0        CMPW    62(R6), #3244
                                      14  12 000E6        BNEQ    7$
                          50      40  A3  D0 000E8        MOVL    64(R3), R0                                  ; 2117
```

```
                05      6C    A0              01  E0 000EC          BBS     #1, 108(R0), 6$                                  ;
                06      6C    A0              02  E1 000F1          BBC     #2, 108(R0), 7$                                  ; 2119
                     FF05     CF              00  FB 000F6 6$:      CALLS   #0, EXCH$RT11_OPEN_FILE                          ; 2121
                                              04  000FB            RET                                                      ;
          3A    A7      65    A8        46    A6  C1 000FC 7$:      ADDL3   70(R6), 101(R8), 58(R7)                          ; 2126
                04      AE      0100    8F    3C 00103            MOVZWL  #256, 4(SP)                                      ; 2129
                      5A          5A    A7    9E 00109            MOVAB   90(R7), R10                                      ; 2130
    04  AE            00      69    A8    65  A8 2C 0010D         MOVC5   101(R8), 105(R8), #0, 4(SP), (R10)               ;
                                              6A  00115                                                                    ;
                                              12  18 00116        BGEQ    8$                                               ;
                      5A          65    A8    C0 00118            ADDL2   101(R8), R10                                     ;
                04    AE          65    A8    C2 0011C            SUBL2   101(R8), 4(SP)                                   ;
    04  AE            00      54    A6    46  A6 2C 00121         MOVC5   70(R6), 84(R6), #0, 4(SP), (R10)                 ;
                                              6A  00129                                                                    ;
                                        24    A6  D4 0012A 8$:      CLRL    36(R6)                                           ; 2136
                1C    A6          72    A6    D0 0012D            MOVL    114(R6), 28(R6)                                  ; 2137
                50    A6          40    A6    3C 00132            MOVZWL  64(R6), R0                                       ; 2138
                50    A6          72    A6    C0 00136            ADDL2   114(R6), R0                                      ;
                20    A6          FF    A0    9E 0013A            MOVAB   -1(R0), 32(R6)                                   ;
                3E    A7          40    A6    3C 0013F            MOVZWL  64(R6), 62(R7)                                   ; 2139
                                  42    A7    7C 00144            CLRQ    66(R7)                                           ; 2141
                                  64    A6    9F 00147            PUSHAB  100(R6)                                          ; 2145
                                              57  DD 0014A        PUSHL   R7                                               ;
          00000000G   EF                      02  FB 0014C        CALLS   #2, EXCH$CMD_FETCH_RECFMT_IMPLIED                 ;
                      01          29    AB    91 00153            CMPB    41(OUT_FILB), #1                                 ; 2149
                                              06  13 00157        BEQL    9$                                               ;
                      01          29    A7    91 00159            CMPB    41(R7), #1                                       ; 2151
                                              0A  12 0015D        BNEQ    10$                                              ;
                28    A7                    02  90 0015F 9$:      MOVB    #2, 40(R7)                                       ; 2154
                35    A7        0200        8F  3C 00163        MOVZWL  #512, 53(R7)                                     ; 2155
                28    A6                    01  D0 00169 10$:     MOVL    #1, 40(R6)                                       ; 2160
                2B    A7                    08  88 0016D        BISB2   #8, 43(R7)                                       ; 2161
                                        18    A6  D5 00171        TSTL    24(R6)                                           ; 2165
                                              10  12 00174        BNEQ    11$                                              ;
                      7E        1800        8F  3C 00176        MOVZWL  #6144, -(SP)                                     ; 2167
          00000000G   EF                      01  FB 0017B        CALLS   #1, EXCH$UTIL_VM_ALLOCATE                        ;
                18    A6                    50  D0 00182        MOVL    R0, 24(R6)                                       ;
                2C    A6          72    A6    D0 00186 11$:     MOVL    114(R6), 44(R6)                                  ; 2171
    30  A6          72    A6                  01  C3 0018B        SUBL3   #1, 114(R6), 48(R6)                              ; 2172
                4A    A7        F131        CF  9E 00191        MOVAB   EXCH$RT11_CLOSE_FILE, 74(R7)                     ; 2176
                                        56    A7  D4 00197        CLRL    86(R7)                                           ; 2177
                52    A7  00000000G         EF  9E 0019A        MOVAB   EXCH$PDP_GET, 82(R7)                             ; 2178
                50                          01  D0 001A2        MOVL    #1, R0                                           ; 2180
                                              04  001A5        RET                                                      ;
          3B        2B    A7                  03  E0 001A6 12$:     BBS     #3, 43(R7), 13$                                  ; 2185
          52              6E                  04  C1 001AB        ADDL3   #4, (SP), R2                                     ; 2194
                                              62  DD 001AF        PUSHL   (R2)                                             ;
                      50          04    BE    3C 001B1            MOVZWL  @4(SP), R0                                       ;
                      51          58    A9    3C 001B5            MOVZWL  88(R9), R1                                       ;
                                      6140    9F 001B9            PUSHAB  (R1)[R0]                                         ;
                                        69    A8  9F 001BC        PUSHAB  105(R8)                                          ; 2193
                                        65    A8  DD 001BF        PUSHL   101(R8)                                          ;
                                    0000'     CF  9F 001C2        PUSHAB  P.AAI                                            ;
          00000000G   EF                      05  FB 001C6        CALLS   #5, EXCH$UTIL_FAO_BUFFER                         ;
                                              50  DD 001CD        PUSHL   FAO_DESC                                         ; 2195
                                              01  DD 001CF        PUSHL   #1                                               ;
                                00000000G     8F  DD 001D1        PUSHL   #EXCH$_FILENOTFOUND                              ;
```

```
                 00000000G  00          03  FB 001D7          CALLS   #3, LIB$SIGNAL                    ;
                            50 00018292  8F  D0 001DE          MOVL    #98962, R0                       ; 2196
                                         04 001E5          RET                                      ;
                                         50 D4 001E6 13$:  CLRL    R0                               ; 2203
                                         04 001E8          RET                                      ;

; Routine Size:  489 bytes,    Routine Base:  EXCH$RT11_CODE + 0F59
```

```
;  2131          2204  1 GLOBAL ROUTINE exch$rt11_write_cleanup (volb : $ref_bblock) : NOVALUE = %SBTTL 'exch$rt11_write_cleanup (vol
;  2132          2205  2 BEGIN
;  2133          2206  2 !++
;  2134          2207  2 !
;  2135          2208  2 ! FUNCTIONAL DESCRIPTION:
;  2136          2209  2 !
;  2137          2210  2 !       Finish writing to the volume.  Clear file marks and flush caches.
;  2138          2211  2 !
;  2139          2212  2 ! INPUTS:
;  2140          2213  2 !
;  2141          2214  2 !       volb - pointer to volb which has been connected to the RT-11 device
;  2142          2215  2 !
;  2143          2216  2 ! IMPLICIT INPUTS:
;  2144          2217  2 !
;  2145          2218  2 !       none
;  2146          2219  2 !
;  2147          2220  2 ! OUTPUTS:
;  2148          2221  2 !
;  2149          2222  2 !       none
;  2150          2223  2 !
;  2151          2224  2 ! IMPLICIT OUTPUTS:
;  2152          2225  2 !
;  2153          2226  2 !       none
;  2154          2227  2 !
;  2155          2228  2 ! ROUTINE VALUE:
;  2156          2229  2 !
;  2157          2230  2 !       none
;  2158          2231  2 !
;  2159          2232  2 ! SIDE EFFECTS:
;  2160          2233  2 !
;  2161          2234  2 !       error conditions will be signaled
;  2162          2235  2 !--
;  2163          2236  2
;  2164          2237  2 $dbgtrc_prefix ('rt11_write_cleanup> ');
;  2165          2238  2
;  2166          2239  2 $trace_print_fao ('entry - volb !XL', .volb);
;  2167          2240  2
;  2168          2241  2 exch$rt11_zero_marks (.volb);                    ! Clear all the file marks
;  2169          2242  2
;  2170          2243  2 exch$rt11_dircache_stop (.volb);                 ! Clear caching and flush the directory
;  2171          2244  2
;  2172          2245  2 RETURN;
;  2173          2246  1 END;


                                        0000 00000          .ENTRY  EXCH$RT11_WRITE_CLEANUP, Save nothing       ; 2204
                                  04    AC   DD 00002       PUSHL   VOLB                                        ; 2241
                        0000V     CF         01   FB 00005  CALLS   #1, EXCH$RT11_ZERO_MARKS
                                  04    AC   DD 0000A       PUSHL   VOLB                                        ; 2243
                        F90F      CF         01   FB 0000D  CALLS   #1, EXCH$RT11_DIRCACHE_STOP
                                             04   00012     RET                                                 ; 2246

; Routine Size:  19 bytes,    Routine Base:  EXCH$RT11_CODE + 1142
```

```
;  2175     2247  1 GLOBAL ROUTINE exch$rt11_write_prepare (volb : $ref_bblock) : NOVALUE = %SBTTL 'exch$rt11_write_prepare (vol
;  2176     2248  2 BEGIN
;  2177     2249  2 !++
;  2178     2250  2 !
;  2179     2251  2 ! FUNCTIONAL DESCRIPTION:
;  2180     2252  2 !
;  2181     2253  2 !       Prepare to write to the volume.  Set up caches and clear file marks.
;  2182     2254  2 !
;  2183     2255  2 ! INPUTS:
;  2184     2256  2 !
;  2185     2257  2 !       volb - pointer to volb which has been connected to the RT-11 device
;  2186     2258  2 !
;  2187     2259  2 ! IMPLICIT INPUTS:
;  2188     2260  2 !
;  2189     2261  2 !       none
;  2190     2262  2 !
;  2191     2263  2 ! OUTPUTS:
;  2192     2264  2 !
;  2193     2265  2 !       none
;  2194     2266  2 !
;  2195     2267  2 ! IMPLICIT OUTPUTS:
;  2196     2268  2 !
;  2197     2269  2 !       none
;  2198     2270  2 !
;  2199     2271  2 ! ROUTINE VALUE:
;  2200     2272  2 !
;  2201     2273  2 !       none
;  2202     2274  2 !
;  2203     2275  2 ! SIDE EFFECTS:
;  2204     2276  2 !
;  2205     2277  2 !       error conditions will be signaled
;  2206     2278  2 !--
;  2207     2279  2
;  2208     2280  2 $dbgtrc_prefix ('rt11_write_prepare> ');
;  2209     2281  2
;  2210     2282  2 $trace_print_fao ('entry - volb !XL', .volb);
;  2211     2283  2
;  2212     2284  2 exch$rt11_dircache_start (.volb);                ! Start caching on the directory
;  2213     2285  2
;  2214     2286  2 exch$rt11_zero_marks (.volb);                    ! Clear all the file marks
;  2215     2287  2
;  2216     2288  2 RETURN;
;  2217     2289  1 END;


                                        0000 00000          .ENTRY  EXCH$RT11_WRITE_PREPARE, Save nothing       ; 2247
                                04      AC   DD 00002       PUSHL   VOLB                                        ; 2284
                        F86F    CF           01   FB 00005  CALLS   #1, EXCH$RT11_DIRCACHE_START                ;
                                04      AC   DD 0000A       PUSHL   VOLB                                        ; 2286
                        0000V   CF           01   FB 0000D  CALLS   #1, EXCH$RT11_ZERO_MARKS                    ;
                                             04 00012       RET                                                 ; 2289

; Routine Size:  19 bytes,    Routine Base:  EXCH$RT11_CODE + 1155
```

```
;  2219          2290  1 GLOBAL ROUTINE exch$rt11_zero_marks (volb : $ref_bblock) : NOVALUE =    %SBTTL 'exch$rt11_zero_marks (volb)'
;  2220          2291  2 BEGIN
;  2221          2292  2 !++
;  2222          2293  2 !
;  2223          2294  2 ! FUNCTIONAL DESCRIPTION:
;  2224          2295  2 !
;  2225          2296  2 !       Clear the file marks in every entry in the directory.  EXCHANGE flags the JOB byte (rt11ent$b_job) o
;  2226          2297  2 !       directory entry while it is entering files.  The flags are cleared before and after each COPY comman
;  2227          2298  2 !       therebye giving us the means to recognize that we might have to delete a file which we just created.
;  2228          2299  2 !
;  2229          2300  2 ! INPUTS:
;  2230          2301  2 !
;  2231          2302  2 !       volb - pointer to volb which has been connected to the RT-11 device
;  2232          2303  2 !
;  2233          2304  2 ! IMPLICIT INPUTS:
;  2234          2305  2 !
;  2235          2306  2 !       none
;  2236          2307  2 !
;  2237          2308  2 ! OUTPUTS:
;  2238          2309  2 !
;  2239          2310  2 !       none
;  2240          2311  2 !
;  2241          2312  2 ! IMPLICIT OUTPUTS:
;  2242          2313  2 !
;  2243          2314  2 !       none
;  2244          2315  2 !
;  2245          2316  2 ! ROUTINE VALUE:
;  2246          2317  2 !
;  2247          2318  2 !       none
;  2248          2319  2 !
;  2249          2320  2 ! SIDE EFFECTS:
;  2250          2321  2 !
;  2251          2322  2 !       error conditions will be signaled
;  2252          2323  2 !--
;  2253          2324  2
;  2254          2325  2 $dbgtrc_prefix ('exch$rt11_zero_marks> ');
;  2255          2326  2
;  2256          2327  2 LOCAL
;  2257          2328  2     seg : $ref_bblock,                        ! a pointer to the current directory segment
;  2258          2329  2     cur : $ref_bblock,                        ! a pointer to the current directory entry
;  2259          2330  2     seg_num,                                  ! current segment number
;  2260          2331  2     ent_len                                   ! length of a directory entry
;  2261          2332  2     ;
;  2262          2333  2
;  2263          2334  2 $trace_print_fao ('entry - volb !XL', .volb);
;  2264          2335  2
;  2265          2336  2 $block_check (2, .volb, volb, 459);
;  2266          2337  2 $logic_check (2, (.volb [volb$v_write]), 199);  ! We shouldn't get this far if we aren't supposed to write t
```

```
;  2268     2338  2 ! Loop through all the directory entries to clear the mark flag.
;  2269     2339  2 !
;  2270     2340  2 seg_num = 1;                                   ! Start with the first directory segment
;  2271     2341  2 WHILE .seg_num NEQ 0
;  2272     2342  2 DO
;  2273     2343  3     BEGIN
;  2274     2344  3
;  2275     2345  3     ! Get a pointer to the current segment, return if error
;  2276     2346  3     !
;  2277     2347  3     seg = exch$rt11_dirseg_get (.volb, .seg_num);
;  2278     2348  3     $logic_check (2, (.seg NEQ 0), 197);
;  2279     2349  3     ent_len = rt11ent$k_length + .seg [rt11hdr$w_extra_bytes];  ! Actually the same for all segments
;  2280     2350  3
;  2281     2351  3     ! Get a pointer to the first directory entry
;  2282     2352  3     !
;  2283     2353  3     cur = .seg + rt11hdr$k_length;
;  2284     2354  3
;  2285     2355  3     ! Look through the segment
;  2286     2356  3     !
;  2287     2357  4     WHILE (.cur LSSU (.seg + rt11$k_dirseglen))
;  2288     2358  3     DO
;  2289     2359  4         BEGIN
;  2290     2360  4
;  2291     2361  4         CASE .cur [rt11ent$v_type] FROM 0 TO rt11ent$m_typ_end_segment OF
;  2292     2362  4         SET
;  2293     2363  4         [rt11ent$m_typ_tentative, rt11ent$m_typ_permanent, rt11ent$m_typ_empty] :
;  2294     2364  5                 BEGIN
;  2295     2365  5
;  2296     2366  5                 ! If the marker isn't clear, clear it and remember that the segment has been changed
;  2297     2367  5                 !
;  2298     2368  5                 IF .cur [rt11ent$b_job] NEQ 0
;  2299     2369  5                 THEN
;  2300     2370  6                     BEGIN
;  2301     2371  6                     cur [rt11ent$b_job] = 0;
;  2302     2372  6                     exch$rt11_dirseg_put (.volb, .seg_num);     ! Caching is on, this won't give us an I/O n
;  2303     2373  5                     END;
;  2304     2374  4                 END;
;  2305     2375  4
;  2306     2376  4         [rt11ent$m_typ_end_segment] :
;  2307     2377  4                 EXITLOOP;
;  2308     2378  4
;  2309     2379  4         [INRANGE, OUTRANGE] :
;  2310     2380  4                 ;
;  2311     2381  4
;  2312     2382  4         TES;
;  2313     2383  4
;  2314     2384  4         cur = .cur + .ent_len;                  ! Skip to the next entry
;  2315     2385  4
;  2316     2386  3         END;
;  2317     2387  3
;  2318     2388  3     seg_num = .seg [rt11hdr$w_next_seg];        ! Skip to the next segment
;  2319     2389  3
;  2320     2390  2     END;
;  2321     2391  2
;  2322     2392  2 RETURN;
;  2323     2393  1 END;
```

```
                                07FC 00000          .ENTRY  EXCH$RT11_ZERO_MARKS, Save R2,R3,R4,R5,R6,- ; 2290
                                                            R7,R8,R9,R10
                    5A 00000000G  00  9E 00002      MOVAB   LIB$STOP, R10
                    59 00000000G  8F  D0 00009      MOVL    #EXCH$_BADLOGIC, R9
                    55       04   AC  D0 00010      MOVL    VOLB, R5                                    ; 2336
                    52 041B00F3   8F  D0 00014      MOVL    #68878579, R2
                    51     01CB   8F  3C 0001B      MOVZWL  #459, R1
                    50           55  D0 00020      MOVL    R5, R0
                00000000G        EF  16 00023      JSB     EXCH$UTIL_BLOCK_CHECK
          0B    48   A5           05  E0 00029      BBS     #5, 72(R5), 1$                              ; 2337
                7E     C7        8F  9A 0002E      MOVZBL  #199, -(SP)
                                 01  DD 00032      PUSHL   #1
                                 59  DD 00034      PUSHL   R9
                6A               03  FB 00036      CALLS   #3, LIB$STOP
                54               01  D0 00039 1$:  MOVL    #1, SEG_NUM                                 ; 2340
                                 68  13 0003C 2$:  BEQL    9$                                          ; 2341
                                 54  DD 0003E      PUSHL   SEG_NUM                                     ; 2347
                                 55  DD 00040      PUSHL   R5
            F99B  CF             02  FB 00042      CALLS   #2, EXCH$RT11_DIRSEG_GET
                53               50  D0 00047      MOVL    R0, SEG
                                 0B  12 0004A      BNEQ    3$                                          ; 2348
                7E     C5        8F  9A 0004C      MOVZBL  #197, -(SP)
                                 01  DD 00050      PUSHL   #1
                                 59  DD 00052      PUSHL   R9
                6A               03  FB 00054      CALLS   #3, LIB$STOP
                57       06      A3  3C 00057 3$:  MOVZWL  6(SEG), ENT_LEN                             ; 2349
                57               0E  C0 0005B      ADDL2   #14, ENT_LEN
                52       0A      A3  9E 0005E      MOVAB   10(R3), CUR                                 ; 2353
                56     0400      C3  9E 00062      MOVAB   1024(R3), R6                                ; 2357
                56               52  D1 00067 4$:  CMPL    CUR, R6
                                 34  1E 0006A      BGEQU   8$
          58    01   A2           04  EF 0006C      EXTZV   #0, #4, 1(CUR), R8                          ; 2361
                     08           00  CF 00072      CASEL   R8, #0, #8
          0025       0014         0014      0025    .WORD   7$-5$,-
          0025       0025         0025      0014            6$-5$,-
                                            002A            6$-5$,-
                                                            7$-5$,-
                                                            6$-5$,-
                                                            7$-5$,-
                                                            7$-5$,-
                                                            7$-5$,-
                                                            8$-5$
                                 11  11 00088      BRB     7$
                       0B        A2  95 0008A 6$:  TSTB    11(CUR)                                     ; 2368
                                 0C  13 0008D      BEQL    7$
                       0B        A2  94 0008F      CLRB    11(CUR)                                     ; 2371
                                 54  DD 00092      PUSHL   SEG_NUM                                     ; 2372
                                 55  DD 00094      PUSHL   R5
                FA12  CF             02  FB 00096      CALLS   #2, EXCH$RT11_DIRSEG_PUT
                52               57  C0 0009B 7$:  ADDL2   ENT_LEN, CUR                               ; 2384
                                 C7  11 0009E      BRB     4$                                          ; 2357
                54       02      A3  3C 000A0 8$:  MOVZWL  2(SEG), SEG_NUM                             ; 2388
                                 96  11 000A4      BRB     2$                                          ; 2341
```

```
                                    04 000A6 9$:      RET                                     ; 2393

; Routine Size:  167 bytes,    Routine Base:  EXCH$RT11_CODE + 1168
```

```
; 2325          2394  1 END
; 2326          2395  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL, LIB$STOP

;                               PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  EXCH$RT11_CODE                 4623  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  EXCH$RT11_PLIT                  116  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         22         0         1000        00:01.9
;  _$255$DUA28:[EXCHNG.OBJ]EXCLIB.L32;1   1151        188        16           79        00:01.4



;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:EXCRT11/OBJ=OBJ$:EXCRT11 MSRC$:EXCRT11/UPDATE=(ENH$:EXCRT11)

; Size:          4623 code + 116 data bytes
; Run Time:         01:22.6
; Elapsed Time:     04:13.3
; Lines/CPU Min:     1739
; Lexemes/CPU-Min: 23430
; Memory Used:  374 pages
; Compilation Complete
```

EXCRT11.
LIS

EXCMAIN
LIS

EXCMSG
LIS

EXCPDP.
LIS

EXCMOUN
LIS

EXCSHOW
LIS

F11A

F11AACP
MAP

EXCRTNAM
LIS

EXCRTACP
LIS

EXCUTIL
LIS