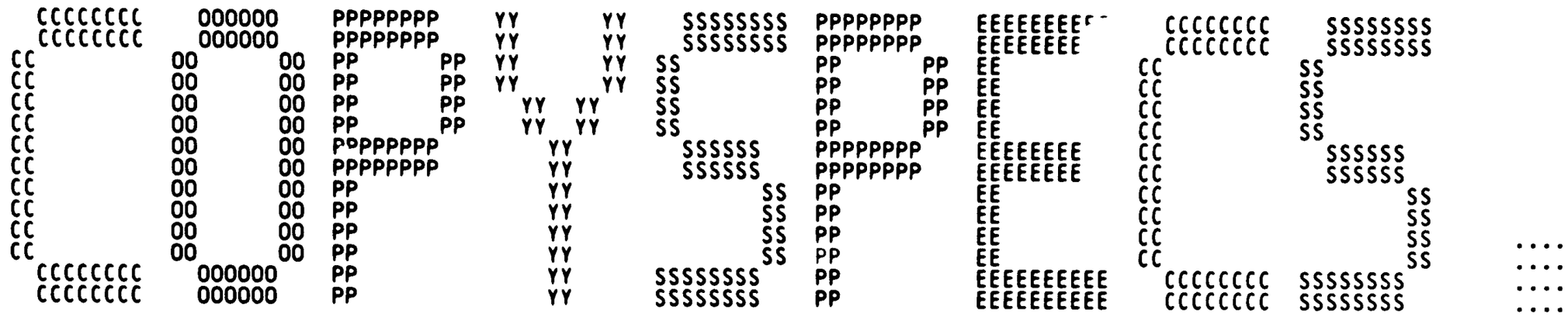
```
   CCCCCCCCCCC     OOOOOOOOO     PPPPPPPPPPPP     YYY         YYY
   CCCCCCCCCCC     OOOOOOOOO     PPPPPPPPPPPP     YYY         YYY
   CCCCCCCCCCC     OOOOOOOOO     PPPPPPPPPPPP     YYY         YYY
CCC               OOO     OOO   PPP         PPP  YYY         YYY
CCC               OOO     OOO   PPP         PPP  YYY         YYY
CCC               OOO     OOO   PPP         PPP  YYY         YYY
CCC               OOO     OOO   PPP         PPP     YYY   YYY
CCC               OOO     OOO   PPP         PPP     YYY   YYY
CCC               OOO     OOO   PPP         PPP     YYY   YYY
CCC               OOO     OOO   PPPPPPPPPPPP           YYY
CCC               OOO     OOO   PPPPPPPPPPPP           YYY
CCC               OOO     OOO   PPPPPPPPPPPP           YYY
CCC               OOO     OOO   PPP                    YYY
CCC               OOO     OOO   PPP                    YYY
CCC               OOO     OOO   PPP                    YYY
CCC               OOO     OOO   PPP                    YYY
CCC               OOO     OOO   PPP                    YYY
CCC               OOO     OOO   PPP                    YYY
   CCCCCCCCCCC     OOOOOOOOO     PPP                    YYY
   CCCCCCCCCCC     OOOOOOOOO     PPP                    YYY
   CCCCCCCCCCC     OOOOOOOOO     PPP                    YYY
```


CCCCCCCC 000000 PPPPPPPP YY YY SSSSSSSS PPPPPPPP EEEEEEEE CCCCCCCC SSSSSSSS
CCCCCCCC 000000 PPPPPPPP YY YY SSSSSSSS PPPPPPPP EEEEEEEE CCCCCCCC SSSSSSSS
CC 00 00 PP PP YY YY SS PP PP EE CC SS
CC 00 00 PP PP YY YY SS PP PP EE CC SS
CC 00 00 PP PP YY YY SS PP PP EE CC SS
CC 00 00 PP PP YY YY SS PP PP EE CC SS
CC 00 00 PPPPPPPP YY SSSSSS PPPPPPPP EEEEEEEE CC SSSSSS
CC 00 00 PPPPPPPP YY SSSSSS PPPPPPPP EEEEEEEE CC SSSSSS
CC 00 00 PP YY SS PP EE CC SS
CC 00 00 PP YY SS PP EE CC SS
CC 00 00 PP YY SS PP EE CC SS
CC 00 00 PP YY SS PP EE CC SS
CCCCCCCC 000000 PP YY SSSSSSSS PP EEEEEEEEEE CCCCCCCC SSSSSSSS
CCCCCCCC 000000 PP YY SSSSSSSS PP EEEEEEEEEE CCCCCCCC SSSSSSSS
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 copyspecs (  ! Manipulates input and output specifications for COPY utility
;     2          0002  0                       LANGUAGE (BLISS32),
;     3          0003  0                       IDENT = 'V04-000'
;     4          0004  0                       ) =
;     5          0005  1 BEGIN
;     6          0006  1
;     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:     COPY Command
;    33          0033  1 !
;    34          0034  1 ! ABSTRACT:
;    35          0035  1 !
;    36          0036  1 !       This module obtains input and output specifications from the CLI and opens
;    37          0037  1 !       the associated files.
;    38          0038  1 !
;    39          0039  1 ! ENVIRONMENT:
;    40          0040  1 !
;    41          0041  1 !       VAX/VMS operating system, unprivileged user mode utility,
;    42          0042  1 !       operates at non-AST level.
;    43          0043  1 !
;    44          0044  1 !--
;    45          0045  1 !++
;    46          0046  1 !
;    47          0047  1 ! AUTHOR:       Carol Peters,   CREATION DATE:  14 April 1978 14:17
;    48          0048  1 !
;    49          0049  1 ! Modified by:
;    50          0050  1 !
;    51          0051  1 !       V03-011 TSK0010         Tamar Krichevsky          8-May-1984
;    52          0052  1 !               Rearrange the calls to CLI$GET_VALUE and LIB$FIND_FILE, for
;    53          0053  1 !               input filename processing.  This will fix the problem of
;    54          0054  1 !               COPY a.a,a.a,a.a,a.a NL: copying every other file, instead of
;    55          0055  1 !               every file.
;    56          0056  1 !
;    57          0057  1 !       V03-010 TSK0009         Tamar Krichevsky          20-Apr-1984
```

```
;    58          0058  1 ;
;    59          0059  1 ;                Before the input file is opened, clear the longest record
;    60          0060  1 ;                length field in the input file's file header XAB.  This will
;    61          0061  1 ;                insure that the LRL value will be correct for record oriented
;    62          0062  1 ;                devices.  RMS does not clear this field if it is inappropriate.
;    63          0063  1 ;                As a result, the LRL could be carried from one file to another.
;    64          0064  1 ;                For example, given the command -- COPY foo.txt,SYS$INPUT a.a --
;    65          0065  1 ;                SYS$INPUT inherited the LRL form foo.txt.  (Not kosher!)
;    66          0066  1 ;
;    67          0067  1 ;        V03-009 TSK0008         Tamar Krichevsky        28-Mar-1984
;    68          0068  1 ;                Fix IF statement in COPY$OPN_OUTFIL which sets up the
;    69          0069  1 ;                default name string as ";*".  It was broken by TSK007.
;    70          0070  1 ;
;    71          0071  1 ;        V03-008 TSK0007         Tamar Krichevsky        2-Mar-1984
;    72          0072  1 ;                Convert input file parsing and searching to LIB$FIND_FILE.
;    73          0073  1 ;                Place the check for WILD_OUTPUT before the potential reparse
;    74          0074  1 ;                of the output file.  RMS changed how it set the bits in the
;    75          0075  1 ;                NAM$L_FNB field.
;    76          0076  1 ;
;    77          0077  1 ;        V03-007 TSK0006         Tamar Krichevsky        16-Feb-1984
;    78          0078  1 ;                Copy the input and output file names form the command line
;    79          0079  1 ;                into the appropriate buffers.  They were getting lost and
;    80          0080  1 ;                some error messages were being displayed like so:
;    81          0081  1 ;                "Error opening  as input"
;    82          0082  1 ;
;    83          0083  1 ;                Also add in check to see if the input file's record format
;    84          0084  1 ;                is VFC and the fixed control region size is zero.  The SOS
;    85          0085  1 ;                editor created files like this.  It knew that the smallest
;    86          0086  1 ;                fixed header size was two bytes; so it assumed 2 when it
;    87          0087  1 ;                saw 0.  RMS compenstated for this by setting the size to
;    88          0088  1 ;                two bytes.  Unfortunately, the incompatible attributes
;    89          0089  1 ;                comparison would fail because the input file's HSZ field in
;    90          0090  1 ;                the XABFHC was zero, but the output file's HSZ was two.
;    91          0091  1 ;                When COPY encounters such an input file, it will change the
;    92          0092  1 ;                HSZ field to two.
;    93          0093  1 ;
;    94          0094  1 ;        V03-006 TSK0005         Tamar Krichevsky        3-Oct-1983
;    95          0095  1 ;                Move the $DISPLAY, which was added in V03-005, to after the
;    96          0096  1 ;                the check for a successful file $CREATE or $OPEN.  Otherwise,
;    97          0097  1 ;                an extra message is issued when the file can not be accessed
;    98          0098  1 ;                for the $DISPLAY.
;    99          0099  1 ;
;   100          0100  1 ;        V03-005 LMP0150         L. Mark Pilant,         9-Sep-1983  11:19
;   101          0101  1 ;                Add a $DISPLAY to COPY$OPN_OUTFIL so that the protection
;   102          0102  1 ;                of the created file may be obtained.
;   103          0103  1 ;
;   104          0104  1 ;        V03-004 TSK0004         Tamar Krichevsky        8-Aug-1983
;   105          0105  1 ;                Fix ACCVIO during append operations.  Output file's XABPRO
;   106          0106  1 ;                should not be removed from XAB chain until file is closed.
;   107          0107  1 ;
;   108          0108  1 ;        V03-003 TSK0004         Tamar Krichevsky        8-Aug-1983
;   109          0109  1 ;                Modify COPY$OPN_OUTFILE, SETUP_OUTXAB and APPLY_OUT_QUAL so
;   110          0110  1 ;                that file protection and revision inforamtion is not propogated
;   111          0111  1 ;                to the output file from the input file.  Fix bug which clears
;   112          0112  1 ;                the expiration date when the output device is mag-tape. Fix
;   113          0113  1 ;                bug in /PROTECTION qualifier so that unspecified fields are
;   114          0114  1 ;                left alone.
```

```
;  115          0115  1 ;        V03-002 TSK0003         Tamar Krichevsky        4-Feb-1982
;  116          0116  1 ;                Change over to the new CLI.  Move external declarations from
;  117          0117  1 ;                COPY.REQ into this module.
;  118          0118  1 ;
;  119          0119  1 ;        V03-001 TSK0002         Tamar Krichevsky        4-Feb-1982
;  120          0120  1 ;                Copy the buckets size from the input FAB in the output XAB to
;  121          0121  1 ;                insure that the file is created with the correct bucket size.
;  122          0122  1 ;                When a file is created, if there are any allocation XABs, the
;  123          0123  1 ;                bucket size in the FAB is ignored.  Therefore, if the input file
;  124          0124  1 ;                has several areas, and area 0 does not have largest BKZ, something
;  125          0125  1 ;                other than the BKZ in the first (and only, in COPY's case) XABALL
;  126          0126  1 ;                must be used.  The largest bucket size is kept in the input file's
;  127          0127  1 ;                FAB. ****************** NOTE:  This works only if the ISAM files (the
;  128          0128  1 ;                worst offenders) are copied block mode.  IF FOR ANY REASON ISAM FILES
;  129          0129  1 ;                ARE COPIED USING RECORD MODE IN THE FUTURE, THIS PROCEDURE WILL HAVE TO
;  130          0130  1 ;                BE CHANGED.
;  131          0131  1 ;
;  132          0132  1 ;        X00025  TSK0001         Tamar Krichevsky        5-Feb-1982
;  133          0133  1 ;                Have Global Buffer Count (GBC) transferred from input FAB to
;  134          0134  1 ;                outout FAB.
;  135          0135  1 ;
;  136          0136  1 ;        X00024  KRM0038         Karl Malik      12-Jan-1982
;  137          0137  1 ;                Warn the user (in COPY$OPN_OUTFIL) if the output file
;  138          0138  1 ;                was forced to stream format ( in a network copy to
;  139          0139  1 ;                a 10,20 or RT system ).
;  140          0140  1 ;
;  141          0141  1 ;        X00023  KRM0035         Karl Malik      31-Dec-1981
;  142          0142  1 ;                Check for network quoted string in single output filespec
;  143          0143  1 ;                & if found, do not force multiple output files.
;  144          0144  1 ;
;  145          0145  1 ;        X00022  WMC0030         Wayne Cardoza   15-Dec-1981
;  146          0146  1 ;                Disallow output directory wildcards remaining after the output
;  147          0147  1 ;                file parse with the related input file.
;  148          0148  1 ;
;  149          0149  1 ;        X00021  WMC0021         Wayne Cardoza   8-Dec-1981
;  150          0150  1 ;                Set no_output_spec if only directory is wild and no explicit
;  151          0151  1 ;                filename components.
;  152          0152  1 ;
;  153          0153  1 ;        X00020  KFH0001         Ken Henderson   28-Sep-1981
;  154          0154  1 ;                Expiration and Backup dates are not copied from input file,
;  155          0155  1 ;                but instead are defaulted.
;  156          0156  1 ;
;  157          0157  1 ;        X00019  WMC0001         Wayne Cardoza   22-Jul-1981
;  158          0158  1 ;                Explicit protection specification should not cause old dates
;  159          0159  1 ;                to be preserved if a file spec is also present.
;  160          0160  1 ;
;  161          0161  1 ;        X00018  SPF0001         S. Forgey       27-Jan-1981
;  162          0162  1 ;                Allow wildcard directories in output file specifications to
;  163          0163  1 ;                go along with RMS now handling "sticky" directories.
;  164          0164  1 ;
;  165          0165  1 ;        X00017  JAK0017         J. Krycka       18-Sep-1980
;  166          0166  1 ;                Alter the X00006 special check for network access in setting up
;  167          0167  1 ;                the output Allocation XAB (i.e., gat ALQ and DEQ values from the
;  168          0168  1 ;                FHC XAB).
;  169          0169  1 ;
;  170          0170  1 ;        X00016  TMH0015         Tim Halvorsen   24-Mar-1980
;  171          0171  1 ;                Force creation of a new file (creation date, owner, prot)
```

```
;  172     0172  1 !               if the output file specification is explicit to maintain
;  173     0173  1 !               compatibility with release 1 behavior.  This involves changing
;  174     0174  1 !               the previous update to remove remove xabpro,rdt,dat if
;  175     0175  1 !               explicit output filespec as long as /PROT was not specified
;  176     0176  1 !               (If /PROT specified, xabpro must not be removed to allow it
;  177     0177  1 !               to work).
;  178     0178  1 !
;  179     0179  1 !       X00015  TMH0014         Tim Halvorsen   19-Mar-1980
;  180     0180  1 !               Do not remove output XABPRO,RDT,DAT blocks if concat follows
;  181     0181  1 !               flag is set because we were only trying to prevent changing
;  182     0182  1 !               characteristics on existing files -- concatenation always
;  183     0183  1 !               produces a new file.  Also, inhibit wildcard directories on
;  184     0184  1 !               output file specifications.
;  185     0185  1 !
;  186     0186  1 !       X00014  TMH0013         Tim Halvorsen   17-Mar-1980
;  187     0187  1 !               Issue ENDPRM2 call at the same time as ENDPRM1 call
;  188     0188  1 !               to eliminate problems with parameter ordering (in MCR,
;  189     0189  1 !               the parameters appear in reverse order).
;  190     0190  1 !
;  191     0191  1 !       X00013  JAK0003         J. Krycka       14-Jan-1980
;  192     0192  1 !               Undo X00005 change so that COPY will be able to use block I/O
;  193     0193  1 !               to copy relative and indexed files over the network.
;  194     0194  1 !
;  195     0195  1 !       X00012  TMH0012         T. Halvorsen    29-Dec-1979
;  196     0196  1 !               Remove XABPRO on appends since changing both owner or
;  197     0197  1 !               protection is prohibited (see X00010)
;  198     0198  1 !
;  199     0199  1 !       X00011  TMH0011         T. Halvorsen    15-Nov-1979
;  200     0200  1 !               Call CLI back with ENDPRM2 after output filespec is
;  201     0201  1 !               obtained to signal any unprocessed qualifiers.
;  202     0202  1 !
;  203     0203  1 !       X00010  TMH0010         T. Halvorsen    13-Nov-1979
;  204     0204  1 !               Zero the owner UIC field of the XABPRO on appends since
;  205     0205  1 !               changing the owner UIC for an existing file is prohibited.
;  206     0206  1 !
;  207     0207  1 !       X00009  TMH0009         T. Halvorsen    24-Oct-1979
;  208     0208  1 !               Test for output spec of only an explicit nodename
;  209     0209  1 !               so that the filename is defaulted correctly.
;  210     0210  1 !               Fix relative volume placement control to be hard (issue an
;  211     0211  1 !               error if the file cannot completely be placed on the volume).
;  212     0212  1 !
;  213     0213  1 !       X00008  T. Halvorsen    25-Jul-1979
;  214     0214  1 !               Add relative volume placement control.
;  215     0215  1 !               Fix message to indicate contiguous-best-try is being tried
;  216     0216  1 !               when there is not enough contigous space rather than issuing
;  217     0217  1 !               an error message.
;  218     0218  1 !
;  219     0219  1 !       X00007  T. Halvorsen    14-Jul-1979
;  220     0220  1 !               Fix problem copying ISAM files after another file (BIO
;  221     0221  1 !               was left on from previous file).
;  222     0222  1 !
;  223     0223  1 !       X00006  JAK0002         J. Krycka       16-Mar-1978     14:00
;  224     0224  1 !               To support copy of files over the network, get ALQ and DEQ
;  225     0225  1 !               values from input XABALL if NET bit is set.
;  226     0226  1 !
;  227     0227  1 !       X00005  JAK0001         J. Krycka       16-Mar-1978     14:00
;  228     0228  1 !               To support copy of relative files over the network, set
```

```
;  229          0229  1 !                BRO bit in output FAB if NET bit is set.
;  230          0230  1 !
;  231          0231  1 !        X00004  CHP20339        C. Peters       25-Oct-1978     14:10
;  232          0232  1 !                In COPY$GET_INFILE, zero ESL and RSL fields to avoid
;  233          0233  1 !                reporting wrong file specification on error.
;  234          0234  1 !
;  235          0235  1 !        X00003  CHP19547        C. Peters       7-Oct-1978      14:27
;  236          0236  1 !                Don't make version numbers sticky in an APPEND command.
;  237          0237  1 !
;  238          0238  1 !--
```

```
;  240          0239  1 !
;  241          0240  1 ! Table of Contents
;  242          0241  1 !
;  243          0242  1 FORWARD ROUTINE
;  244          0243  1     copy$get_infile,                                ! Obtains the input file specification
;  245          0244  1     copy$opn_infile,                                ! Opens the current input file
;  246          0245  1     copy$get_outfil,                                ! Obtains the output file specification
;  247          0246  1     copy$opn_outfil,                                ! Opens the current output file
;  248          0247  1     setup_extend,                                   ! Sets up an output file to be extended.
;  249          0248  1     setup_outxab        : NOVALUE,                  ! Sets up XAB fields for an output file.
;  250          0249  1     apply_out_qual      : NOVALUE;                  ! Sets output fields depending on file qualifiers.
;  251          0250  1
;  252          0251  1 !
;  253          0252  1 ! Include files
;  254          0253  1 !
;  255          0254  1
;  256          0255  1 LIBRARY 'SYS$LIBRARY:STARLET.L32';                  ! VAX/VMS system definitions
;  257          0256  1 REQUIRE 'SRC$:COPYMSG.REQ';                         ! Definition of macros to SIGNAL a message
;  258          0337  1
;  259          0338  1 !
;  260          0339  1 ! Macros
;  261          0340  1 !
;  262          0341  1 MACRO
;  263          0342  1
;  264          0343  1     ! Check to see if the global or local qualifier flag is set without the
;  265          0344  1     ! local negation flag being set.
;  266          0345  1     !
;  267        M 0346  1     qualifier_active( global_qual, local_qual, locally_negated ) =
;  268        M 0347  1         (IF (.global_qual AND NOT .locally_negated) OR .local_qual
;  269        M 0348  1          THEN true
;  270          0349  1          ELSE false )%
;  271          0350  1     ;
;  272          0351  1
;  273          0352  1 !
;  274          0353  1 ! External variables
;  275          0354  1 !
;  276          0355  1 EXTERNAL
;  277          0356  1     copy$cli_status : $BBLOCK,
;  278          0357  1     copy$sem_status : $BBLOCK,
;  279          0358  1
;  280          0359  1     curr_allocation_value,
;  281          0360  1     curr_extension_value,
;  282          0361  1     curr_protection_or,
;  283          0362  1     curr_protection_and,
;  284          0363  1     curr_file_max_value,
;  285          0364  1     curr_volume_value,
;  286          0365  1
;  287          0366  1     infile_cli_desc     : $BBLOCK[],                ! Descriptor for input file name returned by CLI
;  288          0367  1     in_name_desc        : VECTOR,                   ! Descriptor of input file specification
;  289          0368  1     out_name_desc       : VECTOR                    ! descriptor for output file specification
;  290          0369  1     ;
;  291          0370  1
;  292          0371  1 REQUIRE
;  293          0372  1     'SRC$:COPY.REQ'                                 ! Field definitions for  COPY$CLI_STATUS and COPY$SE
;  294          0373  1     ;
```

```
; %PRINT:       File: VMSMAC.B32, Version V04-000, Edit 1, WWC, 09-JAN-1978
```

```
;  295          0828  1
;  296          0829  1 EXTERNAL ROUTINE
;  297          0830  1     cli$get_value : addressing_mode( general ),
;  298          0831  1     copy$get_global_qual,                       ! Retrieves command level qualifiers
;  299          0832  1     copy$get_local_qual,                        ! Retrieves local qualifiers
;  300          0833  1     copy$check_file_for_match,                  ! See if input file matches command line criteria
;  301          0834  1     copy$calc_alq,                              ! Calculates a file extension quantity.
;  302          0835  1     copy$close_outf,                            ! Closes an output file
;  303          0836  1     copy$inopn_err,                             ! Handles an input $OPEN error
;  304          0837  1     copy$log_msg,                               ! Logs a message about COPY's activities
;  305          0838  1     copy$oclose_err,                            ! Handles an output file close error.
;  306          0839  1     copy$outopn_err,                            ! Handles an output $OPEN error
;  307          0840  1     copy$find_input_file,                       ! Finds and parses an input file specification
;  308          0841  1     copy$semantics;                             ! Determines semantics of a command
```

```
;  310          0842  1 GLOBAL ROUTINE copy$get_infile (input_fab, input_nam, input_xaball) =
;  311          0843  1                                                    ! Obtain input file specification
;  312          0844  1
;  313          0845  1 !++
;  314          0846  1 ! Functional description:
;  315          0847  1 !
;  316          0848  1 !       This routine gets an input file specification and all
;  317          0849  1 !       related qualifiers from the Command Language Interpreter. Then
;  318          0850  1 !       the file specification is parsed.
;  319          0851  1 !
;  320          0852  1 !       If a wildcard specification is still being processed, or if
;  321          0853  1 !       no more input specifications are available, this routine just
;  322          0854  1 !       returns successfully.
;  323          0855  1 !
;  324          0856  1 !       A series of flags are set if certain conditions obtain. These
;  325          0857  1 !       conditions describe the current list of files that are candidates
;  326          0858  1 !       for concatenation. The flags are set if the file specification
;  327          0859  1 !       contains input wildcards, an explicit wildcard version number, or an explicit version number.
;  328          0860  1 !
;  329          0861  1 !       Another flag applies only to this specification and says whether it contains any wildcards.
;  330          0862  1 !
;  331          0863  1 ! Calling sequence:
;  332          0864  1 !
;  333          0865  1 !       copy$get_infile (input_fab.ra.v, input_nam.ra.v, input_xaball.ra.v)
;  334          0866  1 !
;  335          0867  1 ! Input parameters:
;  336          0868  1 !
;  337          0869  1 !       input_fab       - the FAB to use for this input specification
;  338          0870  1 !       input_nam       - the NAM to use for this input specification
;  339          0871  1 !       input_xaball    - the XABALL to use for this input specification
;  340          0872  1 !
;  341          0873  1 ! Implicit inputs:
;  342          0874  1 !
;  343          0875  1 !       wildcard_active - a bit in COPY$CLI_STATUS that says that we are
;  344          0876  1 !                         already processing an input wildcard.
;  345          0877  1 !
;  346          0878  1 ! Output parameters:
;  347          0879  1 !
;  348          0880  1 !       none
;  349          0881  1 !
;  350          0882  1 ! Implicit outputs:
;  351          0883  1 !
;  352          0884  1 !       The fields of the FAB and the NAM block are filled in according
;  353          0885  1 !       to the CLI call and the $PARSE function call.
;  354          0886  1 !
;  355          0887  1 !       The RSL field of the dummy_nam_blk is filled in by the routine COPY$FIND_INPUT_FILE. This is later
;  356          0888  1 !       used in parsing the name additional input files or output files.
;  357          0889  1 !
;  358          0890  1 !       A bit in COPY$CLI_STATUS may be set:
;  359          0891  1 !
;  360          0892  1 !               multiple_input  - more than one input file specification in the command
;  361          0893  1 !               wildcard_active - if a wildcard is present
;  362          0894  1 !
;  363          0895  1 !       Some bits in COPY$SEM_STATUS may be set:
;  364          0896  1 !
;  365          0897  1 !               wild_input      - wildcard fields exist
;  366          0898  1 !               wild_inp_ver    - a wildcard version number exists
```

```
;  367          0899  1 !                exp_inp_ver     - an explicit version number exists
;  368          0900  1 !
;  369          0901  1 ! Routine value:
;  370          0902  1 !
;  371          0903  1 !       OK                  - success
;  372          0904  1 !       NO_MORE_FILES       - success, no more input specifications
;  373          0905  1 !       NO_FILE             - failure
;  374          0906  1 !
;  375          0907  1 ! Side effects:
;  376          0908  1 !
;  377          0909  1 !       none
;  378          0910  1 !
;  379          0911  1 !--
;  380          0912  1
;  381          0913  2     BEGIN
;  382          0914  2
;  383          0915  2     LOCAL
;  384          0916  2         rtn_status;                                     ! Retrun status from external calls
;  385          0917  2
;  386          0918  2     MAP
;  387          0919  2         input_fab       : REF BLOCK [, BYTE],           ! FAB to use with input file
;  388          0920  2         input_nam       : REF BLOCK [, BYTE],           ! NAM to use with input file
;  389          0921  2         input_xaball    : REF BLOCK [, BYTE];           ! XABALL to use with input file
;  390          0922  2
;  391          0923  2
;  392          0924  2
;  393          0925  2     ! Return if a wildcard file specification is currently being processed or the
;  394          0926  2     ! last input file name has been retrieved from the command line.  Otherwise,
;  395          0927  2     ! set the flag which indicates that more input files have been found.
;  396          0928  2     !
;  397          0929  2
;  398          0930  2     IF .wildcard_active                                 ! If a wildcard specification is currently
;  399          0931  2     THEN                                                ! being processed, then just return to caller.
;  400          0932  2         RETURN ok;
;  401          0933  2
;  402          0934  2     !
;  403          0935  2     ! Reinitialize the RSL and ESL fields of the NAM block so that a parsing
;  404          0936  2     ! error does not report an error in the previous file processed.
;  405          0937  2     !
;  406          0938  2
;  407          0939  2     input_nam [nam$b_esl] = 0;                          ! Expanded string length of zero.
;  408          0940  2     input_nam [nam$b_rsl] = 0;                          ! Resultant string length of zero.
;  409          0941  2
;  410          0942  2
;  411          0943  2     !
;  412          0944  2     ! Call LIB$FIND_FILE to parse the input file specification. This resolves
;  413          0945  2     ! logical names and determines if there are wildcards present, or explicit
;  414          0946  2     ! named fields present.
;  415          0947  2     !
;  416          0948  2
;  417          0949  3     IF NOT (rtn_status = copy$find_input_file ( infile_cli_desc ))
;  418          0950  2     THEN
;  419          0951  2         IF .rtn_status NEQ RMS$_NMF
;  420          0952  2         THEN
;  421          0953  2             RETURN .rtn_status;
;  422          0954  2
;  423          0955  2     !
```

```
;  424          0956  2     ! Initialize the input file FAB.
;  425          0957  2     !
;  426          0958  2
;  427        P 0959  2     $FAB_INIT (                                         ! Setup the input file FAB as follows:
;  428        P 0960  2                 FAB = .input_fab,                     !    FAB address is the input parameter
;  429        P 0961  2                 FAC = <GET,BRO>,                      !         !    Input file, mixed block and record acce
;  430        P 0962  2                 SHR = GET,                            !    Allow others to read the input file
;  431        P 0963  2                 DNA = 0,                              !    No default file specification
;  432        P 0964  2                 RTV = 0,                              !    Use default retrieval window size
;  433        P 0965  2                 RAT = CR,                             !    Carriage control in case unit record input
;  434        P 0966  2                 FOP = <SQO,NAM>,                      !    Sequential I/O only, open by name block
;  435        P 0967  2                 NAM = .input_nam,                     !    NAM block address
;  436          0968  2                 XAB = .input_xaball);                 !    XABALL block address.
;  437          0969  2
;  438          0970  2 !
;  439          0971  2 ! If there were no more files for the current inout specification, get the next
;  440          0972  2 ! one from the command line.
;  441          0973  2 !
;  442          0974  2     IF .rtn_status EQL RMS$_NMF
;  443          0975  2     THEN
;  444          0976  3         BEGIN
;  445          0977  3
;  446          0978  4         IF NOT (rtn_status = CLI$GET_VALUE( $DESCRIPTOR('infile'), infile_cli_desc))
;  447          0979  3         THEN
;  448          0980  3             RETURN no_more_files;
;  449          0981  3
;  450          0982  3         ! Get the qualifiers for this input file.
;  451          0983  3         !
;  452          0984  3         COPY$GET_LOCAL_QUAL();
;  453          0985  3
;  454          0986  3         ! Check to see if more than one input file has been given.
;  455          0987  3         !
;  456          0988  3         IF .rtn_status NEQ SS$_NORMAL
;  457          0989  3         THEN
;  458          0990  3             multiple_input - TRUE;
;  459          0991  3
;  460          0992  3         !
;  461          0993  3         ! Reinitialize the RSL and ESL fields of the NAM block so that a parsing
;  462          0994  3         ! error does not report an error in the previous file processed.
;  463          0995  3         !
;  464          0996  3
;  465          0997  3         input_nam [nam$b_esl] = 0;                          ! Expanded string length of zero.
;  466          0998  3         input_nam [nam$b_rsl] = 0;                          ! Resultant string length of zero.
;  467          0999  3
;  468          1000  3
;  469          1001  3         !
;  470          1002  3         ! Call LIB$FIND_FILE to parse the input file specification. This resolves
;  471          1003  3         ! logical names and determines if there are wildcards present, or explicit
;  472          1004  3         ! named fields present.
;  473          1005  3         !
;  474          1006  3
;  475          1007  4         IF NOT (rtn_sta'      .opy$find_input_file ( infile_cli_desc ))
;  476          1008  3         THEN
;  477          1009  3             RETURN .rtn_     us;
;  478          1010  2         END;
;  479          1011  2
;  480          1012  2 !
```

```
;  481          1013  2 ! Now test the type of expanded name string that we have. Does it contain wildcards? Were
;  482          1014  2 ! certain fields explicitly named?
;  483          1015  2 !
;  484          1016  2
;  485          1017  2     IF .input_nam [nam$v_wildcard]                    ! If there were any wildcards,
;  486          1018  2     THEN
;  487          1019  3         BEGIN
;  488          1020  3         wildcard_active = TRUE;                       !    set WILDCARD_ACTIVE. This says current file
;  489          1021  3                                                       !    specification contains wildcards.
;  490          1022  3         wild_input = TRUE;                            !    Also set WILD_INPUT. This says that the current
;  491          1023  3                                                       !    input list contains wildcard specs somewhere.
;  492          1024  3         first_wild_infile = TRUE;                     ! Indicate this is the first wild input file
;  493          1025  3         END
;  494          1026  2     ELSE                                              ! If no input wildcards in this spec, turn off
;  495          1027  2         wildcard_active = FALSE;                      !    the WILDCARD_ACTIVE flag.
;  496          1028  2
;  497          1029  2     IF .input_nam [nam$v_wild_ver]                    ! If an explicit wildcard version number
;  498          1030  2     THEN                                              !    was specified,
;  499          1031  2         wild_inp_ver = TRUE                           !    set the WILD_INP_VER flag.
;  500          1032  2     ELSE                                              ! Otherwise,
;  501          1033  3         BEGIN
;  502          1034  3         IF .input_nam [nam$v_exp_ver]                 !    see if an explicit version number was specified
;  503          1035  3         THEN                                          !    If it is, set the EXP_INP_VER flag, meaning
;  504          1036  3             exp_inp_ver = TRUE;                       !    that there is an explicit input version number.
;  505          1037  2         END;
;  506          1038  2
;  507          1039  2 !
;  508          1040  2 ! Return with success.
;  509          1041  2 !
;  510          1042  2
;  511          1043  2     RETURN ok;
;  512          1044  1     END;


                                                    .TITLE  COPYSPECS
                                                    .IDENT  \V04-000\

                                                    .PSECT  $PLIT$,NOWRT,NOEXE,2

                                65  6C  69  66  6E  69  00000 P.AAB:  .ASCII  \infile\                                ;
                                                     00006          .BLKB   2
                                            00000006 00008 P.AAA:   .LONG   6                                       ;
                                            00000000' 0000C         .ADDRESS P.AAB                                  ;

                                                    .EXTRN  COPY$MSG_NUMBER
                                                    .EXTRN  COPY$CLI_STATUS
                                                    .EXTRN  COPY$SEM_STATUS
                                                    .EXTRN  CURR_ALLOCATION_VALUE
                                                    .EXTRN  CURR_EXTENSION_VALUE
                                                    .EXTRN  CURR_PROTECTION_OR
                                                    .EXTRN  CURR_PROTECTION_AND
                                                    .EXTRN  CURR_FILE_MAX_VALUE
                                                    .EXTRN  CURR_VOLUME_VALUE
                                                    .EXTRN  INFILE_CLI_DESC
                                                    .EXTRN  IN_NAME_DESC, OUT_NAME_DESC
                                                    .EXTRN  CLI$_PRESENT, CLI$_NEGATED
                                                    .EXTRN  CLI$_LOCPRES, CLI$_LOCNEG
```

```
                                                        .EXTRN  CLI$GET_VALUE, COPY$GET_GLOBAL_QUAL
                                                        .EXTRN  COPY$GET_LOCAL_QUAL
                                                        .EXTRN  COPY$CHECK_FILE_FOR_MATCH
                                                        .EXTRN  COPY$CALC_ALQ, COPY$CLOSE_OUTF
                                                        .EXTRN  COPY$INOPN_ERR, COPY$LOG_MSG
                                                        .EXTRN  COPY$OCLOSE_ERR
                                                        .EXTRN  COPY$OUTOPN_ERR
                                                        .EXTRN  COPY$FIND_INPUT_FILE
                                                        .EXTRN  COPY$SEMANTICS

                                                        .PSECT  $CODE$,NOWRT,2

                                        07FC 00000      .ENTRY  COPY$GET_INFILE, Save R2,R3,R4,R5,R6,R7,R8,-; 0842
                                                                R9,R10
                                5A      0000G   CF  9E  00002      MOVAB   INFILE_CLI_DESC, R10
                                59      0000G   CF  9E  00007      MOVAB   COPY$SEM_STATUS, R9
                    03      02  A9              05  E1  0000C      BBC     #5, COPY$SEM_STATUS+2, 1$       ; 0930
                                            00B5  31  00011      BRW     10$
                                57        08    AC  D0  00014 1$:  MOVL    INPUT_NAM, R7                   ; 0939
                                          0B    A7  94  00018      CLRB    11(R7)
                                          03    A7  94  0001B      CLRB    3(R7)                           ; 0940
                                                5A  DD  0001E      PUSHL   R10                             ; 0949
                            0000G   CF          01  FB  00020      CALLS   #1, COPY$FIND_INPUT_FILE
                                58              50  D0  00025      MOVL    R0, RTN_STATUS
                                09              58  E8  00028      BLBS    RTN_STATUS, 2$
                        000182CA    8F          58  D1  0002B      CMPL    RTN_STATUS, #99018              ; 0951
                                                6F  12  00032      BNEQ    5$
                                56        04    AC  D0  00034 2$:  MOVL    INPUT_FAB, R6                   ; 0968
0050    8F              00              6E          00  2C  00038      MOVC5   #0, (SP), #0, #80, (R6)
                                                    66      0003F
                                66      5003    8F  B0  00040      MOVW    #20483, (R6)
                                  04    A6 01000040 8F  D0  00045      MOVL    #16777280, 4(R6)
                                  16    A6    0242  8F  B0  0004D      MOVW    #578, 22(R6)
                                  1E    A6    0202  8F  B0  00053      MOVW    #514, 30(R6)
                                  24    A6      0C  AC  D0  00059      MOVL    INPUT_XABALL, 36(R6)
                                  28    A6          57  D0  0005E      MOVL    R7, 40(R6)
                        000182CA    8F              58  D1  00062      CMPL    RTN_STATUS, #99018              ; 0974
                                                    3C  12  00069      BNEQ    6$
                                                    5A  DD  0006B      PUSHL   R10                             ; 0978
                                          0000'     CF  9F  0006D      PUSHAB  P.AAA
                        00000000G       00          02  FB  00071      CALLS   #2, CLI$GET_VALUE
                                58              50  D0  00078      MOVL    R0, RTN_STATUS
                                04              58  E8  0007B      BLBS    RTN_STATUS, 3$
                                50              03  D0  0007E      MOVL    #3, R0                          ; 0980
                                                    04  00081      RET
                            0000G   CF          00  FB  00082 3$:  CALLS   #0, COPY$GET_LOCAL_QUAL         ; 0984
                                01              58  D1  00087      CMPL    RTN_STATUS, #1                  ; 0988
                                                    04  13  0008A      BEQL    4$
                                  01    A9          02  88  0008C      BISB2   #2, COPY$SEM_STATUS+1           ; 0990
                                          0B    A7  94  00090 4$:  CLRB    11(R7)                          ; 0997
                                          03    A7  94  00093      CLRB    3(R7)                           ; 0998
                                                5A  DD  00096      PUSHL   R10                             ; 1007
                            0000G   CF          01  FB  00098      CALLS   #1, COPY$FIND_INPUT_FILE
                                58              50  D0  0009D      MOVL    R0, RTN_STATUS
                                04              58  E8  000A0      BLBS    RTN_STATUS, 6$
                                50              58  D0  000A3 5$:  MOVL    RTN_STATUS, R0                  ; 1009
                                                    04  000A6      RET
```

```
                          09        35   A7  E9 000A7 6$:     BLBC    53(R7), 7$                              ; 1017
                          69 02200010    8F  C8 000AB         BISL2   #35651600, COPY$SEM_STATUS                ; 1024
                                         04  11 000B2         BRB     8$                                      ; 1017
                    02    A9           20  8A 000B4 7$:     BICB2   #32, COPY$SEM_STATUS+2                  ; 1027
              05    34    A7           03  E1 000B8 8$:     BBC     #3, 52(R7), 9$                          ; 1029
                          69           20  88 000BD         BISB2   #32, COPY$SEM_STATUS                    ; 1031
                                       07  11 000C0         BRB     10$                                     ;
                          03        34   A7  E9 000C2 9$:     BLBC    52(R7), 10$                             ; 1034
                          69           02  88 000C6         BISB2   #2, COPY$SEM_STATUS                     ; 1036
                          50           01  DC 000C9 10$:    MOVL    #1, R0                                  ; 1043
                                           04 000CC         RET                                             ; 1044

; Routine Size:  205 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  514          1045  1 GLOBAL ROUTINE copy$opn_infile (input_fab) =          ! Open the current input file
;  515          1046  1
;  516          1047  1 !++
;  517          1048  1 ! Functional description:
;  518          1049  1 !
;  519          1050  1 !       This routine opens the current input file. If the input file
;  520          1051  1 !       specification contains a wildcard field, an RMS $SEARCH for the
;  521          1052  1 !       next wildcard match occurs before the actual file open.
;  522          1053  1 !
;  523          1054  1 !       Any input parameter qualifiers are applied to the file's RMS blocks before
;  524          1055  1 !       the open is performed. For now, the only valid qualifier is /READ_CHECK.
;  525          1056  1 !
;  526          1057  1 !       If the OPEN fails, an error is reported to SYS$ERROR. When input wildcards are present,
;  527          1058  1 !       two types of failure are permitted:
;  528          1059  1 !
;  529          1060  1 !               RMS$_NMF        - no more files match given wildcard
;  530          1061  1 !               open failure    - allowed when a file matching a wildcard spec cannot be
;  531          1062  1 !                                 opened, as long as that file would have been copied without concatenation.
;  532          1063  1 !
;  533          1064  1 ! Calling sequence:
;  534          1065  1 !
;  535          1066  1 !       copy$opn_infile (input_fab.ra.v)
;  536          1067  1 !
;  537          1068  1 ! Input parameters:
;  538          1069  1 !
;  539          1070  1 !       input_fab       - the FAB associated with the input file
;  540          1071  1 !
;  541          1072  1 ! Implicit inputs:
;  542          1073  1 !
;  543          1074  1 !       COPY$CLI_STATUS bits are checked:
;  544          1075  1 !
;  545          1076  1 !               iread_check_bit - This bit is set if the /READ_CHECK qualifier was specified for this file.
;  546          1077  1 !               wildcard_active - This specification contains wildcards.
;  547          1078  1 !                                 Find the next file with a $SEARCH function call.
;  548          1079  1 !
;  549          1080  1 !       input file NAM block is read to obtain the length of the resultant name string
;  550          1081  1 !       input file XABFHC to check the HSZ for VFC files.
;  551          1082  1 !
;  552          1083  1 !       COPY$SEM_STATUS bits are checked:
;  553          1084  1 !
;  554          1085  1 !               multiple_output - Multiple files are being produced. This is checked to allow for
;  555          1086  1 !                                 open failure on a wildcard specified file.
;  556          1087  1 !
;  557          1088  1 ! Output parameters:
;  558          1089  1 !
;  559          1090  1 !       none
;  560          1091  1 !
;  561          1092  1 ! Implicit outputs:
;  562          1093  1 !
;  563          1094  1 !       in_name_desc    - the length field of the input name descriptor is written from the RSL
;  564          1095  1 !                         field in the NAM block
;  565          1096  1 !
;  566          1097  1 !       The FAB$V_RCK bit in the input FAB is set if /READ_CHECK was specified.
;  567          1098  1 !
;  568          1099  1 !       COPY$CLI_STATUS bit settings may be altered:
;  569          1100  1 !
;  570          1101  1 !               wildcard_active - turned off if no more files that match wildcard are found.
```

```
;  571          1102  1 !               infile_open     - set if the file is opened successfully
;  572          1103  1 !
;  573          1104  1 ! Routine value:
;  574          1105  1 !
;  575          1106  1 !       OK              - input file open
;  576          1107  1 !       NO_MORE_FILES   - no further wildcard match found
;  577          1108  1 !       NO_WILD_OPEN    - open failure on wildcard match file
;  578          1109  1 !       NO_FILE         - input file not found
;  579          1110  1 !
;  580          1111  1 ! Side effects:
;  581          1112  1 !
;  582          1113  1 !       The input file is opened.
;  583          1114  1 !       If an RMS SEARCH function fails, then an error is reported on SYS$ERROR.
;  584          1115  1 !
;  585          1116  1 !--
;  586          1117  1
;  587          1118  2     BEGIN
;  588          1119  2
;  589          1120  2     MAP
;  590          1121  2         input_fab       : REF BLOCK [, BYTE];           ! input FAB block
;  591          1122  2
;  592          1123  2     BIND
;  593          1124  2         input_xaball    =                               ! input file XABALL block
;  594          1125  2                 .input_fab [fab$l_xab]          : BLOCK [, BYTE],
;  595          1126  2         input_xabdat    =                               ! input file XABDAT block
;  596          1127  2                 .input_xaball [xab$l_nxt]       : BLOCK [, BYTE],
;  597          1128  2         input_xabfhc    =                               ! input file XABFHC block
;  598          1129  2                 .input_xabdat [xab$l_nxt]       : BLOCK [, BYTE],
;  599          1130  2         input_nam       =                               ! input NAM block address
;  600          1131  2                         .input_fab [fab$l_nam] : BLOCK [, BYTE];
;  601          1132  2
;  602          1133  2     LOCAL
;  603          1134  2         status;                                         ! RMS status code variable
;  604          1135  2
;  605          1136  2 !
;  606          1137  2 ! If a wildcard specification is active, call RMS to search for the next wildcard match.
;  607          1138  2 !
;  608          1139  2
;  609          1140  2     IF .wildcard_active                                 ! If an input wildcard field is present,
;  610          1141  2     THEN
;  611          1142  2         IF NOT .first_wild_infile
;  612          1143  2         THEN
;  613          1144  3             BEGIN
;  614          1145  3             status = COPY$FIND_INPUT_FILE( infile_cli_desc );
;  615          1146  3
;  616          1147  3             IF .status EQL rms$_nmf                     ! If no more wildcard matches exist,
;  617          1148  3             THEN
;  618          1149  4                 BEGIN
;  619          1150  4                 wildcard_active = FALSE;            !    turn off the WILDCARD_ACTIVE flag,
;  620          1151  4                 RETURN no_more_files;               !    and return with success status of NO_MORE_FILES
;  621          1152  3                 END;
;  622          1153  3
;  623          1154  3             IF NOT .status                              ! If RMS returned some other error code,
;  624          1155  3             THEN
;  625          1156  4                 BEGIN
;  626          1157  4                 copy$inopn_err (.input_fab);            !    then call the RMS error action routine.
;  627          1158  4                 wildcard_active = FALSE;                !    Turn off the wildcard flag so that we don't loo
```

```
  4                RETURN no_file;                          !    for the file again. Return to caller with NO_FI
  3                END;                                     !    error code.
  3            END                                          ! End of special wildcar: search processing.
  2        ELSE
  2            first_wild_infile = FALSE;
  2
  2 !
  2 ! If the user specified the input read checking qualifier, turn on the appropriate bit in the FAB.
  2 !
  2
  3     IF qualifier_active( read_chk_qual, loc_read_chk_qual, neg_read_chk_qual)
  2     THEN
  2         input_fab [fab$v_rck] = TRUE                    !    then turn on the FAB read check indicator.
  2     ELSE
  2         input_fab [fab$v_rck] = FALSE;                  !    Otherwise, turn it off.
  2
  2 !
  2 ! Open the input file.  First, zero the LRL field in the file header XAB.  This
  2 ! insures that it will have the appropriate value if the input device is record
  2 ! oriented (i.e. SYS$INPUT).
  2 !
  2
  2     input_xabfhc[ XAB$W_LRL ] = 0;
P 2     IF $RMS_OPEN (                                      ! Open the input file with RMS.
P 2                 FAB = .input_fab,                       !    Specify the input parameter for the FAB,
  3                 ERR = copy$inopn_err)                   !    and an error action routine.
  2     THEN                                                ! If the OPEN is successful,
  3         BEGIN
  3         infile_open = TRUE;                             !    indicate that the file is open
  3         in_name_desc [0] = .input_nam [nam$b_rsl];      !    and set the length of the input file name descr
  3
  3         ! If record format is VFC and the HSZ is 0, then set the HSZ to 2.
  3         ! If this isn't done, the incompatible attributes check will
  3         ! incorrectly fail.
  3         !
  3         IF .input_fab [FAB$B_RFM] EQL FAB$C_VFC
  3                 AND
  3            .input_xabfhc [XAB$B_HSZ] EQL 0
  3         THEN
  3            input_xabfhc [XAB$B_HSZ] = 2;
  3         RETURN ok;                                      ! Return to caller with success code.
  3         END                                             ! End of successful OPEN processing
  3
  2     ELSE
  3         BEGIN
  3
  3 !
  3 ! If multiple output files are being produced, and this is a file that matches a wildcard specification,
  3 ! allow the open to fail. This means that one file that matches the wildcard specification is not copied
  3 ! to a new output file.
  3 !
  3
  3         IF .wildcard_active AND
  4             (.multiple_output OR NOT .explicit_concat_qual )
  3         THEN
  3             RETURN no_wild_open
  3         ELSE
```

```
;  685         1216  3                 RETURN no_file;
;  686         1217  2             END;
;  687         1218  2
;  688         1219  1         END;


                                                              .EXTRN  SYS$OPEN

                                          007C 00000          .ENTRY  COPY$OPN_INFILE, Save R2,R3,R4,R5,R6             ; 1045
                   56      0000G  CF   9E 00002               MOVAB   COPY$CLI_STATUS+4, R6                            ;
                   55      0000G  CF   9E 00007               MOVAB   COPY$SEM_STATUS, R5                              ;
                   52         04  AC   D0 0000C               MOVL    INPUT_FAB, R2                                    ; 1125
                   50         24  A2   D0 00010               MOVL    36(R2), R0                                       ;
                   50         04  A0   D0 00014               MOVL    4(R0), R0                                        ; 1127
                   53         04  A0   D0 00018               MOVL    4(R0), R3                                        ; 1129
                   54         28  A2   D0 0001C               MOVL    40(R2), R4                                       ; 1131
  33          02   A5              05  E1 00020               BBC     #5, COPY$SEM_STATUS+2, 3$                        ; 1140
  2A          03   A5              01  E0 00025               BBS     #1, COPY$SEM_STATUS+3, 2$                        ; 1142
                           0000G   CF  9F 0002A               PUSHAB  INFILE_CLI_DESC                                  ; 1145
          0000G   CF               01  FB 0002E               CALLS   #1, COPY$FIND_INPUT_FILE                         ;
       000182CA   8F               50  D1 00033               CMPL    STATUS, #99018                                   ; 1147
                                   08  12 0003A               BNEQ    1$                                               ;
              02  A5               20  8A 0003C               BICB2   #32, COPY$SEM_STATUS+2                           ; 1150
                  50               03  D0 00040               MOVL    #3, R0                                           ; 1151
                                       04 00043               RET                                                      ;
                  11               50  E8 00044 1$:           BLBS    STATUS, 3$                                       ; 1154
                                   52  DD 00047               PUSHL   R2                                               ; 1157
           0000G  CF               01  FB 00049               CALLS   #1, COPY$INOPN_ERR                               ;
              02  A5               20  8A 0004E               BICB2   #32, COPY$SEM_STATUS+2                           ; 1158
                                   5D  11 00052               BRB     11$                                              ; 1159
              03  A5               02  8A 00054 2$:           BICB2   #2, COPY$SEM_STATUS+3                            ; 1163
                  04               66  E9 00058 3$:           BLBC    COPY$CLI_STATUS+4, 4$                            ; 1169
  04              66               02  E1 0005B               BBC     #2, COPY$CLI_STATUS+4, 5$                        ;
  07              66               01  E1 0005F 4$:           BBC     #1, COPY$CLI_STATUS+4, 6$                        ;
              06  A2          80   8F  88 00063 5$:           BISB2   #128, 6(R2)                                      ; 1171
                                   05  11 00068               BRB     7$                                               ;
              06  A2          80   8F  8A 0006A 6$:           BICB2   #128, 6(R2)                                      ; 1173
                              0A   A3  B4 0006F 7$:           CLRW    10(R3)                                           ; 1181
                           0000G   CF  9F 00072               PUSHAB  COPY$INOPN_ERR                                   ; 1184
                                   52  DD 00076               PUSHL   R2                                               ;
       00000000G  00               02  FB 00078               CALLS   #2, SYS$OPEN                                     ;
                  1D               50  E9 0007F               BLBC    R0, 9$                                           ;
              02  A5               04  88 00082               BISB2   #4, COPY$SEM_STATUS+2                            ; 1187
           0000G  CF          03   A4  9A 00086               MOVZBL  3(R4), IN_NAME_DESC                              ; 1188
              03              1F   A2  91 0008C               CMPB    31(R2), #3                                       ; 1194
                                   09  12 00090               BNEQ    8$                                               ;
                              17   A3  95 00092               TSTB    23(R3)                                           ; 1196
                                   04  12 00095               BNEQ    8$                                               ;
              17  A3               02  90 00097               MOVB    #2, 23(R3)                                       ; 1198
                  50               01  D0 0009B 8$:           MOVL    #1, R0                                           ; 1203
                                       04 0009E               RET                                                      ;
  0D          02  A5               05  E1 0009F 9$:           BBC     #5, COPY$SEM_STATUS+2, 11$                       ; 1211
                  05          01   A5  E8 000A4               BLBS    COPY$SEM_STATUS+1, 10$                           ; 1212
  04          FC  A6               02  E0 000A8               BBS     #2, COPY$CLI_STATUS, 11$                         ;
                  50               05  D0 000AD 10$:          MOVL    #5, R0                                           ; 1216
                                       04 000B0               RET                                                      ;
```

```
                                   50  D4 000B1 11$:    CLRL    R0                                              ; 1219
                                       04 000B3         RET                                                     ;

; Routine Size:  180 bytes,    Routine Base:  $CODE$ + 00CD
```

```
;  690          1220  1 GLOBAL ROUTINE copy$get_outfil (output_fab, output_nam, output_xabfhc) =
;  691          1221  1                                                        ! Obtain the output file specification
;  692          1222  1
;  693          1223  1 !++
;  694          1224  1 ! Functional description:
;  695          1225  1 !
;  696          1226  1 !       This routine obtains the output file specification and all
;  697          1227  1 !       related qualifiers from the Command Language Interpreter. Then
;  698          1228  1 !       the file specification is parsed without any help from related input file name
;  699          1229  1 !       blocks. This initial parse determines whether the file specification had null file
;  700          1230  1 !       name, type, and version number fields.
;  701          1231  1 !
;  702          1232  1 !       If no output file name, type, or version number is given, a flag
;  703          1233  1 !       is set in COPY$SEM_STATUS.
;  704          1234  1 !
;  705          1235  1 ! Calling sequence:
;  706          1236  1 !
;  707          1237  1 !       copy$get_outfil (output_fab.ra.v, output_nam.ra.v, output_xabfhc.ra.v)
;  708          1238  1 !
;  709          1239  1 ! Input parameters:
;  710          1240  1 !
;  711          1241  1 !       output_fab      - the FAB to use for this output specification
;  712          1242  1 !       output_nam      - the NAM to use for this output specification
;  713          1243  1 !       output_xabfhc   - the XABFHC to use for this output specification
;  714          1244  1 !
;  715          1245  1 ! Implicit inputs:
;  716          1246  1 !
;  717          1247  1 !       The RLF field of the output NAM block contains the address of the input file NAM block.
;  718          1248  1 !
;  719          1249  1 ! Output parameters:
;  720          1250  1 !
;  721          1251  1 !       none
;  722          1252  1 !
;  723          1253  1 ! Implicit outputs:
;  724          1254  1 !
;  725          1255  1 !       The fields of the FAB and the NAM block are filled in according
;  726          1256  1 !       to the CLI call, FAB initialization, and the $PARSE function call.
;  727          1257  1 !
;  728          1258  1 !       A bit may be set in COPY$SEM_STATUS:
;  729          1259  1 !
;  730          1260  1 !               no_output_spec  - no output name, type, or version number specified.
;  731          1261  1 !
;  732          1262  1 ! Routine value:
;  733          1263  1 !
;  734          1264  1 !       OK              - success
;  735          1265  1 !       NO_FILE         - the $PARSE function call returned an error code
;  736          1266  1 !
;  737          1267  1 ! Side effects:
;  738          1268  1 !
;  739          1269  1 !       An error is reported if the $PARSE function returns an error status code and
;  740          1270  1 !       COPY$OUTOPN_ERR is called.
;  741          1271  1 !
;  742          1272  1 !--
;  743          1273  1
;  744          1274  2     BEGIN
;  745          1275  2
;  746          1276  2     MAP
```

```
;  747          1277  2          output_fab      : REF BLOCK [, BYTE],           ! FAB to use with output file
;  748          1278  2          output_nam      : REF BLOCK [, BYTE],           ! NAM to use with output file
;  749          1279  2          output_xabfhc   : REF BLOCK [, BYTE];           ! XABFHC to use with output file
;  750          1280  2
;  751          1281  2      LOCAL
;  752          1282  2          cli_desc : $BBLOCK[ DSC$C_S_BLN ],              ! Descriptor for qualifier values
;  753          1283  2          temp_rlf;                                       ! Holds the output RLF field
;  754          1284  2
;  755          1285  2
;  756          1286  2
;  757          1287  2      ! Initialize descriptor.  Retrieve the output file specification.
;  758          1288  2      !
;  759          1289  2      CH$FILL( 0, DSC$C_S_BLN, cli_desc );
;  760          1290  2      cli_desc[ DSC$B_CLASS ] = DSC$K_CLASS_D;
;  761          1291  2
;  762          1292  2      CLI$GET_VALUE( $DESCRIPTOR('OUTFILE'), cli_desc);
;  763          1293  2
;  764          1294  2      ! Save the file name in the output name descriptor; in case the name
;  765          1295  2      ! doesn't parse.  The name given on the command line will be used
;  766          1296  2      ! in the error message returned to the user.
;  767          1297  2      !
;  768          1298  2      out_name_desc[0] = .cli_desc[DSC$W_LENGTH];
;  769          1299  2      CH$MOVE(.cli_desc[DSC$W_LENGTH], .cli_desc[DSC$A_POINTER], .out_name_desc[1]);
;  770          1300  2
;  771          1301  2      ! Get the qualifiers for the output file.
;  772          1302  2      !
;  773          1303  2      COPY$GET_GLOBAL_QUAL();
;  774          1304  2
;  775          1305  2 !
;  776          1306  2 ! Initialize the output file FAB.
;  777          1307  2 !
;  778          1308  2
;  779        P 1309  2      $FAB_INIT (                                         ! Setup the output file FAB as follows:
;  780        P 1310  2                  FAB = .output_fab,                      !    FAB address is the output parameter
;  781        P 1311  2                  FAC = <PUT,TRN>,                        !    Output file
;  782        P 1312  2                  SHR = NIL,                              !    No file sharing
;  783        P 1313  2                  FNA = .cli_desc [DSC$A_POINTER],        !    File name address from CLI
;  784        P 1314  2                  FNS = .cli_desc [DSC$W_LENGTH],         !    File name size from CLI also
;  785        P 1315  2                  RTV = 0,                                !    Use the system default retrieval window size
;  786        P 1316  2                  FOP = <SQO,OFP,NAM>,                    !    Sequential operations only, output file parse,
;  787        P 1317  2                  NAM = .output_nam,                      !    NAM block address
;  788          1318  2                  XAB = .output_xabfhc);                  !    XABFHC block address
;  789          1319  2                                                          !        name block open
;  790          1320  2
;  791          1321  2 !
;  792          1322  2 ! Zero the expanded string length so that the COPY error routine, copy$outopn_err, can
;  793          1323  2 ! decide if an expanded name string was created by RMS.
;  794          1324  2 !
;  795          1325  2
;  796          1326  2      output_nam [nam$b_esl] = 0;                         ! Zero the output expanded string length.
;  797          1327  2
;  798          1328  2 !
;  799          1329  2 ! Temporarily remove the RLF field of the output NAM block so that the
;  800          1330  2 ! output file specification can be tested for null name, type, and
;  801          1331  2 ! version number fields.
;  802          1332  2 !
;  803          1333  2
```

```
;  804          1334  2     temp_rlf = .output_nam [nam$l_rlf];                 ! Save the RLF field because it may be needed later.
;  805          1335  2     output_nam [nam$l_rlf] = 0;                         ! Set the RLF field to null.
;  806          1336  2
;  807          1337  2 !
;  808          1338  2 ! Parse the output file specification.
;  809          1339  2 !
;  810          1340  2
;  811        P 1341  2     IF NOT $RMS_PARSE (                                 ! Call the RMS function that parses file specificati
;  812        P 1342  2                         FAB = .output_fab,              !    specifying the output FAB parameter,
;  813          1343  3                         ERR = copy$outopn_err)          !    and an error routine.
;  814          1344  2     THEN                                                ! If the PARSE is not successful,
;  815          1345  2         RETURN no_file;                                 !    then return an error code to the caller.
;  816          1346  2
;  817          1347  2 !
;  818          1348  2 ! Test for an absence of the file name, type, and version number fields
;  819          1349  2 ! (or the presence of a network quoted string).
;  820          1350  2 !
;  821          1351  2
;  822          1352  2     IF (NOT .output_nam [nam$v_wild_name]) AND          ! If no output wildcards are present,
;  823          1353  2         (NOT .output_nam [nam$v_wild_type]) AND
;  824          1354  2         (NOT .output_nam [nam$v_wild_ver]) AND
;  825          1355  2         (NOT .output_nam [nam$v_quoted]) AND            !    and no quoted string
;  826          1356  2         (NOT .output_nam [nam$v_exp_name]) AND          !    and no output name,
;  827          1357  2         (NOT .output_nam [nam$v_exp_type]) AND          !    and no output type,
;  828          1358  2         (NOT .output_nam [nam$v_exp_ver]) AND           !    and no output version number,
;  829          1359  3         (.output_nam [nam$v_exp_dir] OR                 !    and an explicit directory
;  830          1360  3         .output_nam [nam$v_exp_dev] OR                  !    or device name
;  831          1361  3         .output_nam [nam$v_node])                       !    or node name is given,
;  832          1362  2     THEN
;  833          1363  2         no_output_spec = TRUE;                          !    then set NO_OUTPUT_SPEC bit.
;  834          1364  2
;  835          1365  2
;  836          1366  2     ! If the file name, file type or version fields are ALL either wild or no specified and
;  837          1367  2     ! the output file spec does not contain a quoted string, then set the flag which indicates
;  838          1368  2     ! that the output file spec was completely wild.
;  839          1369  2     !
;  840          1370  3     IF  (.output_nam [nam$v_wild_name] OR NOT .output_nam [nam$v_exp_name])
;  841          1371  2             AND
;  842          1372  3         (.output_nam [nam$v_wild_type] OR NOT .output_nam [nam$v_exp_type])
;  843          1373  2             AND
;  844          1374  3         (.output_nam [nam$v_exp_ver] OR NOT .output_nam [nam$v_wild_ver])
;  845          1375  2             AND
;  846          1376  2         NOT .output_nam [nam$v_quoted]
;  847          1377  2     THEN
;  848          1378  2         no_expl_out_fields = TRUE;
;  849          1379  2 !
;  850          1380  2 ! Reload the RLF field. Another PARSE will be performed later in the routine
;  851          1381  2 ! COPY$OPN_OUTFIL and may take fields from the input resultant file string.
;  852          1382  2 !
;  853          1383  2
;  854          1384  2     output_nam [nam$l_rlf] = .temp_rlf;
;  855          1385  2
;  856          1386  2 !
;  857          1387  2 ! Return with a success code.
;  858          1388  2 !
;  859          1389  2
;  860          1390  2     RETURN ok;                                          ! Return successfully.
```

```
;  861          1391  2
;  862          1392  1      END;


                                                               .PSECT   $PLIT$,NOWRT,NOEXE,2

                       45  4C  49  46  54  55  4F  00010 P.AAD:  .ASCII   \OUTFILE\                                ;
                                                   00017          .BLKB    1
                                         00000007  00018 P.AAC:  .LONG    7                                        ;
                                         00000000' 0001C          .ADDRESS P.AAD                                    ;

                                                               .EXTRN   SYS$PARSE

                                                               .PSECT   $CODE$,NOWRT,2

                                              007C 00000          .ENTRY   COPY$GET_OUTFIL, Save R2,R3,R4,R5,R6     ; 1220
                           5E              08  C2 00002          SUBL2    #8, SP                                   ;
       08          00      6E              00  2C 00005          MOVC5    #0, (SP), #0, #8, CLI_DESC               ; 1289
                                           6E     0000A                                                            ;
                     03    AE              02  90 0000B          MOVB     #2, CLI_DESC+3                           ; 1290
                                           5E  DD 0000F          PUSHL    SP                                       ; 1292
                                    0000'  CF  9F 00011          PUSHAB   P.AAC                                    ;
               00000000G   00              02  FB 00015          CALLS    #2, CLI$GET_VALUE                        ;
                   0000G   CF              6E  3C 0001C          MOVZWL   CLI_DESC, OUT_NAME_DESC                  ; 1298
           0000G  DF   04  BE              6E  28 00021          MOVC3    CLI_DESC, @CLI_DESC+4, @OUT_NAME_DESC+4  ; 1299
                   0000G   CF              00  FB 00028          CALLS    #0, COPY$GET_GLOBAL_QUAL                 ; 1303
                           56       04     AC  D0 0002D          MOVL     OUTPUT_FAB, R6                           ; 1318
  0050   8F        00      6E              00  2C 00031          MOVC5    #0, (SP)  #0, #80, (R6)                  ;
                                           66     00038                                                            ;
                           66     5003     8F  B0 00039          MOVW     #20483, (R6)                             ;
                     04    A6 21000040     8F  D0 0003E          MOVL     #553648192, 4(R6)                        ;
                     16    A6     2011     8F  B0 00046          MOVW     #8209, 22(R6)                            ;
                     1F    A6              02  90 0004C          MOVB     #2, 31(R6)                               ;
                     24    A6       0C     AC  D0 00050          MOVL     OUTPUT_XABFHC, 36(R6)                    ;
                           52       08     AC  D0 00055          MOVL     OUTPUT_NAM, R2                           ;
                     28    A6              52  D0 00059          MOVL     R2, 40(R6)                               ;
                     2C    A6       04     AE  D0 0005D          MOVL     CLI_DESC+4, 44(R6)                       ;
                     34    A6              6E  90 00062          MOVB     CLI_DESC, 52(R6)                         ;
                                    0B     A2  94 00066          CLRB     11(R2)                                   ; 1326
                           53       10     A2  D0 00069          MOVL     16(R2), TEMP_RLF                         ; 1334
                                    10     A2  D4 0006D          CLRL     16(R2)                                   ; 1335
                                 0000G     CF  9F 00070          PUSHAB   COPY$OUTOPN_ERR                          ; 1343
                                           56  DD 00074          PUSHL    R6                                       ;
               00000000G   00              02  FB 00076          CALLS    #2, SYS$PARSE                            ;
                           58              50  E9 0007D          BLBC     R0, 7$                                   ;
                           50       34     A2  9E 00080          MOVAB    52(R2), R0                               ; 1352
               30          60              05  E0 00084          BBS      #5  (R0), 3$                             ;
               24          60              04  E0 00088          BBS      #[illegible]), 2$                        ; 1353
               20          60              03  E0 0008C          BBS      # [illegible], 2$                        ; 1354
               1C          60              12  E0 00090          BBS      [illegible], 2$                          ; 1355
               18          60              02  E0 00094          BBS      # [illegible] (R0), 2$                   ; 1356
               14          60              01  E0 00098          BBS      # [illegible] (R0), 2$                   ; 1357
                           11              60  E8 0009C          BLBS     (R0), 2$                                 ; 1358
               08          60              06  E0 0009F          BBS      #6, (R0), 1$                             ; 1359
                                           60  95 000A3          TSTB     (R0)                                     ; 1360
                                           04  19 000A5          BLSS     1$                                       ;
```

```
05         60   11 E1 000A7       BBC     #17,    ), 2$              ; 1361
   0000G   CF   08 88 000AB 1$:   BISB2   #8, C(  $SEM_STATUS        ; 1363
04         60   05 E0 000B0 2$:   BBS     #5, (F  , 3$               ; 1370
18         60   02 E0 000B4       BBS     #2, (F  , 6$               ;
04         60   04 E0 000B8 3$:   BBS     #4, (R   4$                ; 1372
10         60   01 E0 000BC       BBS     #1, (R, . 6$               ;
           04   60 E8 000C0 4$:   BLBS    (R0),                      ; 1374
09         60   03 E0 000C3       BBS     #3, (RC  6$                ;
05         60   12 E0 000C7 5$:   BBS     #18, (R  , 6$              ; 1376
   0000G   CF   01 88 000CB       BISB2   #1, COF $SEM_STATUS+3      ; 1378
      10   A2   53 D0 000D0 6$:   MOVL    TEMP_R  , 16(R2)           ; 1384
           50   01 D0 000D4       MOVL    #1, R0                     ; 1390
                   04 000D7       RET                                ;
                50 D4 000D8 7$:   CLRL    R0                         ; 1392
                   04 000DA       RET                                ;

; Routine Size:  219 bytes,    Routine Base:  $CODE$ + 0181
```

```
;  864          1393  1 GLOBAL ROUTINE copy$opn_outfil (output_fab, output_rab, input_fab, out_file_count) =
;  865          1394  1                                                     ! Opens the current output file
;  866          1395  1
;  867          1396  1 !++
;  868          1397  1 ! Functional description:
;  869          1398  1 !
;  870          1399  1 !       This routine opens the current output file. If it is already open due
;  871          1400  1 !       to input file concatenation, the output file RAB is simply disconnected from
;  872          1401  1 !       the FAB to permit switching from block mode I/O to record mode I/O.
;  873          1402  1 !
;  874          1403  1 !       Many of the fields in the input FAB and XAB blocks are copied into the corresponding
;  875          1404  1 !       output FAB and XAB blocks. Also, bits and values are set in the output XAB and FAB blocks
;  876          1405  1 !       because of output file qualifiers specified on the command.
;  877          1406  1 !
;  878          1407  1 !       If the output file already exists, and is being overwritten, it is opened
;  879          1408  1 !       for output. If the output file does not exist, it is allocated and then opened.
;  880          1409  1 !
;  881          1410  1 ! Calling sequence:
;  882          1411  1 !
;  883          1412  1 !       copy$opn_outfil (output_fab.ra.v, output_rab.ra.v, input_fab.ra.v, out_file_count.wl.r)
;  884          1413  1 !
;  885          1414  1 ! Input parameters:
;  886          1415  1 !
;  887          1416  1 !       output_fab      - the address of the FAB associated with the output file
;  888          1417  1 !       output_rab      - the address of the RAB to be used with the output file
;  889          1418  1 !       input_fab       - the address of the FAB associated with the input file
;  890          1419  1 !
;  891          1420  1 ! Implicit inputs:
;  892          1421  1 !
;  893          1422  1 !       copy$cli_status - the OUTFILE_OPEN bit indicates whether an output file is already open.
;  894          1423  1 !                       - bits indicate the settings of the output file qualifiers
;  895          1424  1 !
;  896          1425  1 !       Fields from the input NAM and XAB block are used in the output NAM and XAB blocks.
;  897          1426  1 !
;  898          1427  1 ! Output parameters:
;  899          1428  1 !
;  900          1429  1 !       out_file_count  - a counter that is incremented if a new file is opened.
;  901          1430  1 !
;  902          1431  1 ! Implicit outputs:
;  903          1432  1 !
;  904          1433  1 !       copy$cli_status - OUTFILE_OPEN is set once the file is opened.
;  905          1434  1 !                       - EXTEND_OUTFILE is set if the output file is being extended.
;  906          1435  1 !
;  907          1436  1 !       Fields are written in the output_fab and its associated NAM and XAB blocks.
;  908          1437  1 !
;  909          1438  1 !       out_name_desc   - a descriptor for the output file. Its length field is written.
;  910          1439  1 !
;  911          1440  1 !       When the output file name is parsed, various bits are set in
;  912          1441  1 !       COPY$SEM_STATUS. These include:
;  913          1442  1 !
;  914          1443  1 !               wild_output     - output spec includes explicit wildcards
;  915          1444  1 !               wild_out_ver    - explicit wildcard version number
;  916          1445  1 !
;  917          1446  1 ! Routine value
;  918          1447  1 !
;  919          1448  1 !       OK              - output file successfully created or readied for more output
;  920          1449  1 !       NO_FILE         - output file could not be opened, created, or readied for output
```

```
;  921          1450  1 !
;  922          1451  1 ! Side effects:
;  923          1452  1 !
;  924          1453  1 !       The routine SETUP_EXTEND is called if the output file is open. The value of this call
;  925          1454  1 !               is returned to the caller.
;  926          1455  1 !       The routine SETUP_OUTXAB is called to write most of the output XAB block fields.
;  927          1456  1 !       Messages are output if a file was created during an APPEND command, if versions were
;  928          1457  1 !               slipped under higher existing versions, or if files were replaced or overlaid.
;  929          1458  1 !
;  930          1459  1 !--
;  931          1460  1
;  932          1461  2     BEGIN
;  933          1462  2
;  934          1463  2     MAP
;  935          1464  2         output_fab      : REF BLOCK [, BYTE],           ! FAB to use with output file
;  936          1465  2         output_rab      : REF BLOCK [, BYTE],           ! RAB to use with output file
;  937          1466  2         input_fab       : REF BLOCK [, BYTE],           ! FAB of the current input file
;  938          1467  2         out_file_count  : REF VECTOR;                   ! pointer to number of output files written
;  939          1468  2
;  940          1469  2     BIND
;  941          1470  2         output_nam      =                               ! output NAM block address
;  942          1471  2                 .output_fab [fab$l_nam]         : BLOCK [, BYTE],
;  943          1472  2         output_xabfhc   =                               ! output XAB file header characteristics block
;  944          1473  2                 .output_fab [fab$l_xab]         : BLOCK [, BYTE],
;  945          1474  2         output_xaball   =                               ! output XAB date block
;  946          1475  2                 .output_xabfhc [xab$l_nxt]      : BLOCK [, BYTE],
;  947          1476  2         output_xabdat   =                               ! output XAB date block
;  948          1477  2                 .output_xaball [xab$l_nxt]      : BLOCK [, BYTE],
;  949          1478  2         output_xabrdt   =                               ! output XAB date block
;  950          1479  2                 .output_xabdat [xab$l_nxt]      : BLOCK [, BYTE],
;  951          1480  2         output_xabpro   =                               ! output XAB date block
;  952          1481  2                 .output_xabrdt [xab$l_nxt]      : BLOCK [, BYTE];
;  953          1482  2
;  954          1483  2     LOCAL
;  955          1484  2         status;                                         ! Status variable for calling semantic routine.
;  956          1485  2
;  957          1486  2 !
;  958          1487  2 ! If the output file is already open (due to input file concatenation), call a routine,
;  959          1488  2 ! SETUP_EXTEND, to prepare the file to contain more data.
;  960          1489  2 !
;  961          1490  2
;  962          1491  2     IF .outfile_open                                    ! If the output file is already open,
;  963          1492  2     THEN
;  964          1493  2         RETURN setup_extend (                           !    call a routine to set the file up
;  965          1494  2                              .output_rab);              !    to be extended.
;  966          1495  2
;  967          1496  2 !
;  968          1497  2 ! Copy a set of FAB attributes from the input to the output FAB.
;  969          1498  2 !
;  970          1499  2
;  971          1500  2     output_fab [fab$b_org] = .input_fab [fab$b_org];    ! The fields copied are file organization,
;  972          1501  2     output_fab [fab$b_rat] = .input_fab [fab$b_rat];    !    record attributes
;  973          1502  2     output_fab [fab$w_mrs] = .input_fab [fab$w_mrs];    !    maximum record size
;  974          1503  2     output_fab [fab$l_mrn] = .input_fab [fab$l_mrn];    !    maximum record number
;  975          1504  2     output_fab [fab$b_rfm] = .input_fab [fab$b_rfm];    !    record format
;  976          1505  2     output_fab [fab$b_fsz] = .input_fab [fab$b_fsz];    !    fixed control area size
;  977          1506  2     output_fab [fab$b_bks] = .input_fab [fab$b_bks];    !    bucket size
```

```
;  978          1507  2     output_fab [fab$w_gbc] = .input_fab [fab$w_gbc];     !    global buffer count
;  979          1508  2
;  980          1509  2 !
;  981          1510  2 ! If the input file has read or write checking options, copy them to the output file.
;  982          1511  2 !
;  983          1512  2
;  984          1513  2     output_fab [fab$l_fop] = .output_fab [fab$l_fop] OR ! OR together the current FOP output field
;  985          1514  2         (.input_fab [fab$l_fop] AND (fab$m_rck OR fab$m_wck));
;  986          1515  2                                                      !    and the read and write check bits of the
;  987          1516  2                                                      !    FOP input field.
;  988          1517  2
;  989          1518  2 !
;  990          1519  2 ! Decide on block or record I/O.
;  991          1520  2 !
;  992          1521  2
;  993          1522  2     IF .input_fab [fab$b_org] EQL fab$c_seq          ! If the input file is a sequential file,
;  994          1523  2     THEN
;  995          1524  2         output_fab [fab$v_bro] = TRUE                !    then indicate mixed block and record I/O.
;  996          1525  2     ELSE
;  997          1526  3         BEGIN
;  998          1527  3         output_fab [fab$v_bio] = true;               ! Otherwise, indicate only block I/O.
;  999          1528  3         output_fab [fab$v_bro] = false;              ! and turn off block/record I/O
; 1000          1529  2         END;
; 1001          1530  2
; 1002          1531  2 !
; 1003          1532  2 ! Copy input blocksize for tapes. Otherwise let RMS set the output blocksize.
; 1004          1533  2 !
; 1005          1534  2
; 1006          1535  2     IF .input_fab [$FAB_DEV (sqd)]                   ! If input device is a tape,
; 1007          1536  2     THEN
; 1008          1537  2         output_fab [fab$w_bls] = .input_fab [fab$w_bls] !    then copy the blocksize to the output FAB.
; 1009          1538  2     ELSE
; 1010          1539  2         output_fab [fab$w_bls] = 0;                  ! Otherwise, let RMS choose blocksize.
; 1011          1540  2
; 1012          1541  2 !
; 1013          1542  2 ! Test the expanded name string for the output file. Does it contain wildcards? If so,
; 1014          1543  2 ! is there an explicit wildcard version number?
; 1015          1544  2 !
; 1016          1545  2
; 1017          1546  2     IF .output_nam [nam$v_wildcard]                  ! If there were any wildcards,
; 1018          1547  2     THEN
; 1019          1548  2         wild_output = TRUE;                          !    set flag saying that the file specification
; 1020          1549  2                                                      !    contained some wildcard fields.
; 1021          1550  2
; 1022          1551  2     IF .output_nam [nam$v_wild_ver]                  ! If the version number is a wildcard,
; 1023          1552  2     THEN
; 1024          1553  2        wild_out_ver = TRUE                           !    output version number, remember it.
; 1025          1554  2     ELSE
; 1026          1555  2         IF .output_nam [nam$v_exp_ver]               ! Otherwise, see if an explicit version number was s
; 1027          1556  2         THEN
; 1028          1557  2             exp_out_ver = TRUE;                      !    If so, set the EXP_OUT_VER flag.
; 1029          1558  2
; 1030          1559  2 !
; 1031          1560  2 ! Reparse the output string with a wildcard version number, if this is not
; 1032          1561  2 ! an APPEND operation and one of the following cases is true:
; 1033          1562  2 !       - no output file name, type or version number was given
; 1034          1563  2 !         (e.g. COPY x.x [dir])
```

```
; 1035          1564  2 !       - wild or explicit version numbers were given for the input file, but
; 1036          1565  2 !         the version field for the output file was not specified
; 1037          1566  2 !         (e.g. COPY x.x;* a.a)
; 1038          1567  2 !       - the output spec is wild (e.g. COPY x.x *, or COPY x.x *.*)
; 1039          1568  2
; 1040          1569  2     IF NOT .append_command
; 1041          1570  2               AND
; 1042          1571  3         (.no_output_spec
; 1043          1572  3               OR
; 1044          1573  5         ((.wild_inp_ver OR .exp_inp_ver)
; 1045          1574  4         AND NOT  output_nam [nam$v_wild_ver]
; 1046          1575  4         AND NOT .output_nam [nam$v_exp_ver])
; 1047          1576  3               OR
; 1048          1577  4         (NOT .output_nam [nam$v_exp_ver]
; 1049          1578  5         AND (.output_nam [nam$v_wild_type] OR NOT .output_nam [nam$v_exp_type])
; 1050          1579  3         AND .output_nam [nam$v_wild_name]))
; 1051          1580  2     THEN
; 1052          1581  3         BEGIN                                       ! Then provide a default name string
; 1053          1582  3         output_fab [fab$l_dna] = UPLIT (';*');      !    of an explicit output wildcard
; 1054          1583  3         output_fab [fab$b_dns] = 2;                 !    version number,
; 1055          1584  2         END;
; 1056          1585  2
; 1057          1586  2
; 1058          1587  2     ! Now $PARSE (this may be a reparse) the output file specification.
; 1059          1588  2     !
; 1060          1589  3     IF NOT $RMS_PARSE ( FAB = .output_fab, ERR = copy$outopn_err)
; 1061          1590  2     THEN
; 1062          1591  2             RETURN no_file;                         ! On failure, return with an error code.
; 1063          1592  2
; 1064          1593  2
; 1065          1594  2     !
; 1066          1595  2     ! No director wildcards allowed to remain at this time
; 1067          1596  2     !
; 1068          1597  3     BEGIN
; 1069          1598  3     BIND
; 1070          1599  3         lastchar = .output_nam[nam$l_dir] + .output_nam[nam$b_dir] - 2 : byte;
; 1071          1600  3     IF .lastchar EQL %C'*' OR
; 1072          1601  3         .lastchar EQL %C'.'
; 1073          1602  3     THEN
; 1074          1603  4         BEGIN
; 1075          1604  4         LOCAL
; 1076          1605  4            outputstr : vector[2];
; 1077          1606  4         outputstr[0] = .output_nam [nam$b_esl];
; 1078          1607  4         outputstr[1] = .output_nam [nam$l_esa];
; 1079        P 1608  4         PUT_MESSAGE( MSG$_SYNTAX,
; 1080        P 1609  4                      1,
; 1081        P 1610  4                      outputstr,
; 1082          1611  4                      0 );
; 1083          1612  4         RETURN no_file;
; 1084          1613  3         END;
; 1085          1614  2     END;
; 1086          1615  2
; 1087          1616  2     !
; 1088          1617  2     ! See if the output file fits the criteria given on the command line.
; 1089          1618  2     !
; 1090          1619  3     IF NOT (status = copy$check_file_for_match())
; 1091          1620  2     THEN
```

```
;  1092     1621  2          RETURN .status;
;  1093     1622  2
;  1094     1623  2  !
;  1095     1624  2  ! Call the routine SETUP_OUTXAB to copy output XAB fields from the corresponding input XAB fields.
;  1096     1625  2  !
;  1097     1626  2
;  1098     1627  2      setup_outxab (                                      ! Write output XAB fields by calling
;  1099     1628  2                    .output_fab,                          !    a routine that selects the necessary fields fro
;  1100     1629  2                    .input_fab);                          !    the input FAB and writes them into the output F
;  1101     1630  2
;  1102     1631  2  !
;  1103     1632  2  ! Call the routine APPLY_OUT_QUAL to write RMS fields according to output parameter qualifiers.
;  1104     1633  2  !
;  1105     1634  2
;  1106     1635  2      apply_out_qual (                                    ! Process output file qualifiers
;  1107     1636  2                      .output_fab);
;  1108     1637  2
;  1109     1638  2  !
;  1110     1639  2  ! Call the routine COPY$SEMANTICS to determine the semantic effects of
;  1111     1640  2  ! this particular combination of input and output file specifications and qualifiers.
;  1112     1641  2  !
;  1113     1642  2
;  1114     1643  2      IF NOT copy$semantics (                             ! Decide what semantic behavior is required.
;  1115     1644  2                             copy$sem_status,             !    Pass the status variable copy$sem_status,
;  1116     1645  2                             .input_fab,                  !    the input FAB block address,
;  1117     1646  2                             .output_fab)                 !    and the output FAB block address.
;  1118     1647  2      THEN                                                ! If the input/output spec combination makes no sens
;  1119     1648  2          RETURN no_file;                                 !    then return with error status code.
;  1120     1649  2
;  1121     1650  2  !
;  1122     1651  2  ! Perform special XAB setup if a concatenated file is being created.
;  1123     1652  2  !
;  1124     1653  2
;  1125     1654  3      IF (.append_command                                 ! If appending to existing file,
;  1126     1655  3          OR .concat_follows                              !    or concatenating
;  1127     1656  3          OR NOT .no_expl_out_fields                      !    or if explicit field in output spec
;  1128     1657  3          OR NOT .input_fab [$fab_dev (fod``)             !    or the input device is not file structured,
;  1129     1658  2      THEN
;  1130     1659  2          output_xaball [xab$l_nxt] = .output_xabrdt [xab$l_nxt] ! Do not provide any date information
;  1131     1660  2      ELSE
;  1132     1661  3          BEGIN
;  1133     1662  3          output_xaball [xab$l_nxt] = output_xabdat;      ! Otherwise, include the output date/time XAB block
;  1134     1663  3          output_xabdat [xab$l_nxt] = output_xabrdt;      !     and the revision date/time XAB block;
;  1135     1664  2          END;
;  1136     1665  2
;  1137     1666  2  !
;  1138     1667  2  ! Create (or simply open) the output file.
;  1139     1668  2  !
;  1140     1669  2
;  1141     1670  2      extend_outfile = FALSE;                             ! Assume that the output file is not being extended.
;  1142     1671  2
;  1143     1672  2  !
;  1144     1673  2  ! If a file needn't be created, just open an existing file.
;  1145     1674  2  !
;  1146     1675  2
;  1147     1676  2      IF .append_command AND                              ! If this is an APPEND command and
;  1148     1677  2          NOT .new_version_qual                           !    and output file creation was not requested,
```

```
;  1149          1678  2     THEN
;  1150          1679  3         BEGIN
;  1151          1680  4             IF NOT (status = $RMS_OPEN ( FAB = .output_fab, ERR = copy$outopn_err))
;  1152          1681  3             THEN RETURN .status;
;  1153          1682  3         END
;  1154          1683  2     ELSE
;  1155          1684  3         BEGIN
;  1156          1685  3         status = $RMS_CREATE (FAB = .output_fab);       ! Else, create (or open if it exists) file
;  1157          1686  3
;  1158          1687  3     !
;  1159          1688  3     ! If the file could not be created as a contiguous file bec. se the disk was too full,
;  1160          1689  3     ! then try to create it contiguous best try.
;  1161          1690  3     !
;  1162          1691  3
;  1163          1692  3         IF .status EQL rms$_ful
;  1164          1693  3             AND .output_xaball [xab$v_ctg]
;  1165          1694  4             AND NOT qualifier_active( contig_qual, loc_contig_qual, neg_contig_qual )
;  1166          1695  3         THEN
;  1167          1696  4             BEGIN
;  1168          1697  4             output_xaball [xab$v_ctg] = FALSE;          !    then turn off the contiguous indicator,
;  1169          1698  4             output_xaball [xab$v_cbt] = TRUE;           !    turn on the contiguous best try indicator,
;  1170        P 1699  4             status = $RMS_CREATE (                      !    and retry the create.
;  1171        P 1700  4                                     FAB = .output_fab,  ! Specify the address of the FAB block
;  1172          1701  4                                     ERR = copy$outopn_err);
;  1173          1702  4             IF .status                                  ! If contig-best-try ok,
;  1174          1703  4             THEN
;  1175          1704  4                 put_message (msg$_cbt);                 ! then issue message
;  1176          1705  4             END                                         !    and an error action routine.
;  1177          1706  3         ELSE
;  1178          1707  3         IF NOT .status                                  ! Else, if error,
;  1179          1708  3         THEN
;  1180          1709  3             copy$outopn_err (.output_fab);              ! issue error message
;  1181          1710  3
;  1182          1711  3     !
;  1183          1712  3     ! Change the RMS return status to "created" if indeed the file was created.
;  1184          1713  3     !
;  1185          1714  3
;  1186          1715  3         IF NOT .output_fab [fab$v_cif] AND              ! Sinc. RMS returns RMS$_NORMAL whether or not the
;  1187          1716  3             .status EQL rms$_normal                     !    file was created, for internal reporting, chang
;  1188          1717  3         THEN                                            !    the status code to RMS$_CREATED if appropriate.
;  1189          1718  3             status = rms$_created;                      !    (I.e., if the file was created.)
;  1190          1719  3
;  1191          1720  3     ! If the file was indeed created, issue a $DISPLAY to obtain information
;  1192          1721  3     ! about the newly created file.
;  1193          1722  3
;  1194          1723  3         IF NOT .status                                  ! If the open or create failed,
;  1195          1724  3         THEN
;  1196          1725  3             RETURN no_file;                             !    then return an error status code.
;  1197          1726  3
;  1198          1727  4         IF NOT $RMS_DISPLAY (FAB = .output_fab)
;  1199          1728  3         THEN
;  1200          1729  3             copy$outopn_err (.output_fab);
;  1201          1730  3     !
;  1202          1731  3     ! If the output file was copied to a 10,20 or RT node and it was forced to a
;  1203          1732  3     ! stream format file, then (if the /LOG qualifier was specified) warn the user
;  1204          1733  3     ! of the conversion.
;  1205          1734  3     !
```

```
;  1206          1735  3          IF .status EQL rms$_cre_stm AND .LOG_MSG_QUAL
;  1207          1736  3          THEN
;  1208          1737  4              BEGIN
;  1209          1738  4              out_name_desc [0] = .output_nam [nam$b_rsl]; ! Store the length of the filespec
;  1210        P 1739  4              put_message (msg$_createdstm,1,             ! Issue the message
;  1211          1740  4                           out_name_desc);
;  1212          1741  4              status = rms$_created;                      ! Change the status as above
;  1213          1742  3              END;
;  1214          1743  2          END;
;  1215          1744  2
;  1216          1745  2      outfile_open = TRUE;                                ! Otherwise, set a flag saying that an output file i
;  1217          1746  2      out_name_desc [0] = .output_nam [nam$b_rsl];        !    and store the length of the file specification.
;  1218          1747  2
;  1219          1748  2  !
;  1220          1749  2  ! Clean up the output open procedure by reporting to the user if necessary and
;  1221          1750  2  ! updating more fields.
;  1222          1751  2  !
;  1223          1752  2
;  1224          1753  2      SELECTONE .status OF                                ! Select additional processing based on the
;  1225          1754  2
;  1226          1755  2          SET                                             !    RMS completion code from the OPEN or CREATE.
;  1227          1756  2
;  1228          1757  2          [rms$_created]:                                 ! Output file was created.
;  1229          1758  3              BEGIN
;  1230          1759  3              out_file_count [0] =                        !    Update count of files created.
;  1231          1760  3                  .out_file_count [0] + 1;
;  1232          1761  3
;  1233          1762  3              IF .append_command                          !    If this is an APPEND command,
;  1234          1763  3              THEN
;  1235          1764  3                  copy$log_msg (                          !    send the following message to the user:
;  1236          1765  3                                msg$_created);            !        "<file-name> created" because creation is u
;  1237          1766  3
;  1238          1767  3              IF .output_nam [nam$v_highver] AND          !    If a higher version of this file exists,
;  1239          1768  3                  NOT .quiet_slip                         !    and warnings about versions are not suppressed,
;  1240          1769  3              THEN
;  1241        P 1770  3                  put_message (                           !    send the following message to the user:
;  1242        P 1771  3                                msg$_highver, 1,          !        "higher version of <file-name> exists"
;  1243          1772  3                                out_name_desc);           !    because this may cause version confusion.
;  1244          1773  3
;  1245          1774  2              END;
;  1246          1775  2
;  1247          1776  2
;  1248          1777  2          [rms$_supersede]:                               ! Output file caused deletion of file of same name.
;  1249          1778  3              BEGIN
;  1250          1779  3              out_file_count [0] =                        !    Update count of files created.
;  1251          1780  3                  .out_file_count [0] + 1;
;  1252          1781  3
;  1253          1782  3              copy$log_msg (                              !    Send the following message to the user:
;  1254          1783  3                            msg$_replaced);               !        "<file-name> replaced" because
;  1255          1784  3                                                          !        supersession is unusual.
;  1256          1785  3
;  1257          1786  2              END;
;  1258          1787  2
;  1259          1788  2
;  1260          1789  2          [rms$_normal]:                                  ! Output file existed previously and was opened.
;  1261          1790  3              BEGIN
;  1262          1791  3              IF .append_command                          !    If this is an APPEND command,
```

```
;  1263          1792  3            THEN
;  1264          1793  4                BEGIN
;  1265          1794  4                extend_outfile = TRUE;                  !    set a flag saying that the file is being extend
;  1266          1795  4
;  1267          1796  4                output_xaball [xab$l_alq] =             !    Calculate the necessary extension quantity
;  1268          1797  4                    copy$calc_alq ();                   !    with a call to COPY$CALC_ALQ.
;  1269          1798  4
;  1270          1799  4                IF .output_xaball [xab$l_alq] NEQ 0     !    If the extension quantity is not null,
;  1271          1800  4                THEN
;  1272        P 1801  4                    IF NOT $RMS_EXTEND (                !    then try to extend the file.
;  1273        P 1802  4                                        FAB = .output_fab,
;  1274          1803  5                                        ERR = copy$outopn_err)
;  1275          1804  4                    THEN                                !    If the extend fails,
;  1276          1805  4                        RETURN no_file;                 !    then return an error status code.
;  1277          1806  4
;  1278          1807  4                END
;  1279          1808  4
;  1280          1809  3            ELSE                                        !    If this is a COPY command,
;  1281          1810  4                BEGIN
;  1282          1811  4                copy$log_msg (                          !    send the following message to the user:
;  1283          1812  4                              msg$_overlay);            !        "<file-name> being overwritten"
;  1284          1813  4
;  1285          1814  4 ! ******
;  1286          1815  4 !       Omitted here is the revision of the output file's attributes. Ward had this
;  1287          1816  4 !       commented out.
;  1288          1817  4 ! ******
;  1289          1818  4
;  1290          1819  3                END;
;  1291          1820  3
;  1292          1821  2            END;
;  1293          1822  2
;  1294          1823  2        TES;                                            ! End of SELECT expression.
;  1295          1824  2
;  1296          1825  2 !
;  1297          1826  2 ! Return to the caller with a success status code.
;  1298          1827  2 !
;  1299          1828  2
;  1300          1829  2     RETURN ok;                                         ! Return with a success code.
;  1301          1830  1     END;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                                00  00  2A  3B  00020 P.AAE:  .ASCII  \;*\<0><0>                                       ;

                                                        .EXTRN  SYS$CREATE, SYS$DISPLAY
                                                        .EXTRN  SYS$EXTEND

                                                        .PSECT  $CODE$,NOWRT,2

                                        0FFC 00000      .ENTRY  COPY$OPN_OUTFIL, Save R2,R3,R4,R5,R6,R7,R8,-; 1393
                                                                R9,R10,R11                                      ;
                                5B      0000G  CF  9E 00002     MOVAB   COPY$CLI_STATUS, R11                    ;
                                5A      0000G  CF  9E 00007     MOVAB   COPY$SEM_STATUS, R10                    ;
                                5E             08  C2 0000C     SUBL2   #8, SP                                  ;
                                53       04    AC  D0 0000F     MOVL    OUTPUT_FAB, R3                          ; 1471
```

```
                  56        28   A3  D0 00013          MOVL    40(R3), R6                      ;
                  50        24   A3  D0 00017          MOVL    36(R3), R0                      ; 1473
                  55        04   A0  D0 0001B          MOVL    4(R0), R5                       ; 1475
                  58        04   A5  D0 0001F          MOVL    4(R5), R8                       ; 1477
                  59        04   A8  D0 00023          MOVL    4(R8), R9                       ; 1479
    09       02   A:             01  E1 00027          BBC     #1, COPY$SEM_STATUS+2, 1$       ; 1491
                            08   AC  DD 0002C          PUSHL   OUTPUT_RAB                      ; 1494
         0000V    CF             01  FB 0002F          CALLS   #1, SETUP_EXTEND                ;
                                     04 00034          RET                                     ; 1493
                  52        0C   AC  D0 00035 1$:      MOVL    INPUT_FAB, R2                   ; 1500
             1D   A3        1D   A2  B0 00039          MOVW    29(R2), 29(R3)                  ;
             36   A3        36   A2  B0 0003E          MOVW    54(R2), 54(R3)                  ; 1502
             38   A3        38   A2  D0 00043          MOVL    56(R2), 56(R3)                  ; 1503
             1F   A3        1F   A2  90 00048          MOVB    31(R2), 31(P3)                  ; 1504
             3E   A3        3E   A2  B0 0004D          MOVW    62(R2), 62(R3)                  ; 1506
             48   A3        48   A2  B0 00052          MOVW    72(R2), 72(R3)                  ; 1507
    50       04   A2 FF7FFDFF    8F  CB 00057          BICL3   #-8389121, 4(R2), R0            ; 1514
             04   A3             50  C8 00060          BISL2   R0, 4(R3)                       ;
                            1D   A2  95 00064          TSTB    29(R2)                          ; 1522
                                 07  12 00067          BNEQ    2$                              ;
             16   A3        40   8F  88 00069          BISB2   #64, 22(R3)                     ; 1524
                                 09  11 0006E          BRB     3$                              ;
             16   A3             20  88 00070 2$:      BISB2   #32, 22(R3)                     ; 1527
             16   A3        40   8F  8A 00074          BICB2   #64, 22(R3)                     ; 1528
    07       40   A2             05  E1 00079 3$:      BBC     #5, 64(R2), 4$                  ; 1535
             3C   A3        3C   A2  B0 0007E          MOVW    60(R2), 60(R3)                  ; 1537
                                 03  11 00083          BRB     5$                              ;
                            3C   A3  B4 00085 4$:      CLRW    60(R3)                          ; 1539
                  57        34   A6  9E 00088 5$:      MOVAB   52(R6), R7                      ; 1546
                  04        01   A7  E9 0008C          BLBC    1(R7), 6$                       ;
                  6A        40   8F  88 00090          BISB2   #64, COPY$SEM_STATUS            ; 1548
    06            67             03  E1 00094 6$:      BBC     #3, (R7), 7$                    ; 1551
                  6A        80   8F  88 00098          BISB2   #128, COPY$SEM_STATUS           ; 1553
                                 06  11 0009C          BRB     8$                              ;
                  03             67  E9 0009E 7$:      BLBC    (R7), 8$                        ; 1555
                  6A             04  88 000A1          BISB2   #4, COPY$SEM_STATUS             ; 1557
                  2C             6B  E8 000A4 8$:      BLBS    COPY$CLI_STATUS, 13$            ; 1569
    1E            6A             03  E0 000A7          BBS     #3, COPY$SEM_STATUS, 12$        ; 1571
    04            6A             05  E0 000AB          BBS     #5, COPY$SEM_STATUS, 9$         ; 1573
    07            6A             0¹  E1 000AF          BBC     #1, COPY$SEM_STATUS, 10$        ;
    03            67             03  E0 000B3 9$:      BBS     #3, (R7), 10$                   ; 1574
                  0F             67  E9 000B7          BLBC    (R7), 12$                       ; 1575
                  16             67  E8 000BA 10$:     BLBS    (R7), 13$                       ; 1577
    04            67             04  E0 000BD          BBS     #4, (R7), 11$                   ; 1578
    0E            67             01  E0 000C1          BBS     #1, (R7), 13$                   ;
    0A            67             05  E1 000C5 11$:     BBC     #5, (k7), 13$                   ; 1579
             30   A3      0000'  CF  9E 000C9 12$:     MOVAB   P.AAE, 48(R3)                   ; 1582
             35   A3             02  90 000CF          MOVB    #2, 53(R3)                      ; 1583
                          0000G  CF  9F 000D3 13$:     PUSHAB  COPY$OUTOPN_ERR                 ; 1589
                                 53  DD 000D7          PUSHL   R3                              ;
      00000000G   00             02  FB 000D9          CALLS   #2, SYS$PARSE                   ;
                  37             50  E9 000E0          BLBC    R0, 15$                         ;
                  50        3A   A6  9A 000E3          MOVZBL  58(R6), R0                      ; 1599
                  50        48   A6  C0 000E7          ADDL2   72(R6), R0                      ;
                  2A        FE   A0  91 000EB          CMPB    -2(R0), #42                     ; 1600
                                 06  13 000EF          BEQL    14$                             ;
                  2E        FE   A0  91 000F1          CMPB    -2(R0), #46                     ; 1601
```

```
                                  26  12 000F5        BNEQ    16$                                  ;
                  6E        0B    A6  9A 000F7  14$:  MOVZBL  11(R6), OUTPUTSTR                    ; 1606
        04        AE        0C    A6  D0 000FB        MOVL    12(R6), OUTPUTSTR+4                  ; 1607
                                  7E  D4 00100        CLRL    -(SP)                                ; 1611
                            04    AE  9F 00102        PUSHAB  OUTPUTSTR                            ;
                                  01  DD 00105        PUSHL   #1                                   ;
                  7E      10FC    8F  3C 00107        MOVZWL  #4348, -(SP)                         ;
     0000G        CF              01  FB 0010C        CALLS   #1, COPY$MSG_NUMBER                  ;
                                  50  DD 00111        PUSHL   R0                                   ;
 00000000G        00              04  FB 00113        CALLS   #4, LIB$STOP                         ;
                                01C7  31 0011A  15$:  BRW     35$                                  ; 1612
     0000G        CF              00  FB 0011D  16$:  CALLS   #0, COPY$CHECK_FILE_FOR_MATCH        ; 1619
                  54              50  D0 00122        MOVL    R0, STATUS                           ;
                  5E              54  E9 00125        BLBC    STATUS, 20$                          ;
                                  52  DD 00128        PUSHL   R2                                   ; 1629
                                  53  DD 0012A        PUSHL   R3                                   ; 1628
     0000V        CF              02  FB 0012C        CALLS   #2, SETUP_OUTXAB                     ;
                                  53  DD 00131        PUSHL   R3                                   ; 1636
     0000V        CF              01  FB 00133        CALLS   #1, APPLY_OUT_QUAL                   ;
                                  0C  BB 00138        PUSHR   #^M<R2,R3>                           ; 1645
                                  5A  DD 0013A        PUSHL   R10                                  ; 1643
     0000G        CF              03  FB 0013C        CALLS   #3, COPY$SEMANTICS                   ;
                  D6              50  E9 00141        BLBC    R0, 15$                              ;
                  0E              6B  E8 00144        BLBS    COPY$CLI_STATUS, 17$                 ; 1654
09        02      AA              03  E0 00147        BBS     #3, COPY$SEM_STATUS+2, 17$           ; 1655
                  05        03    AA  E9 0014C        BLBC    COPY$SEM_STATUS+3, 17$               ; 1656
07        41      A2              06  E0 00150        BBS     #6, 65(R2), 18$                      ; 1657
          04      A5        04    A9  D0 00155  17$:  MOVL    4(R9), 4(R5)                         ; 1659
                                  08  11 0015A        BRB     19$                                  ;
          04      A5              58  D0 0015C  18$:  MOVL    R8, 4(R5)                            ; 1662
          04      A8              59  D0 00160        MOVL    R9, 4(R8)                            ; 1663
          02      AA        80    8F  8A 00164  19$:  BICB2   #128, COPY$SEM_STATUS+2              ; 1670
                  1E              6B  E9 00169        BLBC    COPY$CLI_STATUS, 21$                 ; 1676
1A                6B              04  E0 0016C        BBS     #4, COPY$CLI_STATUS, 21$             ; 1677
                          0000G   CF  9F 00170        PUSHAB  COPY$OUTOPN_ERR                      ; 1680
                                  53  DD 00174        PUSHL   R3                                   ;
 00000000G        00              02  FB 00176        CALLS   #2, SYS$OPEN                         ;
                  54              50  D0 0017D        MOVL    R0, STATUS                           ;
                  03              54  E9 00180        BLBC    STATUS, 20$                          ;
                                00C9  31 00183        BRW     28$                                  ;
                  50              54  D0 00186  20$:  MOVL    STATUS, R0                           ; 1681
                                      04 00189        RET                                          ;
                                  53  DD 0018A  21$:  PUSHL   R3                                   ; 1685
 00000000G        00              01  FB 0018C        CALLS   #1, SYS$CREATE                       ;
                  54              50  D0 00193        MOVL    R0, STATUS                           ;
 00018544         8F              54  D1 00196        CMPL    STATUS, #99652                       ; 1692
                                  45  12 0019D        BNEQ    23$                                  ;
                            08    A5  95 0019F        TSTB    8(R5)                                ; 1693
                                  40  18 001A2        BGEQ    23$                                  ;
05        02      AB              03  E1 001A4        BBC     #3, COPY$CLI_STATUS+2, 22$           ; 1694
36        02      AB              06  E1 001A9        BBC     #6, COPY$CLI_STATUS+2, 23$           ;
31        02      AB              05  E0 001AE  22$:  BBS     #5, COPY$CLI_STATUS+2, 23$           ;
          08      A5        80    8F  8A 001B3        BICB2   #128, 8(R5)                          ; 1697
          08      A5              20  88 001B8        BISB2   #32, 8(R5)                           ; 1698
                          0000G   CF  9F 001BC        PUSHAB  COPY$OUTOPN_ERR                      ; 1701
                                  53  DD 001C0        PUSHL   R3                                   ;
 00000000G        00              02  FB 001C2        CALLS   #2, SYS$CREATE                       ;
```

```
               54             50 D0 001C9          MOVL    R0, STATUS                          ;
               1F             54 E9 001CC          BLBC    STATUS, 24$                         ; 1702
               7E      1288   8F 3C 001CF          MOVZWL  #4744, -(SP)                        ; 1704
        0000G  CF             01 FB 001D4          CALLS   #1, COPY$MSG_NUMBER                 ;
                              50 DD 001D9          PUSHL   R0                                  ;
    00000000G  00             01 FB 001DB          CALLS   #1, LIB$SIGNAL                      ;
                              0A 11 001E2          BRB     24$                                 ; 1692
               07             54 E8 001E4 23$:     BLBS    STATUS, 24$                         ; 1707
                              53 DD 001E7          PUSHL   R3                                  ; 1709
        0000G  CF             01 FB 001E9          CALLS   #1, COPY$OUTOPN_ERR                 ;
  10       07  A3             01 E0 001EE 24$:     BBS     #1, 7(R3), 25$                      ; 1715
     00010001  8F             54 D1 001F3          CMPL    STATUS, #65537                      ; 1716
                              07 12 001FA          BNEQ    25$                                 ;
               54 00010619    8F D0 001FC          MOVL    #67097, STATUS                      ; 1718
               03             54 E8 00203 25$:     BLBS    STATUS, 26$                         ; 1723
                            00DB 31 00206          BRW     35$                                 ;
                              53 DD 00209 26$:     PUSHL   R3                                  ; 1727
    00000000G  00             01 FB 0020B          CALLS   #1, SYS$DISPLAY                     ;
               07             50 E8 00212          BLBS    R0, 27$                             ;
                              53 DD 00215          PUSHL   R3                                  ; 1729
        0000G  CF             01 FB 00217          CALLS   #1, COPY$OUTOPN_ERR                 ;
     00018069  8F             54 D1 0021C 27$:     CMPL    STATUS, #98409                      ; 1735
                              2A 12 00223          BNEQ    28$                                 ;
  26           6B             01 E1 00225          BBC     #1, COPY$CLI_STATUS, 28$            ;
        0000G  CF        03   A6 9A 00229          MOVZBL  3(R6), OUT_NAME_DESC                ; 1738
                      0000G   CF 9F 0022F          PUSHAB  OUT_NAME_DESC                       ; 1740
                              01 DD 00233          PUSHL   #1                                  ;
               7E      12FB   8F 3C 00235          MOVZWL  #4859, -(SP)                        ;
        0000G  CF             01 FB 0023A          CALLS   #1, COPY$MSG_NUMBER                 ;
                              50 DD 0023F          PUSHL   R0                                  ;
    00000000G  00             03 FB 00241          CALLS   #3, LIB$SIGNAL                      ;
               54 00010619    8F D0 00248          MOVL    #67097, STATUS                      ; 1741
           02  AA             02 88 0024F 28$:     BISB2   #2, COPY$SEM_STATUS+2               ; 1745
        0000G  CF        03   A6 9A 00253          MOVZBL  3(R6), OUT_NAME_DESC                ; 1746
     00010619  8F             54 D1 00259          CMPL    STATUS, #67097                      ; 1757
                              33 12 00260          BNEQ    30$                                 ;
                         10   BC D6 00262          INCL    @OUT_FILE_COUNT                     ; 1760
               0A             6B E9 00265          BLBC    COPY$CLI_STATUS, 29$                ; 1762
               7E      1073   8F 3C 00268          MOVZWL  #4211, -(SP)                        ; 1764
        0000G  CF             01 FB 0026D          CALLS   #1, COPY$LOG_MSG                    ;
                              67 B5 00272 29$:     TSTW    (R7)                                ; 1767
                              6A 18 00274          BGEQ    34$                                 ;
               66        02   AA E8 00276          BLBS    COPY$SEM_STATUS+2, 34$              ; 1768
                      0000G   CF 9F 0027A          PUSHAB  OUT_NAME_DESC                       ; 1772
                              01 DD 0027E          PUSHL   #1                                  ;
               7E      1148   8F 3C 00280          MOVZWL  #4424, -(SP)                        ;
        0000G  CF             01 FB 00285          CALLS   #1, COPY$MSG_NUMBER                 ;
                              50 DD 0028A          PUSHL   R0                                  ;
    00000000G  00             03 FB 0028C          CALLS   #3, LIB$SIGNAL                      ;
                              4B 11 00293          BRB     34$                                 ; 1753
     00010631  8F             54 D1 00295 30$:     CMPL    STATUS, #67121                      ; 1777
                              0A 12 0029C          BNEQ    31$                                 ;
                         10   BC D6 0029E          INCL    @OUT_FILE_COUNT                     ; 1780
               7E      10BB   8F 3C 002A1          MOVZWL  #4283, -(SP)                        ; 1782
                              33 11 002A6          BRB     33$                                 ;
     00010001  8F             54 D1 002A8 31$:     CMPL    STATUS, #65537                      ; 1789
                              2F 12 002AF          BNEQ    34$                                 ;
```

```
                22             6B  E9 002B1          BLBC    COPY$C  STATUS, 32$                  ; 1791
          02    AA       80    8F  88 002B4          BISB2   #128, COPY$SEM_STATUS+2               ; 1794
        0000G   CF             00  FB 002B9          CALLS   #0, COPY$ ALC_ALQ                     ; 1797
          10    A5             50  D0 002BE          MOVL    R0, 16(R5)                            ;
                               1C  13 002C2          BEQL    34$                                   ; 1799
                       0000G   CF  9F 002C4          PUSHAB  COPY$OUTOPN_ERR                       ; 1803
                               53  DD 002C8          PUSHL   R3                                    ;
    00000000G   00             02  FB 002CA          CALLS   #2, SYS$EXTEND                        ;
                0C             50  E8 002D1          BLBS    R0, 34$                               ;
                               0E  11 002D4          BRB     35$                                   ; 1805
                7E       10AB  8F  3C 002D6 32$:     MOVZWL  #4267, -(SP)                          ; 1811
        0000G   CF             01  FB 002DB 33$:     CALLS   #1, COPY$LOG_MSG                      ;
                50             01  D0 002E0 34$:     MOVL    #1, R0                                ; 1829
                                   04 002E3          RET                                           ;
                               50  D4 002E4 35$:     CLRL    R0                                    ; 1830
                                   04 002E6          RET                                           ;

; Routine Size:  743 bytes,    Routine Base:  $CODE$ + 025C
```

```
;  1303          1831  1 ROUTINE setup_extend (output_rab) =                    ! Setup a file to be extended.
;  1304          1832  1
;  1305          1833  1 !++
;  1306          1834  1 ! Functional description:
;  1307          1835  1 !
;  1308          1836  1 !       This routine takes an open file and prepares it to be extended.
;  1309          1837  1 !
;  1310          1838  1 !       First, a DISCONNECT is performed. This permits switching from block mode I/O
;  1311          1839  1 !       to record mode I/O, if desired. Then update the output file allocation information,
;  1312          1840  1 !       set a bit in COPY$CLI_STATUS saying that the file is being extended, calculate
;  1313          1841  1 !       the file extension quantity, and extend the file.
;  1314          1842  1 !
;  1315          1843  1 ! Calling sequence:
;  1316          1844  1 !
;  1317          1845  1 !       setup_extend (output_rab.ra.v)
;  1318          1846  1 !
;  1319          1847  1 ! Input parameters
;  1320          1848  1 !
;  1321          1849  1 !       output_rab      - the RAB connected to the output FAB
;  1322          1850  1 !
;  1323          1851  1 ! Implicit inputs
;  1324          1852  1 !
;  1325          1853  1 !       The FAB and XAB blocks associated with the specified output RAB block.
;  1326          1854  1 !
;  1327          1855  1 ! Output parameters
;  1328          1856  1 !
;  1329          1857  1 !       none
;  1330          1858  1 !
;  1331          1859  1 ! Implicit outputs
;  1332          1860  1 !
;  1333          1861  1 !       The allocation information in the FAB is updated.
;  1334          1862  1 !       The EXTEND_OUTFILE bit in COPY$CLI_STATUS is set.
;  1335          1863  1 !       The ALQ field in the output XAB block is set to an appropriate extension quantity.
;  1336          1864  1 !
;  1337          1865  1 ! Routine value
;  1338          1866  1 !
;  1339          1867  1 !       OK              - success
;  1340          1868  1 !       NO_FILE         - failure
;  1341          1869  1 !
;  1342          1870  1 ! Side effects
;  1343          1871  1 !
;  1344          1872  1 !       If the file cannot be extended, the file is closed.
;  1345          1873  1 !
;  1346          1874  1 !--
;  1347          1875  1
;  1348          1876  2     BEGIN
;  1349          1877  2
;  1350          1878  2     MAP
;  1351          1879  2         output_rab      : REF BLOCK [, BYTE];           ! output FAB of the open output file
;  1352          1880  2
;  1353          1881  2     BIND
;  1354          1882  2         output_fab      =                               ! associated output FAB block
;  1355          1883  2                 .output_rab [rab$l_fab]         : BLOCK [, BYTE],
;  1356          1884  2         output_xabfhc   =                               ! associated output XAB block
;  1357          1885  2                 .output_fab [fab$l_xab]         : BLOCK [, BYTE],
;  1358          1886  2         output_xaball   =                               !   second XAB in XAB chain
;  1359          1887  2                 .output_xabfhc [xab$l_nxt]      : BLOCK [, BYTE];
```

```
;  1360          1888  2
;  1361          1889  2      LOCAL
;  1362          1890  2          status;                                     ! Holds RMS status values
;  1363          1891  2
;  1364          1892  2
;  1365          1893  2      ! See if the input file fits the criteria given on the command line.
;  1366          1894  2      !
;  1367          1895  3      IF NOT (status = copy$check_file_for_match())
;  1368          1896  2      THEN
;  1369          1897  2          RETURN .status;
;  1370          1898  2
;  1371          1899  2
;  1372          1900  2      ! Disconnect the RAB from the FAB. On error, close the file and return
;  1373          1901  2      ! with error status code.
;  1374          1902  2      !
;  1375        P 1903  2      IF NOT $RMS_DISCONNECT (                        ! Disconnect the output file RAB from its FAB.
;  1376        P 1904  2                              RAB = .output_rab,       !    Specify the RAB block address
;  1377          1905  3                              ERR = copy$oclose_err)   !    and an error routine.
;  1378          1906  2      THEN
;  1379          1907  3          BEGIN                                       ! If the DISCONNECT fails,
;  1380          1908  3          copy$close_outf (                           !    close the output file,
;  1381          1909  3                           output_fab);
;  1382          1910  3          RETURN no_file;                             !    and return with an error code.
;  1383          1911  2          END;
;  1384          1912  2
;  1385          1913  2  !
;  1386          1914  2  ! Shortening the XAB chain to include only the FHC (file header characteristics) XAB,
;  1387          1915  2  ! call the RMS function $DISPLAY to update the output file allocation information
;  1388          1916  2  ! as recorded in the XABFHC.
;  1389          1917  2  !
;  1390          1918  2
;  1391          1919  2      output_xabfhc [xab$l_nxt] = 0;                  ! Leave only the FHC XAB on the XAB chain.
;  1392          1920  2
;  1393        P 1921  2      status = $RMS_DISPLAY (                         ! Call DISPLAY to update the XAB information
;  1394        P 1922  2                              FAB = output_fab,        !    about the file's allocation.
;  1395          1923  2                              ERR = copy$outopn_err);  !    Specify an error action routine.
;  1396          1924  2
;  1397          1925  2      output_xabfhc [xab$l_nxt] = output_xaball;      ! Restore the XAB chain.
;  1398          1926  2
;  1399          1927  2  !
;  1400          1928  2  ! See if the $DISPLAY function succeeded. If not, close the output file and return
;  1401          1929  2  ! an error status code.
;  1402          1930  2  !
;  1403          1931  2
;  1404          1932  2      IF NOT .status                                  ! If the $DISPLAY function failed,
;  1405          1933  2      THEN
;  1406          1934  3          BEGIN
;  1407          1935  3          copy$close_outf (                           !    then close the output file,
;  1408          1936  3                           output_fab);
;  1409          1937  3          RETURN no_file;                             !    and return an error status code.
;  1410          1938  2          END;
;  1411          1939  2
;  1412          1940  2  !
;  1413          1941  2  ! Set the bit in COPY$CLI_STATUS that indicates that the file is to be extended.
;  1414          1942  2  !
;  1415          1943  2
;  1416          1944  2      extend_outfile = TRUE;                          ! Set EXTEND_OUTFILE bit.
```

```
; 1417          1945  2
; 1418          1946  2 !
; 1419          1947  2 ! Calculate the file extension quantity and extend the file with an RMS $EXTEND function call.
; 1420          1948  2 ! The routine COPY$CALC_ALQ does the calculation. It returns a "zero" in the following cases:
; 1421          1949  2 !
; 1422          1950  2 !       The output file is on a magtape or a nonfile-structured device.
; 1423          1951  2 !       The output file is already long enough to hold the size of the file to be appended.
; 1424          1952  2 !
; 1425          1953  2
; 1426          1954  2     output_xaball [xab$l_alq] = copy$calc_alq ();       ! Setup the output file extension quantity in the XA
; 1427          1955  2
; 1428          1956  2     IF .output_xaball [xab$l_alq] EQL 0                 ! If the input file is of zero length,
; 1429          1957  2     THEN
; 1430          1958  2         RETURN ok;                                      !     then return with success code.
; 1431          1959  2
; 1432        P 1960  2     IF $RMS_EXTEND (                                    ! If the output file can be extended successfully,
; 1433        P 1961  2                 FAB = output_fab,
; 1434          1962  3                 ERR = copy$outopn_err)                  !     (specify an error routine)
; 1435          1963  2     THEN
; 1436          1964  2         RETURN ok                                       !     then return with success code.
; 1437          1965  2     ELSE
; 1438          1966  2         RETURN no_file;                                 ! Otherwise, return with error code.
; 1439          1967  2
; 1440          1968  1     END;


                                                     .EXTRN  SYS$DISCONNECT

                                    007C 00000 SETUP_EXTEND:
                                                     .WORD   Save R2,R3,R4,R5,R6                       ; 1831
          54       04   AC   D0 00002                MOVL    OUTPUT_RAB, R4                            ; 1883
          55       3C   A4   D0 00006                MOVL    60(R4), R5                                ;
          52       24   A5   D0 0000A                MOVL    36(R5), R2                                ; 1885
          53       04   A2   D0 0000E                MOVL    4(R2), R3                                 ; 1887
    0000G CF            00   FB 00012                CALLS   #0, COPY$CHECK_FILE_FOR_MATCH             ; 1895
          56            50   D0 00017                MOVL    R0, STATUS                                ;
          04            56   E8 0001A                BLBS    STATUS, 1$                                ;
          50            56   D0 0001D                MOVL    STATUS, R0                                ; 1897
                             04 00020                RET
                0000G   CF   9F 00021 1$:            PUSHAB  COPY$OCLOSE_ERR                           ; 1905
                        54   DD 00025                PUSHL   R4                                        ;
 00000000G 00           02   FB 00027                CALLS   #2, SYS$DISCONNECT                        ;
          1A            50   E9 0002E                BLBC    R0, 2$                                    ;
                   04   A2   D4 00031                CLRL    4(R2)                                     ; 1919
                0000G   CF   9F 00034                PUSHAB  COPY$OUTOPN_ERR                           ; 1923
                        55   DD 00038                PUSHL   R5                                        ;
 00000000G 00           02   FB 0003A                CALLS   #2, SYS$DISPLAY                           ;
          56            50   D0 00041                MOVL    R0, STATUS                                ;
       04 A2            53   D0 00044                MOVL    R3, 4(R2)                                 ; 1925
          09            56   E8 00048                BLBS    STATUS, 3$                                ; 1932
                        55   DD 0004B 2$:            PUSHL   R5                                        ; 1935
    0000G CF            01   FB 0004D                CALLS   #1, COPY$CLOSE_OUTF                       ;
                        25   11 00052                BRB     5$                                        ; 1937
    0000G CF       80   8F   88 00054 3$:            BISB2   #128, COPY$SEM_STATUS+2                   ; 1944
    0000G CF            00   FB 0005A                CALLS   #0, COPY$CALC_ALQ                         ; 1954
       10 A3            50   D0 0005F                MOVL    R0, 16(R3)                                ;
```

```
                                        10  13 00063          BEQL    4$                                      ; 1956
                                0000G   CF  9F 00065          PUSHAB  COPY$OUTOPN_ERR                         ; 1962
                                        55  DD 00069          PUSHL   R5                                      ;
                        00000000G   00      02  FB 0006B      CALLS   #2, SYS$EXTEND                          ;
                                    04      50  E9 00072      BLBC    R0, 5$                                  ;
                                    50      01  D0 00075 4$:  MOVL    #1, R0                                  ; 1966
                                            04 00078          RET                                             ;
                                        50  D4 00079 5$:      CLRL    R0                                      ; 1968
                                            04 0007B          RET                                             ;

; Routine Size:  124 bytes,    Routine Base:  $CODE$ + 0543
```

```
;  1442     1969  1 ROUTINE setup_outxab (output_fab, input_fab) : NOVALUE =
;  1443     1970  1                                                 ! Setup output XAB fields from input XAB fields
;  1444     1971  1
;  1445     1972  1 !++
;  1446     1973  1 ! Functional description:
;  1447     1974  1 !
;  1448     1975  1 !       This routine copies input XAB fields into corresponding output XAB fields.
;  1449     1976  1 !
;  1450     1977  1 ! Calling sequence:
;  1451     1978  1 !
;  1452     1979  1 !       setup_outxab (output_fab.ra.v, input_fab.ra.v)
;  1453     1980  1 !
;  1454     1981  1 ! Input parameters:
;  1455     1982  1 !
;  1456     1983  1 !       output_fab      - FAB block associated with the output file
;  1457     1984  1 !       input_fab       - FAB block associated with the input file
;  1458     1985  1 !
;  1459     1986  1 ! Implicit inputs:
;  1460     1987  1 !
;  1461     1988  1 !       output_xaball   - XABALL block for output file
;  1462     1989  1 !       output_xabdat   - XABDAT block for output file
;  1463     1990  1 !       output_xabfhc   - XABFHC block for output file
;  1464     1991  1 !       output_xabpro   - XABPRO block for output file
;  1465     1992  1 !       output_xabrdt   - XABRDT block for output file
;  1466     1993  1 !
;  1467     1994  1 !       input_xaball    - XABALL block for input file
;  1468     1995  1 !       input_xabdat    - XABDAT block for input file
;  1469     1996  1 !       input_xabfhc    - XABFHC block for input file
;  1470     1997  1 !       input_xabpro    - XABPRO block for input file
;  1471     1998  1 !
;  1472     1999  1 ! Output parameters
;  1473     2000  1 !
;  1474     2001  1 !       none
;  1475     2002  1 !
;  1476     2003  1 ! Implicit outputs
;  1477     2004  1 !
;  1478     2005  1 !       The relevant fields in the output XABs are written.
;  1479     2006  1 !
;  1480     2007  1 ! Routine value
;  1481     2008  1 !
;  1482     2009  1 !       none
;  1483     2010  1 !
;  1484     2011  1 ! Side effects
;  1485     2012  1 !
;  1486     2013  1 !       none
;  1487     2014  1 !
;  1488     2015  1 !--
;  1489     2016  1
;  1490     2017  2     BEGIN
;  1491     2018  2
;  1492     2019  2     MAP
;  1493     2020  2         output_fab      : REF BLOCK [, BYTE],           ! output file FAB block
;  1494     2021  2         input_fab       : REF BLOCK [, BYTE];           ! input file FAB block
;  1495     2022  2
;  1496     2023  2     BIND
;  1497     2024  2         output_nam      =                               ! output NAM block address
;  1498     2025  2                 .output_fab [fab$l_nam]         : BLOCK [, BYTE],
```

```
;  1499     2026  2          output_xabfhc   =                                   ! output XAB file header characteristics block
;  1500     2027  2                  .output_fab [fab$l_xab]          : BLOCK [, BYTE],
;  1501     2028  2          output_xaball   =                                   ! output XAB date block
;  1502     2029  2                  .output_xabfhc [xab$l_nxt]       : BLOCK [, BYTE],
;  1503     2030  2          output_xabdat   =                                   ! output XAB date block
;  1504     2031  2                  .output_xaball [xab$l_nxt]       : BLOCK [, BYTE],
;  1505     2032  2          output_xabrdt   =                                   ! output XAB date block
;  1506     2033  2                  .output_xabdat [xab$l_nxt]       : BLOCK [, BYTE],
;  1507     2034  2          output_xabpro   =                                   ! output XAB date block
;  1508     2035  2                  .output_xabrdt [xab$l_nxt]       : BLOCK [, BYTE],
;  1509     2036  2
;  1510     2037  2          input_xaball    =                                   ! input file XABALL block
;  1511     2038  2                  .input_fab [fab$l_xab]           : BLOCK [, BYTE],
;  1512     2039  2          input_xabdat    =                                   ! input file XABDAT block
;  1513     2040  2                  .input_xaball [xab$l_nxt]        : BLOCK [, BYTE],
;  1514     2041  2          input_xabfhc    =                                   ! input file XABFHC block
;  1515     2042  2                  .input_xabdat [xab$l_nxt]        : BLOCK [, BYTE],
;  1516     2043  2          input_xabpro    =                                   ! input file XABPRO block
;  1517     2044  2                  .input_xabfhc [xab$l_nxt]        : BLOCK [, BYTE];
;  1518     2045  2
;  1519     2046  2  !
;  1520     2047  2  ! Write the output allocation XAB.
;  1521     2048  2  !
;  1522     2049  2
;  1523     2050  2      output_xaball [xab$b_aop] =
;  1524     2051  2                          .input_xaball [xab$b_aop];      ! Write the allocation options,
;  1525     2052  2      output_xaball [xab$b_aln] =
;  1526     2053  2                          .input_xaball [xab$b_aln];      !    and the alignment type.
;  1527     2054  2
;  1528     2055  2      output_xaball [xab$l_alq] = copy$calc_alq ();       ! Calculate and write in the allocation quantity.
;  1529     2056  2
;  1530     2057  2      output_xaball [xab$w_deq] =
;  1531     2058  2                          .input_xabfhc [xab$w_dxq];      ! Write the default extension quantity.
;  1532     2059  2      output_xaball [xab$b_bkz] =                         ! Write the default bucket size
;  1533     2060  2                          .input_fab [fab$b_bks];         !    from the input FAB bucket size.
;  1534     2061  2                                                          !    This insures the file is created with
;  1535     2062  2                                                          !    correct bucksize.  Area 0 not may have
;  1536     2063  2                                                          !    the largest bucket size.
;  1537     2064  2
;  1538     2065  2      output_xaball [xab$w_vol] = 0;                      ! Zero the related volume number,
;  1539     2066  2      output_xaball [xab$l_loc] = 0;                      !    the allocation location,
;  1540     2067  2      output_xaball [xab$b_aid] = 0;                      !    the area id number,
;  1541     2068  2      output_xaball [xab$w_rfi0] = 0;                     !    the related file number
;  1542     2069  2      output_xaball [xab$w_rfi2] = 0;                     !    the related file sequence number
;  1543     2070  2      output_xaball [xab$w_rfi4] = 0;                     !    and the related file revision number.
;  1544     2071  2
;  1545     2072  2      IF .input_fab [$fab_dev(net)] AND                   ! If this is a network operation
;  1546     2073  2         .output_xaball [xab$l_alq] EQL 0                 !    and the calculated ALQ = 0,
;  1547     2074  2      THEN output_xaball [xab$l_alq] =                    !    then get ALQ from the FHC XAB
;  1548     2075  2                          .input_xabfhc [xab$l_hbk];      !
;  1549     2076  2
;  1550     2077  2  !
;  1551     2078  2  ! Write the output Date/Time XAB.
;  1552     2079  2  !
;  1553     2080  2
;  1554     2081  2      output_xabdat [xab$w_rvn] =                         ! Increment the revision number
;  1555     2082  2                          .input_xabdat [xab$w_rvn ] + 1;
```

```
;  1556          2083  2      output_xabdat [xab$l_rdt0] = 0;                         ! Clear the revision date
;  1557          2084  2      output_xabdat [xab$l_rdt4] = 0;
;  1558          2085  2      output_xabdat [xab$l_cdt0] =                            ! Copy the creation date
;  1559          2086  2                              .input_xabdat [xab$l_cdt0];
;  1560          2087  2      output_xabdat [xab$l_cdt4] =                            !    and the creation time
;  1561          2088  2                              .input_xabdat [xab$l_cdt4];
;  1562          2089  2  !
;  1563          2090  2  ! These values are not copied from the input, but defaulted instead,
;  1564          2091  2  ! so the user will get new backup and expiration dates.
;  1565          2092  2  !
;  1566          2093  2
;  1567          2094  2      ! If the output device is tape, then propogate the expiration date.
;  1568          2095  2      ! Otherwise, clear it.
;  1569          2096  2
;  1570          2097  2      IF .output_fab[ $FAB_DEV(sqd) ]
;  1571          2098  2      THEN
;  1572          2099  3          BEGIN
;  1573          2100  3          output_xabdat [xab$l_edt0] = .input_xabdat [xab$l_edt0];
;  1574          2101  3          output_xabdat [xab$l_edt4] = .input_xabdat [xab$l_edt4];
;  1575          2102  3          END
;  1576          2103  2      ELSE
;  1577          2104  3          BEGIN
;  1578          2105  3          output_xabdat [xab$l_edt0] = 0;
;  1579          2106  3          output_xabdat [xab$l_edt4] = 0;
;  1580          2107  2          END;
;  1581          2108  2
;  1582          2109  2      output_xabdat [xab$l_bdt0] = 0;                         !    the backup date
;  1583          2110  2      output_xabdat [xab$l_bdt4] = 0;                         !    and the backup time
;  1584          2111  2
;  1585          2112  2  !
;  1586          2113  2  ! Write the output File Header Characteristics XAB block.
;  1587          2114  2  !
;  1588          2115  2
;  1589          2116  2      output_xabfhc [xab$b_rfo] =                             ! The XABFHC includes the
;  1590          2117  2                              .input_xabfhc [xab$b_rfo];      !    record format and file organization,
;  1591          2118  2      output_xabfhc [xab$b_atr] =                             !    the record attributes,
;  1592          2119  2                              .input_xabfhc [xab$b_atr];
;  1593          2120  2      output_xabfhc [xab$w_lrl] =                             !    the length of the longest record,
;  1594          2121  2                              .input_xabfhc [xab$w_lrl];
;  1595          2122  2      output_xabfhc [xab$b_bkz] =                             !    the bucket size,
;  1596          2123  2                              .input_xabfhc [xab$b_bkz];
;  1597          2124  2      output_xabfhc [xab$b_hsz] =                             !    the VFC header size,
;  1598          2125  2                              .input_xabfhc [xab$b_hsz];
;  1599          2126  2      output_xabfhc [xab$w_mrz] =                             !    the maximum record length,
;  1600          2127  2                              .input_xabfhc [xab$w_mrz];
;  1601          2128  2      output_xabfhc [xab$w_dxq] =                             !    and the default extension quantity.
;  1602          2129  2                              .input_xabfhc [xab$w_dxq];
;  1603          2130  2
;  1604          2131  2      output_xabfhc [xab$l_sbn] = 0;                          ! Zero the starting virtual block number.
;  1605          2132  2
;  1606          2133  2  !
;  1607          2134  2  ! Write the output Protection XAB block. Most of this XAB can only be setup
;  1608          2135  2  ! after the output file has been opened or created. Therefore, it is not done here.
;  1609          2136  2  !
;  1610          2137  2
;  1611          2138  2      output_xabpro [xab$l_uic] = 0;                          ! Clear the file owner field.
;  1612          2139  2
```

```
; 1613      2140  2 !
; 1614      2141  2 ! Write the output Revision Date/Time XAB block.
; 1615      2142  2 !
; 1616      2143  2
; 1617      2144  2     output_xabrdt [xab$w_rvn] =                        ! Increment revision number
; 1618      2145  2                 .input_xabdat [xab$w_rvn ] + 1;
; 1619      2146  2     output_xabrdt [xab$l_rdt0] = 0;                    ! Do not propogate the the input revision date,
; 1620      2147  2     output_xabrdt [xab$l_rdt4] = 0;
; 1621      2148  2
; 1622      2149  2 ! ******
; 1623      2150  2 !       Temporarily, I omit the special saving of XABDAT and XABFHC fields
; 1624      2151  2 !       of a file that may be overwritten. This must go back in.
; 1625      2152  2 ! ******
; 1626      2153  2
; 1627      2154  1     END;


                                        07FC 00000 SETUP_OUTXAB:
                                                          .WORD    Save R2,R3,R4,R5,R6,R7,R8,R9,R10          ; 1969
               58       04   AC  D0 00002                MOVL     OUTPUT_FAB, R8                            ; 2025
               56       24   A8  D0 00006                MOVL     36(R8), R6                                ; 2027
               52       04   A6  D0 0000A                MOVL     4(R6), R2                                 ; 2029
               53       04   A2  D0 0000E                MOVL     4(R2), R3                                 ; 2031
               59       04   A3  D0 00012                MOVL     4(R3), R9                                 ; 2033
               5A       04   A9  D0 00016                MOVL     4(R9), R10                                ; 2035
               57       08   AC  D0 0001A                MOVL     INPUT_FAB, R7                             ; 2038
               50       24   A7  D0 0001E                MOVL     36(R7), R0                                ;
               54       04   A0  D0 00022                MOVL     4(R0), R4                                 ; 2040
               55       04   A4  D0 00026                MOVL     4(R4), R5                                 ; 2042
         08    A2       08   A0  B0 0002A                MOVW     8(R0), 8(R2)                              ; 2051
      0000G    CF            00  FB 0002F                CALLS    #0, COPY$CALC_ALQ                         ; 2055
         10    A2            50  D0 00034                MOVL     R0, 16(R2)                                ;
         14    A2       1A   A5  B0 00038                MOVW     26(R5), 20(R2)                            ; 2058
                        0A   A2  B4 0003D                CLRW     10(R2)                                    ; 2065
                        0C   A2  D4 00040                CLRL     12(R2)                                    ; 2066
         16    A2       3E   A7  9B 00043                MOVZBW   62(R7), 22(R2)                            ; 2060
                        18   A2  D4 00048                CLRL     24(R2)                                    ; 2068
                        1C   A2  B4 0004B                CLRW     28(R2)                                    ; 2070
   0A    41    A7       05   E1  0004E                   BBC      #5, 65(R7), 1$                            ; 2072
               10       A2   D5  00053                   TSTL     16(R2)                                    ; 2073
                        05   12  00056                   BNEQ     1$                                        ;
         10    A2       0C   A5  D0 00058                MOVL     12(R5), 16(R2)                            ; 2075
               50       08   A4  3C 0005D  1$:           MOVZWL   8(R4), R0                                 ; 2082
                             50  D6 00061                INCL     R0                                        ;
         08    A3            50  B0 00063                MOVW     R0, 8(R3)                                 ;
                        0C   A3  7C 00067                CLRQ     12(R3)                                    ; 2083
         14    A3       14   A4  7D 0006A                MOVQ     20(R4), 20(R3)                            ; 2086
   07    40    A8       05   E1  0006F                   BBC      #5, 64(R8), 2$                            ; 2097
         1C    A3       1C   A4  7D 00074                MOVQ     28(R4), 28(R3)                            ; 2100
                        03   11  00079                   BRB      3$                                        ; 2097
                        1C   A3  7C 0007B  2$:           CLRQ     28(R3)                                    ; 2105
                        24   A3  7C 0007E  3$:           CLRQ     36(R3)                                    ; 2109
         08    A6       08   A5  D0 00081                MOVL     R(R5), 8(R6)                              ; 2117
         16    A6       16   A5  D0 00086                MOVL     22(R5), 22(R6)                            ; 2123
```

```
                 1A   A6       1A   A5  B0 0008B        MOVW    26(R5), 26(R6)                  ; 2129
                              28   A6  D4 00090        CLRL    40(R6)                          ; 2131
                              0C   AA  D4 00093        CLRL    12(R10)                         ; 2138
                 08   A9            50  B0 00096        MOVW    R0, 8(R9)                       ; 2145
                              0C   A9  7C 0009A        CLRQ    12(R9)                          ; 2146
                                       04 0009D        RET                                     ; 2154

; Routine Size:  158 bytes,    Routine Base:  $CODE$ + 05BF
```

```
;  1629          2155  1 ROUTINE apply_out_qual (output_fab) : NOVALUE =          ! Applies output parameter qualifiers to FAB and XAB
;  1630          2156  1
;  1631          2157  1 !++
;  1632          2158  1 ! Functional description
;  1633          2159  1 !
;  1634          2160  1 !       This routine looks for the presence of qualifiers on the output file specification,
;  1635          2161  1 !       and sets RMS fields according to the semantics of each qualifier.
;  1636          2162  1 !
;  1637          2163  1 ! Calling sequence:
;  1638          2164  1 !
;  1639          2165  1 !       apply_out_qual (output_fab.ra.v)
;  1640          2166  1 !
;  1641          2167  1 ! Input parameters:
;  1642          2168  1 !
;  1643          2169  1 !       output_fab      - the FAB block related to the output file specification
;  1644          2170  1 !
;  1645          2171  1 ! Implicit inputs:
;  1646          2172  1 !
;  1647          2173  1 !       output_xaball   - The XABALL block associated with the output FAB
;  1648          2174  1 !
;  1649          2175  1 !       The following bits in COPY$CLI_STATUS:
;  1650          2176  1 !
;  1651          2177  1 !               alignment_bit
;  1652          2178  1 !               allocation_bit
;  1653          2179  1 !               contiguous_bit
;  1654          2180  1 !               extension_bit
;  1655          2181  1 !               file_max_bit
;  1656          2182  1 !               overlay_bit
;  1657          2183  1 !               oread_check_bit
;  1658          2184  1 !               replace_bit
;  1659          2185  1 !               truncate_bit
;  1660          2186  1 !               write_check_bit
;  1661          2187  1 !               volume_bit
;  1662          2188  1 !
;  1663          2189  1 !       Some values associated with qualifiers specified for the output file specification:
;  1664          2190  1 !
;  1665          2191  1 !               align_type
;  1666          2192  1 !               align_option
;  1667          2193  1 !               align_location
;  1668          2194  1 !               alloc_value
;  1669          2195  1 !               extension_value
;  1670          2196  1 !               file_max_value
;  1671          2197  1 !               volume_value
;  1672          2198  1 !
;  1673          2199  1 ! Output parameters
;  1674          2200  1 !
;  1675          2201  1 !       none
;  1676          2202  1 !
;  1677          2203  1 ! Implicit outputs
;  1678          2204  1 !
;  1679          2205  1 !       Some fields in the output XABALL block are written:
;  1680          2206  1 !
;  1681          2207  1 !               ALN     - alignment type
;  1682          2208  1 !               AOP     - alignment option
;  1683          2209  1 !               LOC     - alignment location
;  1684          2210  1 !               ALQ     - allocation quantity
;  1685          2211  1 !               CTG     - contiguous file
```

```
;  1686          2212  1 !                 CBT     - contiguous best try file
;  1687          2213  1 !                 DEQ     - file extension quantity
;  1688          2214  1 !                 VOL     - relative volume number
;  1689          2215  1 !
;  1690          2216  1 !       Some fields in the output FAB are written:
;  1691          2217  1 !
;  1692          2218  1 !                 MRN     - maximum record number
;  1693          2219  1 !                 CIF     - create if nonexistent file
;  1694          2220  1 !                 RCK     - read check
;  1695          2221  1 !                 TEF     - truncate files at EOF mark
;  1696          2222  1 !                 SUP     - supersede
;  1697          2223  1 !                 WCK     - write check
;  1698          2224  1 !
;  1699          2225  1 ! Routine value
;  1700          2226  1 !
;  1701          2227  1 !       novalue
;  1702          2228  1 !
;  1703          2229  1 ! Side effects
;  1704          2230  1 !
;  1705          2231  1 !       none
;  1706          2232  1 !
;  1707          2233  1 !--
;  1708          2234  1
;  1709          2235  2     BEGIN
;  1710          2236  2
;  1711          2237  2     MAP
;  1712          2238  2         output_fab      : REF BLOCK [, BYTE];           ! Output file FAB block
;  1713          2239  2
;  1714          2240  2     BIND
;  1715          2241  2         output_nam      =                               ! output NAM block address
;  1716          2242  2                 .output_fab [fab$l_nam]         : BLOCK [, BYTE],
;  1717          2243  2         output_xabfhc   =                               ! output XAB file header characteristics block
;  1718          2244  2                 .output_fab [fab$l_xab]         : BLOCK [, BYTE],
;  1719          2245  2         output_xaball   =                               ! output XAB date block
;  1720          2246  2                 .output_xabfhc [xab$l_nxt]      : BLOCK [, BYTE],
;  1721          2247  2         output_xabdat   =                               ! output XAB date block
;  1722          2248  2                 .output_xaball [xab$l_nxt]      : BLOCK [, BYTE],
;  1723          2249  2         output_xabrdt   =                               ! output XAB date block
;  1724          2250  2                 .output_xabdat [xab$l_nxt]      : BLOCK [, BYTE],
;  1725          2251  2         output_xabpro   =                               ! output XAB date block
;  1726          2252  2                 .output_xabrdt [xab$l_nxt]      : BLOCK [, BYTE];
;  1727          2253  2
;  1728          2254  2 !
;  1729          2255  2 ! Apply the effects of the output file qualifiers to the appropriate XAB blocks.
;  1730          2256  2 !
;  1731          2257  2
;  1732          2258  2     ! /ALLOCATION = n
;  1733          2259  2     !
;  1734          2260  3     IF qualifier_active( alloc_qual, loc_alloc_qual, neg_alloc_qual )
;  1735          2261  2     THEN
;  1736          2262  2         output_xaball [xab$l_alq] = .curr_allocation_value;
;  1737          2263  2
;  1738          2264  3     IF qualifier_active( contig_qual, loc_contig_qual, neg_contig_qual )
;  1739          2265  2     THEN
;  1740          2266  3         BEGIN
;  1741          2267  3         output_xaball [xab$v_ctg] = TRUE;
;  1742          2268  3         output_xaball [xab$v_cbt] = FALSE;
```

```
;  1743          2269  3          END
;  1744          2270  2      ELSE
;  1745          2271  3          BEGIN
;  1746          2272  3          IF .contig_negated OR .neg_contig_qual
;  1747          2273  3          THEN
;  1748          2274  4              BEGIN
;  1749          2275  4              output_xaball [xab$v_ctg] = FALSE;
;  1750          2276  4              output_xaball [xab$v_cbt] = FALSE;
;  1751          2277  3              END;
;  1752          2278  2          END;
;  1753          2279  2
;  1754          2280  3      IF qualifier_active( extend_qual, loc_extend_qual, neg_extend_qual )
;  1755          2281  2      THEN
;  1756          2282  2          output_xaball [xab$w_deq] = .curr_extension_value;
;  1757          2283  2
;  1758          2284  3      IF qualifier_active( file_max_qual, loc_file_max_qual, neg_file_max_qual )
;  1759          2285  2      THEN
;  1760          2286  2          output_fab [fab$l_mrn] = .curr_file_max_value;
;  1761          2287  2
;  1762          2288  2      IF qualifier_active( overlay_qual, loc_overlay_qual, neg_overlay_qual ) OR
;  1763          2289  2          .new_version_qual
;  1764          2290  2      THEN
;  1765          2291  2          output_fab [fab$v_cif] = TRUE;
;  1766          2292  2
;  1767          2293  3      IF qualifier_active( replace_qual, loc_replace_qual, neg_replace_qual )
;  1768          2294  2      THEN
;  1769          2295  2          output_fab [fab$v_sup] = TRUE;
;  1770          2296  2
;  1771          2297  3      IF qualifier_active( truncate_qual, loc_truncate_qual, neg_truncate_qual )
;  1772          2298  2      THEN
;  1773          2299  2          output_fab [fab$v_tef] = TRUE,
;  1774          2300  2
;  1775          2301  3      IF qualifier_active( volume_qual, loc_volume_qual, neg_volume_qual )
;  1776          2302  2      THEN
;  1777          2303  3          BEGIN
;  1778          2304  3          output_xaball [xab$w_vol] = .curr_volume_value;
;  1779          2305  3          output_xaball [xab$b_aln] = xab$c_lbn;
;  1780          2306  3          output_xaball [xab$v_hrd] = 1;
;  1781          2307  2          END;
;  1782          2308  2
;  1783          2309  3      IF qualifier_active( write_chk_qual, loc_write_chk_qual, neg_write_chk_qual )
;  1784          2310  2      THEN
;  1785          2311  2          output_fab [fab$v_wck] = TRUE
;  1786          2312  2      ELSE
;  1787          2313  3          BEGIN
;  1788          2314  3          IF .write_chk_negated
;  1789          2315  3          THEN
;  1790          2316  3              output_fab [fab$v_wck] = FALSE;
;  1791          2317  2          END;
;  1792          2318  2
;  1793          2319  2  !
;  1794          2320  2  ! Return to caller.
;  1795          2321  2  !
;  1796          2322  2
;  1797          2323  1      END;                                        ! Return without a value.
```

```
                                00(  00000 APPLY_OUT_QUAL:
                                                .WORD   Save R2,R3                              ; 2155
                        53      0000G   CF   9E 00002          MOVAB   COPY$CLI_STATUS+4, R3
                        51        04    AC   D0 00007          MOVL    OUTPUT_FAB, R1          ; 2242
                        50        24    A1   D0 0000B          MOVL    36(R1), R0              ; 2244
                        50        04    A0   D0 0000F          MOVL    4(R0), R0               ; 2246
                        52        04    A0   D0 00013          MOVL    4(R0), R2               ; 2248
                        05        FE    A3   E9 00017          BLBC    COPY$CLI_STATUS+2, 1$   ; 2260
                05      FE      A3            02 E1 0001B      BBC     #2, COPY$CLI_STATUS+2, 2$
                06      FE      A3            01 E1 00020 1$:  BBC     #1, COPY$CLI_STATUS+2, 3$
                        10      A0    0000G   CF   D0 00025 2$:  MOVL    CURR_ALLOCATION_VALUE, 16(R0)   ; 2262
                05      FE      A3            03 E1 0002B 3$:  BBC     #3, COPY$CLI_STATUS+2, 4$       ; 2264
                05      FE      A3            06 E1 00030      BBC     #6, COPY$CLI_STATUS+2, 5$
                07      FE      A3            05 E1 00035 4$:  BBC     #5, COPY$CLI_STATUS+2, 6$
                        08      A0      80    8F   88 0003A 5$:  BISB2   #128, 8(R0)              ; 2267
                                              0F   11 0003F      BRB     8$                      ; 2268
                05      FE      A3            04 E0 00041 6$:  BBS     #4, COPY$CLI_STATUS+2, 7$       ; 2272
                09      FE      A3            06 E1 00046      BBC     #6, COPY$CLI_STATUS+2, 9$
                        08      A0      80    8F   8A 0004B 7$:  BICB2   #128, 8(R0)              ; 2275
                        08      A0            20   8A 00050 8$:  BICB2   #32, 8(R0)               ; 2276
                                        FE    A3   95 00054 9$:  TSTB    COPY$CLI_STATUS+2       ; 2280
                                              05   18 00057      BGEQ    10$
                04      FF      A3            01 E1 00059      BBC     #1, COPY$CLI_STATUS+3, 11$
                                06      FF    A3   E9 0005E 10$: BLBC    COPY$CLI_STATUS+3, 12$
                        14      A0    0000G   CF   B0 00062 11$: MOVW    CURR_EXTENSION_VALUE, 20(R0)    ; 2282
                05      FF      A3            02 E1 00068 12$: BBC     #2, COPY$CLI_STATUS+3, 13$      ; 2284
                05      FF      A3            04 E1 0006D      BBC     #4, COPY$CLI_STATUS+3, 14$
                06      FF      A3            03 E1 00072 13$: BBC     #3, COPY$CLI_STATUS+3, 15$
                        38      A1    0000G   CF   D0 00077 14$: MOVL    CURR_FILE_MAX_VALUE, 56(R1)     ; 2286
                                              63   95 0007D 15$: TSTB    COPY$CLI_STATUS+4       ; 2288
                                              05   18 0007F      BGEQ    16$
                09      01      A3            01 E1 00081      BBC     #1, COPY$CLI_STATUS+5, 17$
                                05      01    A3   E8 00086 16$: BLBS    COPY$CLI_STATUS+5, 17$
                04      FC      A3            04 E1 0008A      BBC     #4, COPY$CLI_STATUS, 18$        ; 2289
                        07      A1            02 88 0008F 17$: BISB2   #2, 7(R1)               ; 2291
                05      02      A3            01 E1 00093 18$: BBC     #1, COPY$CLI_STATUS+6, 19$      ; 2293
                05      02      A3            03 E1 00098      BBC     #3, COPY$CLI_STATUS+6, 20$
                04      02      A3            02 E1 0009D 19$: BBC     #2, COPY$CLI_STATUS+6, 21$
                        04      A1            04 88 000A2 20$: BISB2   #4, 4(R1)               ; 2295
                04      01      A3            05 E1 000A6 21$: BBC     #5, COPY$CLI_STATUS+5, 22$      ; 2297
                                05      02    A3   E9 000AB      BLBC    COPY$CLI_STATUS+6, 23$
                                        01    A3   95 000AF 22$: TSTB    COPY$CLI_STATUS+5
                                              04   18 000B2      BGEQ    24$
                        07      A1            10 88 000B4 23$: BISB2   #16, 7(R1)              ; 2299
                05      01      A3            02 E1 000B8 24$: BBC     #2, COPY$CLI_STATUS+5, 25$      ; 2301
                05      01      A3            04 E1 000BD      BBC     #4, COPY$CLI_STATUS+5, 26$
                0E      01      A3            03 E1 000C2 25$: BBC     #3, COPY$CLI_STATUS+5, 27$
                        0A      A0    0000G   CF   B0 000C7 26$: MOVW    CURR_VOLUME_VALUE, 10(R0)       ; 2304
                        09      A0            02 90 000CD      MOVB    #2, 9(R0)               ; 2305
                        08      A0            01 88 000D1      BISB2   #1, 8(R0)               ; 2306
                04              63            03 E1 000D5 27$: BBC     #3, COPY$CLI_STATUS+4, 28$      ; 2309
                04              63            06 E1 000D9      BBC     #6, COPY$CLI_STATUS+4, 29$
                05              63            05 E1 000DD 28$: BBC     #5, COPY$CLI_STATUS+4, 30$
                        05      A1            02 88 000E1 29$: BISB2   #2, 5(R1)               ; 2311
```

```
                                       04 000E5          RET                                          ;
          04          63            04 E1 000E6 30$:     BBC     #4, COPY$CLI_STATUS+4, 31$             ; 2314
                05    A1            02 8A 000EA          BICB2   #2, 5(R1)                              ; 2316
                                       04 000EE 31$:     RET                                          ; 2323

; Routine Size:  239 bytes,    Routine Base:  $CODE$ + 065D
```

```
; 1799          2324  1 END
; 1800          2325  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL, LIB$STOP

;                                  PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $PLIT$                         36  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                       1868  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]STARLET.L32;1      9776       179         1       581          00:01.0


;                                   COMMAND QUALIFIERS

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

; Size:         1868 code + 36 data bytes
; Run Time:       00:51.0
; Elapsed Time:   02:08.1
; Lines/CPU Min:   2735
; Lexemes/CPU-Min: 28632
; Memory Used:  286 pages
; Compilation Complete
```

COPYMSG
REQ

COPYSPECS
LIS

COPY

COPY
MAP

VMSMAC
REQ

COPY
REQ

COPYCLI
LIS

COPYMAIN
LIS

COPYSEMAN
LIS

CRFOR
LIS

CRFMEM
LIS

CRFMACROS
MAR

CLITABDEF
SDL

FILINPUT
LIS

KEYS
LIS

CRFTFRVEC
LIS

DCLDEF
MDL

CRFMDL
MDL

CREF
LIS

CRFERRMSG
LIS

CRF
SDL

CRF

CRFSUB
LIS

FILOUTPUT
LIS

CRFSHR
MAP

CRFGLOBAL
LIS

CRFMAC
REQ

DCL

DCL
MAP