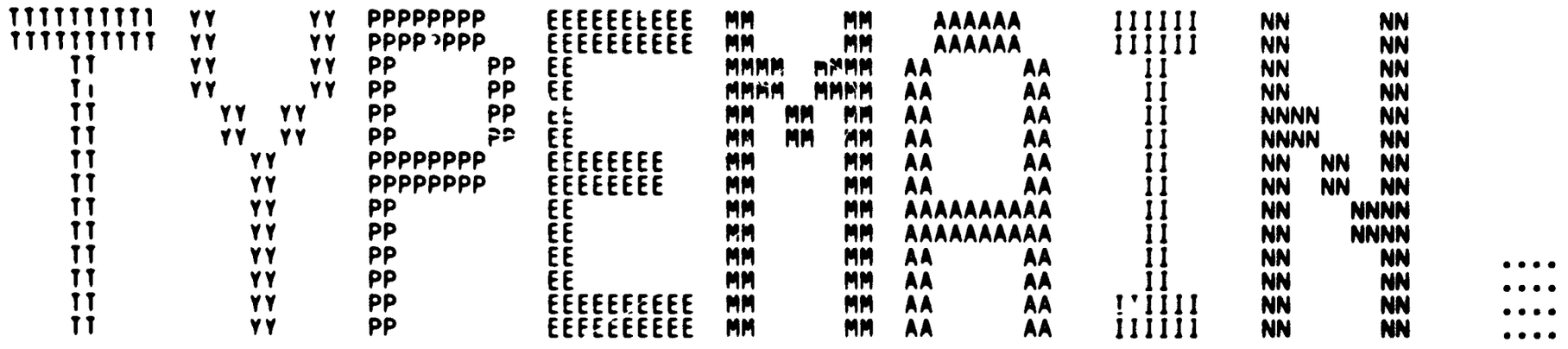
```
   CCCCCCCCCCCC  LLL                  IIIIIIIII     UUU         UUU  TTTTTTTTTTTTTTT  LLL
   CCCCCCCCCCCC  LLL                  IIIIIIIII     UUU         UUU  TTTTTTTTTTTTTTT  LLL
   CCCCCCCCCCCC  LLL                  IIIIIIIII     UUU         UUU  TTTTTTTTTTTTTTT  LLL
CCC              LLL                     III        UUU         UUU        TTT        LLL
CCC              LLL                     III        UUU         UUU        TTT        LLL
CCC              LLL                     III        UUU         UUU        TTT        LLL
CCC              LLL                     III        UUU         UUU        TTT        LLL
CCC              LLL                     III        UUU         UUU        TTT        LLL
CCC              LLL                     III        UUU         UUU        TTT        LLL
CCC              LLL                     III        UUU         UUU        TTT        LLL
CCC              LLL                     III        UUU         UUU        TTT        LLL
CCC              LLL                     III        UUU         UUU        TTT        LLL
CCC              LLL                     III        UUU         UUU        TTT        LLL
CCC              LLL                     III        UUU         UUU        TTT        LLL
CCC              LLL                     III        UUU         UUU        TTT        LLL
CCC              LLL                     III        UUU         UUU        TTT        LLL
CCC              LLL                     III        UUU         UUU        TTT        LLL
CCC              LLL                     III        UUU         UUU        TTT        LLL
   CCCCCCCCCCCC  LLLLLLLLLLLLLLL      IIIIIIIII     UUUUUUUUUUUUUUU        TTT        LLLLLLLLLLLLLLL
   CCCCCCCCCCCC  LLLLLLLLLLLLLLL      IIIIIIIII     UUUUUUUUUUUUUUU        TTT        LLLLLLLLLLLLLLL
   CCCCCCCCCCCC  LLLLLLLLLLLLLLL      IIIIIIIII     UUUUUUUUUUUUUUU        TTT        LLLLLLLLLLLLLLL
```


TTTTTTTTTT YY YY PPPPPPPP EEEEEEEEEE MM MM AAAAAA IIIIII NN NN
TTTTTTTTTT YY YY PPPP PPP EEEEEEEEEE MM MM AAAAAA IIIIII NN NN
TT YY YY PP PP EE MMMM MMMM AA AA II NN NN
TT YY YY PP PP EE MMMM MMMM AA AA II NN NN
TT YY YY PP PP EE MM MM MM AA AA II NNNN NN
TT YY YY PP PP EE MM MM MM AA AA II NNNN NN
TT YY PPPPPPPP EEEEEEEE MM MM AA AA II NN NN NN
TT YY PPPPPPPP EEEEEEEE MM MM AA AA II NN NN NN
TT YY PP EE MM MM AAAAAAAAAA II NN NNNN
TT YY PP EE MM MM AAAAAAAAAA II NN NNNN
TT YY PP EE MM MM AA AA II NN NN ....
TT YY PP EE MM MM AA AA II NN NN ....
TT YY PP EEEEEEEEEE MM MM AA AA IIIIII NN NN ....
TT YY PP EEEEEEEEEE MM MM AA AA IIIIII NN NN ....
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 type_main (       ! STARLET Native Type Utility
;     2          0002  0                           IDENT = 'V04-000',
;     3          0003  0                           MAIN = type$main,
;     4          0004  0                           ADDRESSING_MODE(EXTERNAL=GENERAL)
;     5          0005  0                           ) =
;     6          0006  1 BEGIN
;     7          0007  1
;     8          0008  1 !
;     9          0009  1 !*****************************************************************************
;    10          0010  1 !*                                                                           *
;    11          0011  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    12          0012  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    13          0013  1 !*  ALL RIGHTS RESERVED.                                                      *
;    14          0014  1 !*                                                                           *
;    15          0015  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    16          0016  1 !*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
;    17          0017  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
;    18          0018  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    19          0019  1 !*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY         *
;    20          0020  1 !*  TRANSFERRED.                                                              *
;    21          0021  1 !*                                                                           *
;    22          0022  1 !*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE      *
;    23          0023  1 !*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT          *
;    24          0024  1 !*  CORPORATION.                                                              *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
;    27          0027  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    28          0028  1 !*                                                                           *
;    29          0029  1 !*                                                                           *
;    30          0030  1 !*****************************************************************************
;    31          0031  1
;    32          0032  1 !++
;    33          0033  1 ! FACILITY:  TYPE
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !
;    37          0037  1 !       This utility program types one or more user-specified 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 ! AUTHOR:  Stephen H. Zalewski,         CREATION DATE:  10-Aug-1982
;    45          0045  1 !
;    46          0046  1 ! Modified by:
;    47          0047  1 !
;    48          0048  1 !       V03-011 SHZ0011         Stephen Zalewski        26-Jul-1984
;    49          0049  1 !               Use seperate descriptor when parsing output qualifier.
;    50          0050  1 !
;    51          0051  1 !       V03-010 SHZ0010         Stephen H. Zalewski,    20-Jul-1984
;    52          0052  1 !               Fix type/page for .log files.  Make /OUTPUT
;    53          0053  1 !               default to NL:
;    54          0054  1 !
;    55          0055  1 !       V03-009 ROP0010         Robert Posniak          19-JUN-1984
;    56          0056  1 !               Only find image symbols for screen formating when
;    57          0057  1 !               in page mode.
```

```
;    58          0058  1 !
;    59          0059  1 !       V03-008 SHZ0008         Stephen H. Zalewski,    12-Apr-1984
;    60          0060  1 !               Make TYPE/OUTPUT default to TYPE.LIS instead of SYS$OUTPUT.
;    61          0061  1 !
;    62          0062  1 !       V03-007 SHZ0007         Stephen H. Zalewski,    07-Mar-1984
;    63          0063  1 !               Add /PAGE support for non-vt100 type terminals.
;    64          0064  1 !
;    65          0065  1 !       V03-006 SHZ0006         Stephen H. Zalewski     28-Feb-1984
;    66          0066  1 !               Do not continue reading records from input file if a
;    67          0067  1 !               $GET fails.  The only exception is if rms$_rtb is returned.
;    68          0068  1 !
;    69          0069  1 !       V03-005 SHZ0005         Stephen H. Zalewski     16-Feb-1984
;    70          0070  1 !               Add support for /PAGE.  Add support for sticky searchlists.
;    71          0071  1 !
;    72          0072  1 !       V03-004 SHZ0004         Stephen H. Zalewski     01-Jan-1984
;    73          0073  1 !               Fix line spacing problem when printing headers for
;    74          0074  1 !               files with no RAT attributes.  Do not allow a user to
;    75          0075  1 !               specify SYS$OUTPUT as the input file.
;    76          0076  1 !
;    77          0077  1 !       V03-003 SHZ0003         Stephen H. Zalewski     31-Oct-1983
;    78          0078  1 !               Do not update related file name pointer in related file
;    79          0079  1 !               name block.  Also no longer bother doing a $PARSE for each
;    80          0080  1 !               file spec.  Both of these functions are now done in
;    81          0081  1 !               lib$file_scan.
;    82          0082  1 !
;    83          0083  1 !       V03-002 SHZ0002         Stephen H. Zalewski     20-Apr-1983
;    84          0084  1 !               Add functionality to allow a user to type CNTRL/C,
;    85          0085  1 !               which causes type to immediately go to next file spec.
;    86          0086  1 !
;    87          0087  1 !       V03-001 SHZ0001         Stephen H. Zalewski     5-Apr-1983
;    88          0088  1 !               Set default Recored attribute to be CR on open of a file.
;    89          0089  1 !               This is needed if either we are getting input from TT: or
;    90          0090  1 !               the user did a "task=foo".
;    91          0091  1 !
;    92          0092  1 !--
```

```
;    94     0093  1 LIBRARY 'SYS$LIBRARY:STARLET.L32';                    ! VAX/VMS common definitions
;    95     0094  1 REQUIRE 'SRC$:TYPE.REQ';                              ! COPY literal definitions and macros
;    96     0177  1
;    97     0178  1
;    98     0179  1 FORWARD ROUTINE
;    99     0180  1     type$main,                                        ! Main TYPE control routine
;   100     0181  1     type_file :         NOVALUE,                      ! Type one file
;   101     0182  1     write_header,                                     ! Write a file header to output file
;   102     0183  1     write_line,                                       ! Write a line to the output file
;   103     0184  1     ast_routine :       NOVALUE,                      ! Ast routine called by CNTRL/C
;   104     0185  1     type$get_cmdqual :  NOVALUE,                      ! Get command qualifiers.
;   105     0186  1     get_term_info:      NOVALUE,                      ! Determine terminal characteristics
;   106     0187  1     type$open_output,                                 ! Open the output file.
;   107     0188  1     type$search_error:  NOVALUE,                      ! Error searching for a file.
;   108     0189  1     condit_handler,                                   ! Condition handler for errors and messages
;   109     0190  1     type$file_error :   NOVALUE;                      ! Handles errors on RMS file functions
;   110     0191  1
;   111     0192  1
;   112     0193  1 EXTERNAL ROUTINE
;   113     0194  1     lib$find_image_symbol,
;   114     0195  1     lib$get_command,
;   115     0196  1     lib$qual_file_parse,                              ! Get common command qualifiers
;   116     0197  1      ib$qual_file_match,                              ! Check to see if a file should be typed.
;   117     0198  1     lib$get_vm,                                       ! Get some address space
;   118     0199  1     cli$get_value,                                    ! Get qualifier value
;   119     0200  1     cli$present,                                      ! Check for qualifiers
;   120     0201  1     lib$file_scan;                                    ! Search wildcard specifications.
;   121     0202  1
;   122     0203  1
;   123     0204  1 EXTERNAL
;   124     0205  1     cli$_negated,
;   125     0206  1     lib$_quipro,                                      ! Quit processing
;   126     0207  1     lib$_filfaimat;                                   ! File failed to meet criteria
;   127     0208  1
;   128     0209  1
;   129     0210  1 GLOBAL
;   130     0211  1     channel : VECTOR[1,WORD],
;   131     0212  1     top_line,
;   132     0213  1     bottom_line,
;   133     0214  1     bottom_scroll
;   134     0215  1     bottom_of_page,
;   135     0216  1     line_width,
;   136     0217  1     middle_of_line,
;   137     0218  1     screen_flags        : INITIAL(2),
;   138     0219  1     type$context,                                     ! Context longword for common qualifier package
;   139     0220  1     type$gen_flags      : $BBLOCK [4] INITIAL(0),     ! General flags
;   140     0221  1     type$exit_status    : $BBLOCK [4]                 ! Holds most severe error code
;   141     0222  1                           INITIAL (ss$_normal),
;   142     0223  1     type$input_esa      : VECTOR [nam$c_maxrss, BYTE], ! Expanded string address for input filename
;   143     0224  1     type$input_rsa      : VECTOR [nam$c_maxrss, BYTE], ! Related string address for input filename.
;   144     0225  1     type$output_rsa     : VECTOR [nam$c_maxrss, BYTE], ! Related output address for output file name.
;   145     0226  1     rh_buffer           : VECTOR [type$c_rhb_size, BYTE],! Address of record header fubber.
;   146     0227  1     line_buffer,                                      ! Address of line buffer.
;   147     0228  1     line_counter,                                     ! Used if we paginate file.
;   148     0229  1     input_desc          : $BBLOCK [dsc$c_s_bln],      ! CLI input file descriptor block
;   149     0230  1     output_desc         : $BBLOCK [dsc$c_s_bln],      ! CLI output file descriptor block
;   150     0231  1     command_desc        : $BBLOCK [dsc$c_s_bln],      ! Used to get a command if we are paginating
```

```
;  151          0232  1      input_fab           : $FAB_DECL,                    ! Input FAB.
;  152          0233  1      input_rab           : $RAB_DECL,                    ! Input RAB.
;  153          0234  1      input_nam           : $NAM_DECL,                    ! Input NAM.
;  154          0235  1      output_fab          : $FAB_DECL,                    ! Output FAB.
;  155          0236  1      output_rab          : $RAB_DECL,                    ! Output RAB.
;  156          0237  1      output_nam          : $NAM_DECL;                    ! Output NAM.
;  157          0238  1
;  158          0239  1
;  159        M 0240  1  MACRO LBRSYM(LBRNAME) =
;  160        M 0241  1      PSECT OWN = $OWN$;
;  161        M 0242  1
;  162        M 0243  1      OWN %NAME('LBR_',LBRNAME) : LONG;
;  163        M 0244  1
;  164        M 0245  1      PSECT OWN = LBR_ADDRESSES (NOWRITE,EXECUTE);
;  165        M 0246  1
;  166        M 0247  1      OWN %NAME(LBRNAME,'_ADDR'): INITIAL(%NAME('LBR_',LBRNAME));
;  167        M 0248  1
;  168        M 0249  1      PSECT OWN = LBR_NAMES (NOWRITE,EXECUTE);
;  169        M 0250  1
;  170        M 0251  1      OWN %NAME(LBRNAME) : VECTOR[2,LONG]
;  171        M 0252  1                  INITIAL(%CHARCOUNT(%NAME('LIB$',LBRNAME)),
;  172        M 0253  1                          UPLIT(%STRING('LIB$',%NAME(LBRNAME))));
;  173        M 0254  1
;  174          0255  1      UNDECLARE %NAME(LBRNAME)%;
;  175          0256  1
;  176          0257  1
;  177          0258  1
;  178          0259  1  PSECT OWN = LBR_ADDRESSES(NOWRITE,EXECUTE);
;  179          0260  1
;  180          0261  1  OWN
;  181          0262  1      LBR_ADDR_HEAD : VECTOR[0,LONG];
;  182          0263  1
;  183          0264  1  PSECT OWN = LBR_NAMES(NOWRITE,EXECUTE);
;  184          0265  1
;  185          0266  1  OWN
;  186          0267  1      LBR_NAMES_HEAD : VECTOR[0,LONG];
;  187          0268  1
;  188          0269  1  LBRSYM(ERASE_PAGE);
;  189          0270  1  LBRSYM(ERASE_LINE);
;  190          0271  1  LBRSYM(PUT_SCREEN);
;  191          0272  1  LBRSYM(SCREEN_INFO);
;  192          0273  1  LBRSYM(SET_CURSOR);
;  193          0274  1  LBRSYM(SET_SCROLL);
;  194          0275  1  LBRSYM(UP_SCROLL);
;  195          0276  1
;  196          0277  1  PSECT OWN = LBR_ADDRESSES;
;  197          0278  1
;  198          0279  1  OWN
;  199          0280  1      LBR_ADDR_END : LONG INITIAL(0);
;  200          0281  1
;  201          0282  1  PSECT OWN = $OWN$;
;  202          0283  1
```

```
;  204          0284  1 ROUTINE type$main =
;  205          0285  1
;  206          0286  1 !++
;  207          0287  1 ! Functional description
;  208          0288  1 !
;  209          0289  1 !       This routine is the central control routine for the TYPE utility.
;  210          0290  1 !       It determines the basic logical flow and calls support routines
;  211          0291  1 !       that perform each logical function in typing files.
;  212          0292  1 !
;  213          0293  1 ! Calling sequence
;  214          0294  1 !
;  215          0295  1 !       type$main ()
;  216          0296  1 !
;  217          0297  1 ! Input parameters
;  218          0298  1 !
;  219          0299  1 !       none
;  220          0300  1 !
;  221          0301  1 ! Implicit inputs
;  222          0302  1 !
;  223          0303  1 !       none
;  224          0304  1 !
;  225          0305  1 ! Output parameters
;  226          0306  1 !
;  227          0307  1 !       none
;  228          0308  1 !
;  229          0309  1 ! Implicit outputs
;  230          0310  1 !
;  231          0311  1 !       type$exit_status        - set whenever an error occurs
;  232          0312  1 !
;  233          0313  1 ! Routine value
;  234          0314  1 !
;  235          0315  1 !       Most severe error encountered during processing or SS$_NORMAL
;  236          0316  1 !
;  237          0317  1 ! Side effects
;  238          0318  1 !
;  239          0319  1 !       The specified files are typed.
;  240          0320  1 !
;  241          0321  1 !--
;  242          0322  1
;  243          0323  2 BEGIN
;  244          0324  2
;  245          0325  2 LOCAL
;  246          0326  2     scan_context : INITIAL(0),
;  247          0327  2     status;                                    ! General routine return code
;  248          0328  2
;  249          0329  2 ENABLE condit_handler;                         ! Enable a local handler.
;  250          0330  2
;  251          0331  2 CH$FILL(0, dsc$c_d_bln, input_desc);           ! Make descriptor dynamic
;  252          0332  2 input_desc[dsc$b_class] = dsc$k_class_d;
;  253          0333  2 CH$MOVE(dsc$c_d_bln,input_desc,command_desc);
;  254          0334  2 CH$MOVE(dsc$c_d_bln,input_desc,output_desc);
;  255          0335  2
;  256          0336  2 status = lib$get_vm(UPLIT(32767),line_buffer);  ! Get space for line buffer.
;  257          0337  2 IF NOT .status
;  258          0338  2     THEN SIGNAL_STOP (.status);
;  259          0339  2
;  260        P 0340  2 $FAB_INIT (FAB = input_fab,                    ! Initialize the input FAB
```

```
                CTX = input_rab,                          ! Store address of input RAB.
                RAT = CR,                                 ! Set record attr. in case TT input
                FAC = <GET>,                              ! Input file.
                SHR = <UPI,GET,PUT>,                      ! Allow others to read, write to the inout file.
                DNA = UPLIT BYTE ('.LIS'),                ! Default file type is .LIS
                DNS = 4,                                  ! Size is 4.
                FOP = <NAM,SQO>,                          ! Open by name block.
                NAM = input_nam);                         ! NAM block address.

$RAB_INIT (RAB = input_rab,                               ! Initialize input RAB
                RAC = SEQ,                                ! Record access is sequential.
                ROP = LOC,                                ! Record options
                USZ = 32767,                              ! Input buffer size
                UBF = .line_buffer,                       ! Input buffer address.
                RHB = rh_buffer,                          ! Record header buffer address
                FAB = input_fab);                         ! Address of input FAB.

$NAM_INIT (NAM = input_nam,                               ! Initialize NAM block.
                ESS = nam$c_maxrss,                       ! Expanded string size
                ESA = type$input_esa,                     ! Expanded string address
                RSS = nam$c_maxrss,                       ! Related string size
                RSA = type$input_rsa);                    ! Related string address


type$get_cmdqual();                                       ! Get the command qualifiers.
get_term_info();


!
! The following loop is executed for each file specification supplied by the user.
!

    WHILE true DO                                         ! Beginning of repeat loop
        BEGIN
        IF NOT cli$get_value($descriptor('INPUT'), input_desc)  ! If no more file specs are coming,
            THEN EXITLOOP;
        input_fab[fab$l_fna] = .input_desc [dsc$a_pointer];     ! Move the file name address
        input_fab[fab$b_fns] = .input_desc [dsc$w_length];      !  and length into the file FAB block.
        lib$file_scan(input_fab,                          ! Type every file specified in the
                      type_file,                          !  file spec just found.
                      type$search_error,                  ! Go here if file is not found.
                      scan_context);
        END;                                              ! End of 'WHILE true DO'' file spec processing loop.


IF .output_fab[fab$w_ifi] NEQ 0                           ! If the output file is open, close it.
THEN
    BEGIN
    status = $CLOSE(fab = output_fab);                    ! Close the output file.
    IF NOT .status                                        ! Report an error is we fail.
    THEN
        type$file_error(msg$_closeout,output_fab);
    END;


RETURN .type$exit_status OR sts$m_inhib_msg;              ! EXIT with no message
END;
```

```
                                .TITLE  TYPE_MAIN
                                .IDENT  \V04-000\

                                .PSECT  LBR_NAMES,NOWRT,2

                00000 LBR_NAMES_HEAD:
                                .BLKB   0
0000000E        00000 ERASE_PAGE:
                                .LONG   14
00000000'       00004           .ADDRESS P.AAA
0000000E        00008 ERASE_LINE:
                                .LONG   14
00000000'       0000C           .ADDRESS P.AAB
0000000E        00010 PUT_SCREEN:
                                .LONG   14
00000000'       00014           .ADDRESS P.AAC
0000000F        00018 SCREEN_INFO:
                                .LONG   15
00000000'       0001C           .ADDRESS P.AAD
0000000E        00020 SET_CURSOR:
                                .LONG   14
00000000'       00024           .ADDRESS P.AAE
0000000E        00028 SET_SCROLL:
                                .LONG   14
00000000'       0002C           .ADDRESS P.AAF
0000000D        00030 UP_SCROLL:
                                .LONG   13
00000000'       00034           .ADDRESS P.AAG

                                .PSECT  LBR_ADDRESSES,NOWRT,2

                00000 LBR_ADDR_HEAD:
                                .BLKB   0
00000000'       00000 ERASE_PAGE_ADDR:
                                .ADDRESS LBR_ERASE_PAGE
00000000'       00004 ERASE_LINE_ADDR:
                                .ADDRESS LBR_ERASE_LINE
00000000'       00008 PUT_SCREEN_ADDR:
                                .ADDRESS LBR_PUT_SCREEN
00000000'       0000C SCREEN_INFO_ADDR:
                                .ADDRESS LBR_SCREEN_INFO
00000000'       00010 SET_CURSOR_ADDR:
                                .ADDRESS LBR_SET_CURSOR
00000000'       00014 SET_SCROLL_ADDR:
                                .ADDRESS LBR_SET_SCROLL
00000000'       00018 UP_SCROLL_ADDR:
                                .ADDRESS LBR_UP_SCROLL
00000000        0001C LBR_ADDR_END:
                                .LONG   0

                                .PSECT  $PLIT$,NOWRT,NOEXE,2

00 45 47 41 50 5F 45 53 41 52 45 24 42 49 4C  00000 P.AAA:   .ASCII  \LIB$ERASE_PAGE\<0><0>
                                              00 0000F
00 45 4E 49 4C 5F 45 53 41 52 45 24 42 49 4C  00010 P.AAB:   .ASCII  \LIB$ERASE_LINE\<0><0>
```

```
                                                      00  0001F
00 4E 45 45 52 43 53 5F 54 55 50 24 42 49 4C  00020 P.AAC:  .ASCII  \LIB$PUT_SCREEN\<0><0>
                                                      00  0002F
4F 46 4E 49 5F 4E 45 45 52 43 53 24 42 49 4C  00030 P.AAD:  .ASCII  \LIB$SCREEN_INFO\<0>
                                                      00  0003F
00 52 4F 53 52 55 43 5F 54 45 53 24 42 49 4C  00040 P.AAE:  .ASCII  \LIB$SET_CURSOR\<0><0>
                                                      00  0004F
00 4C 4C 4F 52 43 53 5F 54 45 53 24 42 49 4C  00050 P.AAF:  .ASCII  \LIB$SET_SCROLL\<0><0>
                                                      00  0005F
00 00 4C 4C 4F 52 43 53 5F 50 55 24 42 49 4C  00060 P.AAG:  .ASCII  \LIB$UP_SCROLL\<0><0><0>
                                                      00  0006F
                                                00007FFF  00070 P.AAH:  .LONG   32767
                                           53 49 4C 2E  00074 P.AAI:  .ASCII  \.LIS\
                                        54 55 50 4E 49  00078 P.AAK:  .ASCII  \INPUT\
                                                          0007D         .BLKB   3
                                                00000005  00080 P.AAJ:  .LONG   5
                                                00000000' 00084         .ADDRESS P.AAK

                                                                        .PSECT  $OWN$,NOEXE,2

                                                          00000 LBR_ERASE_PAGE:
                                                                        .BLKB   4
                                                          00004 LBR_ERASE_LINE:
                                                                        .BLKB   4
                                                          00008 LBR_PUT_SCREEN:
                                                                        .BLKB   4
                                                          0000C LBR_SCREEN_INFO:
                                                                        .BLKB   4
                                                          00010 LBR_SET_CURSOR:
                                                                        .BLKB   4
                                                          00014 LBR_SET_SCROLL:
                                                                        .BLKB   4
                                                          00018 LBR_UP_SCROLL:
                                                                        .BLKB   4

                                                                        .PSECT  $GLOBAL$,NOEXE,2

                                                          00000 CHANNEL::
                                                                        .BLKB   2
                                                          00002         .BLKB   2
                                                          00004 TOP_LINE::
                                                                        .BLKB   4
                                                          00008 BOTTOM_LINE::
                                                                        .BLKB   4
                                                          0000C BOTTOM_SCROLL::
                                                                        .BLKB   4
                                                          00010 BOTTOM_OF_PAGE::
                                                                        .BLKB   4
                                                          00014 LINE_WIDTH::
                                                                        .BLKB   4
                                                          00018 MIDDLE_OF_LINE::
                                                                        .BLKB   4
                                                00000002  0001C SCREEN_FLAGS::
                                                                        .LONG   2
                                                          00020 TYPE$CONTEXT::
                                                                        .BLKB   4
                                                00000000  00024 TYPE$GEN_FLAGS::
```

```
                                        .LONG   0                                       ;
00000001  00028 TYPE$EXIT_STATUS::
                                        .LONG   1                                       ;
                0002C TYPE$INPUT_ESA::
                                        .BLKB   255
                0012B                   .BLKB   1
                0012C TYPE$INPUT_RSA::
                                        .BLKB   255
                0022B                   .BLKB   1
                0022C TYPE$OUTPUT_RSA::
                                        .BLKB   255
                0032B                   .BLKB   1
                0032C RH_BUFFER::
                                        .BLKB   255
                0042B                   .BLKB   1
                0042C LINE_BUFFER::
                                        .BLKB   4
                00430 LINE_COUNTER::
                                        .BLKB   4
                00434 INPUT_DESC::
                                        .BLKB   8
                0043C OUTPUT_DESC::
                                        .BLKB   8
                00444 COMMAND_DESC::
                                        .BLKB   8
                0044C INPUT_FAB::
                                        .BLKB   80
                0049C INPUT_RAB::
                                        .BLKB   68
                004E0 INPUT_NAM::
                                        .BLKB   96
                00540 OUTPUT_FAB::
                                        .BLKB   80
                00590 OUTPUT_RAB::
                                        .BLKB   68
                005D4 OUTPUT_NAM::
                                        .BLKB   96

                                $RMS_PTR=       INPUT_FAB
                                $RMS_PTR=       INPUT_RAB
                                $RMS_PTR=       INPUT_NAM
                                        .EXTRN  LIB$FIND_IMAGE_SYMBOL
                                        .EXTRN  LIB$GET_COMMAND
                                        .EXTRN  LIB$QUAL_FILE_PARSE
                                        .EXTRN  LIB$QUAL_FILE_MATCH
                                        .EXTRN  LIB$GET_VM, CLI$GET_VALUE
                                        .EXTRN  CLI$PRESENT, LIB$FIXE_SCAN
                                        .EXTRN  CLI$_NEGATED, LIB$_QUIPRO
                                        .EXTRN  LIB$_FILFAIMAT, SYS$CLOSE

                                        .PSECT  $CODE$,NOWRT,2

                        00FC 00000 TYPE$MAIN:
                                        .WORD   Save R2,R3,R4,R5,R6,R7                  ; 0284
    57      0000'   CF  9E 00002        MOVAB   INPUT_DESC, R7                          ;
                    7E  D4 00007        CLRL    SCAN_CONTEXT                            ; 0323
    6D      0132    CF  DE 00009        MOVAL   5$, (FP)                                ;
```

```
                08              00              6E              00  2C 0000E          MOVC5   #0, (SP), #0, #8, INPUT_DESC                ; 0331
                                                                67     00013
                                        03      A7              02  90 00014          MOVB    #2, INPUT_DESC+3                            ; 0332
                                10      A7              67      08  28 00018          MOVC3   #8, INPUT_DESC, COMMAND_DESC                ; 0333
                                08      A7              67      08  28 0001D          MOVC3   #8, INPUT_DESC, OUTPUT_DESC                 ; 0334
                                                                F8  A7  9F 00022          PUSHAB  LINE_BUFFER                         ; 0336
                                                        0000'   CF  9F 00025          PUSHAB  P.AAR
                                    00000000G   00              02  FB 00029          CALLS   #2, LIB$GET_VM
                                                56              50  D0 00030          MOVL    R0, STATUS
                                                09              56  E8 00033          BLBS    STATUS, 1$                              ; 0337
                                                                56  DD 00036          PUSHL   STATUS                              ; 0338
                                    00000000G   00              01  FB 00038          CALLS   #1, LIB$STOP
        0050    8F              00              6E              00  2C 0003F 1$:      MOVC5   #0, (SP), #0, #80, $RMS_PTR             ; 0348
                                                        18      A7     00046
                                            18      A7      5003    8F  B0 00048          MOVW    #20483, $RMS_PTR
                                            1C      A7  01000040    8F  D0 0004E          MOVL    #16777280, $RMS_PTR+4
                                            2E      A7      4302    8F  B0 00056          MOVW    #17154, $RMS_PTR+22
                                            30      A7        68    A7  9E 0005C          MOVAB   INPUT_RAB, $RMS_PTR+24
                                            36      A7      0202    8F  B0 00061          MOVW    #514, $RMS_PTR+30
                                            40      A7      00AC    C7  9E 00067          MOVAB   INPUT_NAM, $RMS_PTR+40
                                            48      A7      0000'   CF  9E 0006D          MOVAB   P.AAI, $RMS_PTR+48
                                            4D      A7              04  90 00073          MOVB    #4, $RMS_PTR+53
        0044    8F              00              6E              00  2C 00077          MOVC5   #0, (SP), #0, #68, $RMS_PTR             ; 0356
                                                        68      A7     0007E
                                            68      A7      4401    8F  B0 00080          MOVW    #17409, $RMS_PTR
                                            6C      A7  00010000    8F  D0 00086          MOVL    #65536, $RMS_PTR+4
                                                    0086    C7      94  0008E          CLRB    $RMS_PTR+30
                                        0088    C7      7FFF    8F  B0 00092          MOVW    #32767, $RMS_PTR+32
                                        008C    C7        F8    A7  D0 00099          MOVL    LINE_BUFFER, $RMS_PTR+36
                                        0094    C7      FEF8    C7  9E 0009F          MOVAB   RH_BUFFER, $RMS_PTR+44
                                        00A4    C7        18    A7  9E 000A6          MOVAB   INPUT_FAB, $RMS_PTR+60
        0060    8F                  00              6E              00  2C 000AC          MOVC5   #0, (SP), #0, #96, $RMS_PTR             ; 0362
                                                    00AC    C7     000B3
                                        00AC    C7      6002    8F  B0 000B6          MOVW    #24578, $RMS_PTR
                                        00AE    C7              01  8E 000BD          MNEGB   #1, $RMS_PTR+2
                                        00B0    C7      FCF8    C7  9E 000C2          MOVAB   TYPE$INPUT_RSA, $RMS_PTR+4
                                        00B6    C7              01  8E 000C9          MNEGB   #1, $RMS_PTR+10
                                        00B8    C7      FBF8    C7  9E 000CE          MOVAB   TYPE$INPUT_ESA, $RMS_PTR+12
                                        0000V   CF              00  FB 000D5          CALLS   #0, TYPE$GET_CMDQUAL                    ; 0365
                                        0000V   CF              00  FB 000DA          CALLS   #0, GET_TERM_INFO                       ; 0366
                                                                57  DD 000DF 2$:      PUSHL   R7                                      ; 0375
                                                        0000'   CF  9F 000E1          PUSHAB  P.AAJ
                                    00000000G   00              02  FB 000E5          CALLS   #2, CLI$GET_VALUE
                                                1F              50  E9 000EC          BLBC    R0, 3$
                                            44      A7        04    A7  D0 000EF          MOVL    INPUT_DESC+4, INPUT_FAB+44              ; 0377
                                            4C      A7              67  90 000F4          MOVB    INPUT_DESC, INPUT_FAB+52                ; 0378
                                                                5E  DD 000F8          PUSHL   SP                                      ; 0379
                                                        0000V   CF  9F 000FA          PUSHAB  TYPE$SEARCH_ERROR
                                                        00C0V   CF  9F 000FE          PUSHAB  TYPE_FILE
                                                          18    A7  9F 00102          PUSHAB  INPUT_FAB
                                    00000000G   00              04  FB 00105          CALLS   #4, LIB$FILE_SCAN
                                                                D1  11 0010C          BRB     2$                                      ; 0373
                                                    010E    C7  B5 0010E 3$:      TSTW    OUTPUT_FAB+2                            ; 0386
                                                                20  13 00112          BEQL    4$
                                                    010C    C7  9F 00114          PUSHAB  OUTPUT_FAB                              ; 0389
                                    00000000G   00              01  FB 00118          CALLS   #1, SYS$CLOSE
                                                56              50  D0 0011F          MOVL    R0, STATUS
```

```
                      0F                 56  E8 00122        BLBS    STATUS, 4$                        ; 0390
                              010C       C7  9F 00125        PUSHAB  OUTPUT_FAB                        ; 0392
                          00951058       8F  DD 00129        PUSHL   #9769048                          ;
              0000V   CF                 02  FB 0012F        CALLS   #2, TYPE$FILE_ERROR               ;
     50       FBF4    C7  10000000       8F  C9 00134 4$:    BISL3   #268435456, TYPE$EXIT_STATUS, R0  ; 0396
                                             04 0013E        RET                                       ; 0397
                                           0000 0013F 5$:    .WORD   Save nothing                      ; 0323
                                         7E  D4 00141        CLRL    -(SP)                             ;
                                         5E  DD 00143        PUSHL   SP                                ;
                      7E          04     AC  7D 00145        MOVQ    4(AP), -(SP)                      ;
              0000V   CF                 03  FB 00149        CALLS   #3, CONDIT_HANDLER                ;
                                             04 0014E        RET                                       ;

; Routine Size:  335 bytes,    Routine Base:  $CODE$ + 0000


;  318          0398  1
```

```
;  320          0399  1 ROUTINE type_file : NOVALUE =
;  321          0400  1
;  322          0401  1 !++
;  323          0402  1 ! Functional description
;  324          0403  1 !
;  325          0404  1 !       This routine types one file.  If the output file is SYS$OUTPUT, it is closed after we
;  326          0405  1 !       finish writing the input file.  The reason for this is that there could have been several files
;  327          0406  1 !       given on the input line, and they could each have different record characteristics.  By closing
;  328          0407  1 !       the output file each time, we make sure that we pick up the correct record charactersitics for
;  329          0408  1 !       each file when we type it.  This is not possible when the output file is directed to something
;  330          0409  1 !       other that sys$output because we would create a new version of the output file every time we
;  331          0410  1 !       opened and closed. it.
;  332          0411  1 !
;  333          0412  1 !
;  334          0413  1 ! Calling sequence
;  335          0414  1 !
;  336          0415  1 !       type_file
;  337          0416  1 !
;  338          0417  1 ! Input parameters
;  339          0418  1 !
;  340          0419  1 !       none
;  341          0420  1 !
;  342          0421  1 ! Implicit inputs
;  343          0422  1 !
;  344          0423  1 !       none
;  345          0424  1 !
;  346          0425  1 ! Output parameters
;  347          0426  1 !
;  348          0427  1 !       none
;  349          0428  1 !
;  350          0429  1 ! Implicit outputs
;  351          0430  1 !
;  352          0431  1 !       none
;  353          0432  1 !
;  354          0433  1 ! Routine value
;  355          0434  1 !
;  356          0435  1 !       novalue
;  357          0436  1 !
;  358          0437  1 !--
;  359          0438  2 BEGIN
;  360          0439  2
;  361          0440  2 LOCAL
;  362          0441  2     name_desc : VECTOR[2],                          ! Descriptor for file name being typed
;  363          0442  2     prompt_desc,                                    ! Descriptor to hold prompt arguments.
;  364          0443  2     status;                                         ! Holds RMS status codes.
;  365          0444  2
;  366          0445  2 prompt_desc = name_desc;                            ! Point prompt descriptor to name descriptor.
;  367          0446  2 name_desc[0] = .input_nam[nam$b_rsl];                ! Load name desc with related string for file.
;  368          0447  2 name_desc[1] = .input_nam[nam$l_rsa];
;  369          0448  2 status = lib$qual_file_match( type$context,         ! Context pointer.
;  370          0449  2                               input_fab,             ! Fab Pointer.
;  371          0450  2                               0,                     ! No file name.
;  372          0451  2                               $descriptor('!AS, type? [N]:'),! Prompt string.
;  373          0452  2                               prompt_desc,           ! Prompt arguments.
;  374          0453  2                               0);                    ! No prompt routine.
;  375          0454  2
;  376          0455  2 IF NOT .status                                        ! If error status returned
```

```
;  377     0456  2 THEN                                                    .
;  378     0457  3     BEGIN
;  379     0458  3     IF (.status NEQ lib$_filfaimat) AND                 ! and the error was not "file failed to meet criteri
;  380     0459  4        (.status NEQ lib$_quipro)                        ! or "quit processing"
;  381     0460  3     THEN                                                ! Then report an error.
;  382     0461  3         type$file_error(msg$_filnotacc,input_fab);
;  383     0462  3     RETURN true;                                        ! Skip this file.
;  384     0463  2     END;
;  385     0464  2
;  386     0465  2 IF (NOT $OPEN(FAB=input_fab)) OR                        ! If we cannot OPEN
;  387     0466  3    (NOT $CONNECT(RAB=input_rab))                        ! or CONNECT to this file
;  388     0467  2 THEN                                                    ! then
;  389     0468  3     BEGIN                                               ! Report an error and skip this file.
;  390     0469  3     type$file_error(msg$_openin,input_fab);
;  391     0470  3     RETURN false;
;  392     0471  2     END;
;  393     0472  2
;  394     0473  2 IF .output_fab[fab$w_ifi] EQL 0                         ! If output file is not open
;  395     0474  2     THEN type$open_output();                            !  then go open and connect to it.
;  396     0475  2
;  397     0476  2
;  398     0477  2 IF .type$gen_flags[type$v_paginate]
;  399     0478  2 THEN
;  400     0479  3     BEGIN
;  401     0480  3     IF .type$gen_flags[type$v_eof]
;  402     0481  3     THEN
;  403     0482  4         BEGIN
;  404     0483  4         (.LBR_PUT_SCREEN)($descriptor('Press RETURN to continue'),bottom_of_page,middle_of_line,screen_flags
;  405     0484  4         lib$get_command(command_desc);
;  406     0485  4         type$gen_flags[type$v_eof] = false;
;  407     0486  3         END;
;  408     0487  3     (.LBR_ERASE_PAGE)(top_line,top_line);
;  409     0488  3     IF .output_fab[fab$v_cr]
;  410     0489  3     THEN
;  411     0490  4         BEGIN
;  412     0491  4         bottom_scroll = .bottom_of_page - 2;
;  413     0492  4         line_counter = 0;
;  414     0493  4         END
;  415     0494  3     ELSE
;  416     0495  4         BEGIN
;  417     0496  4         bottom_scroll = .bottom_of_page - 1;
;  418     0497  4         line_counter = -1;
;  419     0498  3         END;
;  420     0499  3     (.LBR_SET_SCROLL)(top_line,bottom_scroll);
;  421     0500  2     END;
;  422     0501  2
;  423     0502  2
;  424     0503  2 IF .type$gen_flags[type$v_write_header] OR              ! If write header flag is set
;  425     0504  2    .input_nam[nam$v_wildcard]                           !  or there are wildcards in the input file spec
;  426     0505  2 THEN                                                    ! then write the file name to the output file.
;  427     0506  2     write_header();
;  428     0507  2
;  429     0508  2 type$gen_flags[type$v_write_header] = TRUE;             ! set write header flag to true so we print the next
;  430     0509  2
;  431     0510  2
;  432     0511  2 WHILE true DO                                           ! Until we see "end of file" DO
;  433     0512  3     BEGIN
```

```
;  434          0513  3     IF .type$gen_flags[type$v_controlc]                    ! IF user says CNTRL/C
;  435          0514  3     THEN                                                   ! then
;  436          0515  4         BEGIN
;  437          0516  4         type$gen_flags[type$v_controlc] = false;           ! reset the flag
;  438          0517  4         EXITLOOP;                                          ! and go to next file.
;  439          0518  3         END;
;  440          0519  3     status = $GET(rab = input_rab);                        ! Get a record from the input file.
;  441          0520  3     IF .status EQL rms$_eof                                ! If we are at eof then exit loop.
;  442          0521  3     THEN
;  443          0522  4         BEGIN
;  444          0523  4         type$gen_flags[type$v_eof] = true;
;  445          0524  4         EXITLOOP;
;  446          0525  3         END;
;  447          0526  3     IF NOT .status                                         ! We have an error, so report it.
;  448          0527  3     THEN
;  449          0528  4         BEGIN
;  450          0529  4         type$file_error(msg$_readerr,input_fab);
;  451          0530  4         IF .status NEQ rms$_rtb                            ! Ignore record to big error.
;  452          0531  4             THEN EXITLOOP;
;  453          0532  3         END;
;  454          0533  3     output_rab[rab$w_rsz] = .input_rab[rab$w_rsz];         ! Read succeeded.
;  455          0534  3     output_rab[rab$l_rbf] = .input_rab[rab$l_rbf];         !  write it to the output file
;  456          0535  3     status = write_line();
;  457          0536  3     IF NOT .status
;  458          0537  3         THEN EXITLOOP;
;  459          0538  2     END;
;  460          0539  2
;  461          0540  2 status = $CLOSE(fab = input_fab);                         ! All done with the input file
   462          0541  2 IF NOT .status                                            !  so close it.
;  463          0542  2 THEN
;  464          0543  2     type$file_error(msg$_closein,input_fab);               ! Report an error is CLOSE fails.
;  465          0544  2
;  466          0545  2 IF .type$gen_flags[type$v_sysoutput]                      ! If we are writing to sys$output
;  467          0546  2 THEN                                                     !  then close the output file to.
;  468          0547  3     BEGIN
;  469          0548  3     status = $CLOSE(fab = output_fab);                     ! Report an error if CLOSE fails.
;  470          0549  3     IF NOT .status
;  471          0550  3     THEN
;  472          0551  3         type$file_error(msg$_closeout,output_fab);
;  473          0552  2     END;
;  474          0553  2
;  475          0554  2
;  476          0555  2 RETURN .status;
;  477          0556  1 END;


                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

3A 5D 4E 5B 20 3F 65 70 79 74 20 2C 53 41 21  00088 P.AAM:  .ASCII  \!AS, type? [N]:\                    ;
                                              00097         .BLKB   1
                                   0000000F   00098 P.AAL:  .LONG   15                                  ;
                                   00000000'  0009C         .ADDRESS P.AAM                             ;
6F 74 20 4E 52 55 54 45 52 20 73 73 65 72 50  000A0 P.AAO:  .ASCII  \Press RETURN to continue\          ;
                  65 75 6E 69 74 6E 6F 63 20  000AF
                                   00000018   000B8 P.AAN:  .LONG   24                                  ;
                                   00000000'  000BC         .ADDRESS P.AAO                             ;
```

```
                                        .EXTRN  SYS$OPEN, SYS$CONNECT
                                        .EXTRN  SYS$GET

                                        .PSECT  $CODE$,NOWRT,2

                                003C 00000 TYPE_FILE:
                                        .WORD   Save R2,R3,R4,R5                ; 0399
                    55      0000V   CF  9E 00002        MOVAB   TYPE$FILE_ERROR, R5             ;
                    54  00000000G   00  9E 00007        MOVAB   SYS$CLOSE, R4                   ;
                    53      0000'   CF  9E 0000E        MOVAB   TYPE$GEN_FLAGS, R3              ;
                    5E              08  C2 00013        SUBL2   #8, SP                          ;
                                    5E  DD 00016        PUSHL   SP                              ; 0445
                04  AE        04BF   C3  9A 00018        MOVZBL  INPUT_NAM+3, NAME_DESC          ; 0446
                08  AE        04C0   C3  D0 0001E        MOVL    INPUT_NAM+4, NAME_DESC+4        ; 0447
                                    7E  D4 00024        CLRL    -(SP)                           ; 0448
                                  04    AE  9F 00026        PUSHAB  PROMPT_DESC                     ;
                                0000'   CF  9F 00029        PUSHAB  P.AAL                           ; 0451
                                    7E  D4 0002D        CLRL    -(SP)                           ; 0448
                                0428    C3  9F 0002F        PUSHAB  INPUT_FAB                       ;
                                  FC    A3  9F 00033        PUSHAB  TYPE$CONTEXT                    ;
00000000G       00                  06  FB 00036        CALLS   #6, LIB$QUAL_FILE_MATCH         ;
                52                  50  D0 0003D        MOVL    R0, STATUS                      ;
                25                  52  E8 00040        BLBS    STATUS, 3$                      ; 0455
                50  00000000G      00  9E 00043        MOVAB   LIB$_FILFAIMAT, R0              ; 0458
                50                  52  D1 0004A        CMPL    STATUS, R0                      ;
                                    0A  13 0004D        BEQL    1$                              ;
                50  00000000G      00  9E 0004F        MOVAB   LIB$_QUIPRO, R0                 ; 0459
                50                  52  D1 00056        CMPL    STATUS, R0                      ;
                                    01  12 00059 1$:     BNEQ    2$                              ;
                                        04 0005B        RET                                     ;
                                0428    C3  9F 0005C 2$:    PUSHAB  INPUT_FAB                   ; 0461
                            00951338    8F  DD 00060        PUSHL   #9769784                        ;
                                        26  11 00066        BRB     5$                          ;
                                0428    C3  9F 00068 3$:    PUSHAB  INPUT_FAB                   ; 0465
        00000000G       00              01  FB 0006C        CALLS   #1, SYS$OPEN                    ;
                        0E              50  E9 00073        BLBC    R0, 4$                          ;
                                0478    C3  9F 00076        PUSHAB  INPUT_RAB                   ; 0466
        00000000G       00              01  FB 0007A        CALLS   #1, SYS$CONNECT                 ;
                        0D              50  E8 00081        BLBS    R0, 6$                          ;
                                0428    C3  9F 00084 4$:    PUSHAB  INPUT_FAB                   ; 0469
                            00951098    8F  DD 00088        PUSHL   #9769112                        ;
                                    0105    31  0008E 5$:    BRW     20$                             ;
                                051E    C3  B5 00091 6$:    TSTW    OUTPUT_FAB+2                ; 0473
                                        05  12 00095        BNEQ    7$                          ;
                    0000V       CF          00  FB 00097        CALLS   #0, TYPE$OPEN_OUTPUT        ; 0474
58                              63          03  E1 0009C 7$:    BBC     #3, TYPE$GEN_FLAGS, 11$     ; 0477
21                              63          06  E1 000A0        BBC     #6, TYPE$GEN_FLAGS, 8$      ; 0480
                                  F8    A3  9F 000A4        PUSHAB  SCREEN_FLAGS                    ; 0483
                                  F4    A3  9F 000A7        PUSHAB  MIDDLE_OF_LINE                  ;
                                  EC    A3  9F 000AA        PUSHAB  BOTTOM_OF_PAGE                  ;
                                0000'   CF  9F 000AD        PUSHAB  P.AAN                           ;
                    0000'       DF          04  FB 000B1        CALLS   #4, @LBR_PUT_SCREEN         ;
                                0420    C3  9F 000B6        PUSHAB  COMMAND_DESC                    ; 0484
        00000000G       00              01  FB 000BA        CALLS   #1, LIB$GET_COMMAND             ;
                                63      40  8F 8A 000C1        BICB2   #64, TYPE$GEN_FLAGS         ; 0485
                                        E0  A3  9F 000C5 8$:    PUSHAB  TOP_LINE                    ; 0487
```

```
                                  E0  A3  9F 000C8        PUSHAB  TOP_LINE                                  ;
                 0000'  DF                02  FB 000CB        CALLS   #2, @LBR_ERASE_PAGE                       ;
          0C     053A   C3                01  E1 000D0        BBC     #1, OUTPUT_FAB+30, 9$                     ; 0488
     E8   A3       EC   A3                02  C3 000D6        SUBL3   #2, BOTTOM_OF_PAGE, BOTTOM_SCROLL         ; 0491
                                  040C    C3  D4 000DC        CLRL    LINE_COUNTER                              ; 0492
                                          0B  11 000E0        BRB     10$                                       ; 0488
     E8   A3       EC   A3                01  C3 000E2 9$:    SUBL3   #1, BOTTOM_OF_PAGE, BOTTOM_SCROLL         ; 0496
                 040C   C3                01  CE 000E8        MNEGL   #1, LINE_COUNTER                          ; 0497
                                  E8  A3  9F 000ED 10$:       PUSHAB  BOTTOM_SCROLL                             ; 0499
                                  E0  A3  9F 000F0            PUSHAB  TOP_LINE                                  ;
                 0000'  DF                02  FB 000F3        CALLS   #2, @LBR_SET_SCROLL                       ;
          05            63                01  E0 000F8 11$:   BBS     #1, TYPE$GEN_FLAGS, 12$                   ; 0503
                        05        04F1    C3  E9 000FC        BLBC    INPUT_NAM+53, 13$                         ; 0504
                 0000V  CF                00  FB 00101 12$:   CALLS   #0, WRITE_HEADER                          ; 0506
                        63                02  88 00106 13$:   BISB2   #2, TYPE$GEN_FLAGS                        ; 0508
          05            63                05  E1 00109 14$:   BBC     #5, TYPE$GEN_FLAGS, 15$                   ; 0513
                        63                20  8A 0010D        BICB2   #32, TYPE$GEN_FLAGS                       ; 0516
                                          4F  11 00110        BRB     18$                                       ; 0515
                                  0478    C3  9F 00112 15$:   PUSHAB  INPUT_RAB                                 ; 0519
             00000000G  00                01  FB 00116        CALLS   #1, SYS$GET                               ;
                        52                50  D0 0011D        MOVL    R0, STATUS                                ;
             0001827A   8F                52  D1 00120        CMPL    STATUS, #98938                            ; 0520
                                          06  12 00127        BNEQ    16$                                       ;
                        63        40      8F  88 00129        BISB2   #64, TYPE$GEN_FLAGS                       ; 0523
                                          32  11 0012D        BRB     18$                                       ; 0522
                        16                52  E8 0012F 16$:   BLBS    STATUS, 17$                               ; 0526
                                  0428    C3  9F 00132        PUSHAB  INPUT_FAB                                 ; 0529
                              009510B0    8F  DD 00136        PUSHL   #9769136                                  ;
                        65                02  FB 0013C        CALLS   #2, TYPE$FILE_ERROR                       ;
             000181A8   8F                52  D1 0013F        CMPL    STATUS, #98728                            ; 0530
                                          19  12 00146        BNEQ    18$                                       ;
                 058E   C3        049A    C3  B0 00148 17$:   MOVW    INPUT_RAB+34, OUTPUT_RAB+34               ; 0533
                 0594   C3        04A0    C3  D0 0014F        MOVL    INPUT_RAB+40, OUTPUT_RAB+40               ; 0534
                 0000V  CF                00  FB 00156        CALLS   #0, WRITE_LINE                            ; 0535
                        52                50  D0 0015B        MOVL    R0, STATUS                                ;
                        A8                52  E8 0015E        BLBS    STATUS, 14$                               ; 0536
                                  0428    C3  9F 00161 18$:   PUSHAB  INPUT_FAB                                 ; 0540
                        64                01  FB 00165        CALLS   #1, SYS$CLOSE                             ;
                        52                50  D0 00168        MOVL    R0, STATUS                                ;
                        0D                52  E8 0016B        BLBS    STATUS, 19$                               ; 0541
                                  0428    C3  9F 0016E        PUSHAB  INPUT_FAB                                 ; 0543
                              00951050    8F  DD 00172        PUSHL   #9769040                                  ;
                        65                02  FB 00178        CALLS   #2, TYPE$FILE_ERROR                       ;
          1A            63                02  E1 0017B 19$:   BBC     #2, TYPE$GEN_FLAGS, 21$                   ; 0545
                                  051C    C3  9F 0017F        PUSHAB  OUTPUT_FAB                                ; 0548
                        64                01  FB 00183        CALLS   #1, SYS$CLOSE                             ;
                        52                50  D0 00186        MOVL    R0, STATUS                                ;
                        0D                52  E8 00189        BLBS    STATUS, 21$                               ; 0549
                                  051C    C3  9F 0018C        PUSHAB  OUTPUT_FAB                                ; 0551
                              00951058    8F  DD 00190        PUSHL   #9769048                                  ;
                        65                02  FB 00196 20$:   CALLS   #2, TYPE$FILE_ERROR                       ;
                                              04 00199 21$:   RET                                               ; 0556

; Routine Size:  410 bytes,    Routine Base:  $CODE$ + 014F


;  478          0557  1
```

```
;  480     0558  1 ROUTINE write_header =
;  481     0559  1
;  482     0560  1 !++
;  483     0561  1 ! Functional description
;  484     0562  1 !
;  485     0563  1 !       This routine write the current file specification to the output file.
;  486     0564  1 !
;  487     0565  1 ! Calling sequence
;  488     0566  1 !
;  489     0567  1 !       write_header ()
;  490     0568  1 !
;  491     0569  1 ! Input parameters
;  492     0570  1 !
;  493     0571  1 !       none
;  494     0572  1 !
;  495     0573  1 ! Implicit inputs
;  496     0574  1 !
;  497     0575  1 !       none
;  498     0576  1 !
;  499     0577  1 ! Output parameters
;  500     0578  1 !
;  501     0579  1 !       none
;  502     0580  1 !
;  503     0581  1 ! Implicit outputs
;  504     0582  1 !
;  505     0583  1 !       none
;  506     0584  1 !
;  507     0585  1 ! Routine value
;  508     0586  1 !
;  509     0587  1 !       novalue
;  510     0588  1 !
;  511     0589  1 !--
;  512     0590  2 BEGIN
;  513     0591  2
;  514     0592  2 LOCAL
;  515     0593  2     blank_buffer : INITIAL('    '),
;  516     0594  2     temp_buffer : VECTOR[nam$c_maxrss+1,BYTE];
;  517     0595  2
;  518     0596  2 output_rab[rab$v_cco] = true;                   ! Cancel CNTRL/O at start of each new file.
;  519     0597  2
;  520     0598  2 IF .output_fab[fab$v_ftn]                       ! If FTN format file
;  521     0599  2 THEN                                            ! Then
;  522     0600  3     BEGIN
;  523     0601  3     temp_buffer[0] = %ASCII' ';                 ! Set single space carriage control
;  524     0602  3     output_rab[rab$w_rsz] = 1;
;  525     0603  3     output_rab[rab$l_rbf] = temp_buffer;
;  526     0604  3     write_line();
;  527     0605  3     CH$MOVE(.input_nam[nam$b_rsl],input_nam[nam$l_rsa],temp_buffer[1]);
;  528     0606  3     write_line();
;  529     0607  3     output_rab[rab$w_rsz] = 1;
;  530     0608  3     write_line();
;  531     0609  2     END;
;  532     0610  2
;  533     0611  2 IF .output_fab[fab$v_prn]                       ! If PRN format file
;  534     0612  2 THEN                                            ! Then
;  535     0613  2     rh_buffer = %X'8D01';                       ! Set single space PRN carriage control
;  536     0614  2
```

```
2 IF .output_fab[fab$v_prn] OR                         ! If PRN or CR
2     .output_fab[fab$v_cr]
2 THEN                                                 ! Then
3     BEGIN
3     output_rab[rab$w_rsz] = 1;                       ! Print blank line.
3     output_rab[rab$l_rbf] = blank_buffer;
3     write_line();
3     output_rab[rab$w_rsz] = .input_nam[nam$b_rsl];
3     output_rab[rab$l_rbf] = .input_nam[nam$l_rsa];
3     write_line();
3     output_rab[rab$w_rsz] = 1;                       ! Print blank line.
3     output_rab[rab$l_rbf] = blank_buffer;
3     write_line();
2     END;
2
2 IF NOT .output_fab[fab$v_prn] AND                    ! If no carriage control
2    NOT .output_fab[fab$v_cr] AND
2    NOT .output_fab[fab$v_f+ ]
2 THEN                                                 ! Then
3     BEGIN
3     blank_buffer = %x'0D0A0A';                       ! But <CR><LF><LF> into buffer
3     output_rab[rab$w_rsz] = 3;                       ! Print blank line.
3     output_rab[rab$l_rbf] = blank_buffer;
3     write_line();
3     output_rab[rab$w_rsz] = .input_nam[nam$b_rsl];
3     output_rab[rab$l_rbf] = .input_nam[nam$l_rsa];
3     write_line();
3     output_rab[rab$w_rsz] = 3;                       ! Print blank line.
3     output_rab[rab$l_rbf] = blank_buffer;
3     write_line();
2     END;
2
2 output_rab[rab$v_cco] = false;                       ! Renable CNTRL/O
2
2 RETURN true;
1 END;
```

```
                                00FC 00000 WRITE_HEADER:
                                                .WORD   Save R2,R3,R4,R5,R6,R7                 ; 0558
                57       0000V   CF   9E 00002  MOVAB   WRITE_LINE, R7                          ;
                56       0000'   CF   9E 00007  MOVAB   OUTPUT_RAB+34, R6                       ;
                5E       FF00    CE   9E 0000C  MOVAB   -256(SP), SP                            ;
                     20202020    8F   DD 00011  PUSHL   #538976288                              ; 0590
          E5    A6         80    8F   88 00017  BISB2   #128, OUTPUT_RAB+7                      ; 0596
                24         AC    A6   E9 0001C  BLBC    OUTPUT_FAB+30, 1$                       ; 0598
          04    AE               20   90 00020  MOVB    #32, TEMP_BUFFER                        ; 0601
                66               01   B0 00024  MOVW    #1, OUTPUT_RAB+34                       ; 0602
          06    A6         04    AE   9E 00027  MOVAB   TEMP_BUFFER, OUTPUT_RAB+40              ; 0603
                67               00   FB 0002C  CALLS   #0, WRITE_LINE                          ; 0604
                50       FF31    C6   9A 0002F  MOVZBL  INPUT_NAM+3, R0                         ; 0605
05    AE    FF32    C6               50   28 00034  MOVC3   R0, INPUT_NAM+4, TEMP_BUFFER+1          ;
                67               00   FB 0003B  CALLS   #0, WRITE_LINE                          ; 0606
                66               01   B0 0003E  MOVW    #1, OUTPUT_RAB+34                       ; 0607
```

```
                    67           00 FB 00041        CALLS   #0, WRITE_LINE                          ; 0608
        0C     AC   A6           02 E1 00044 1$:    BBC     #2, OUTPUT_FAB+30, 2$                   ; 0611
             FD7A   C6     8D01  8F 3C 00049        MOVZWL  #36097, RH_BUFFER                       ; 0613
        05     AC   A6           02 E0 00050        BBS     #2, OUTPUT_FAB+30, 3$                   ; 0615
        22     AC   A6           01 E1 00055 2$:    BBC     #1, OUTPUT_FAB+30, 4$                   ; 0616
                    66           01 B0 0005A 3$:    MOVW    #1, OUTPUT_RAB+34                       ; 0619
               06   A6           6E 9E 0005D        MOVAB   BLANK_BUFFER, OUTPUT_RAB+40             ; 0620
                    67           00 FB 00061        CALLS   #0, WRITE_LINE                          ; 0621
                    66     FF31  C6 9B 00064        MOVZBW  INPUT_NAM+3, OUTPUT_RAB+34              ; 0622
               06   A6     FF32  C6 D0 00069        MOVL    INPUT_NAM+4, OUTPUT_RAB+40              ; 0623
                    67           00 FB 0006F        CALLS   #0, WRITE_LINE                          ; 0624
                    66           01 B0 00072        MOVW    #1, OUTPUT_RAB+34                       ; 0625
               06   A6           6E 9E 00075        MOVAB   BLANK_BUFFER, OUTPUT_RAB+40             ; 0626
                    67           00 FB 00079        CALLS   #0, WRITE_LINE                          ; 0627
        32     AC   A6           02 E0 0007C 4$:    BBS     #2, OUTPUT_FAB+30, 5$                   ; 0630
        2D     AC   A6           01 E0 00081        BBS     #1, OUTPUT_FAB+30, 5$                   ; 0631
                    29       AC  A6 E8 00086        BLBS    OUTPUT_FAB+30, 5$                       ; 0632
                    6E 000D0A0A  8F D0 0008A        MOVL    #854538, BLANK_BUFFER                   ; 0635
                    66           03 B0 00091        MOVW    #3, OUTPUT_RAB+34                       ; 0636
               06   A6           6E 9E 00094        MOVAB   BLANK_BUFFER, OUTPUT_RAB+40             ; 0637
                    67           00 FB 00098        CALLS   #0, WRITE_LINE                          ; 0638
                    66     FF31  C6 9B 0009B        MOVZBW  INPUT_NAM+3, OUTPUT_RAB+34              ; 0639
               06   A6     FF32  C6 D0 000A0        MOVL    INPUT_NAM+4, OUTPUT_RAB+40              ; 0640
                    67           00 FB 000A6        CALLS   #0, WRITE_LINE                          ; 0641
                    66           03 B0 000A9        MOVW    #3, OUTPUT_RAB+34                       ; 0642
               06   A6           6E 9E 000AC        MOVAB   BLANK_BUFFER, OUTPUT_RAB+40             ; 0643
                    67           00 FB 000B0        CALLS   #0, WRITE_LINE                          ; 0644
               E5   A6       80  8F 8A 000B3 5$:    BICB2   #128, OUTPUT_RAB+7                      ; 0647
                    50           01 D0 000B8        MOVL    #1, R0                                  ; 0649
                                    04 000BB        RET                                             ; 0650

; Routine Size:  188 bytes,    Routine Base:  $CODE$ + 02E9

;  573          0651  1
```

```
;  575          0652  1 ROUTINE write_line =
;  576          0653  1
;  577          0654  1 !++
;  578          0655  1 ! Functional description
;  579          0656  1 !
;  580          0657  1 !       This routine writes one line of the input file to the output file.  If an
;  581          0658  1 !       error occurs writing to the output file, a severe error is signaled
;  582          0659  1 !       and the program terminates.
;  583          0660  1 !
;  584          0661  1 ! Calling sequence
;  585          0662  1 !
;  586          0663  1 !       write_line ()
;  587          0664  1 !
;  588          0665  1 ! Input parameters
;  589          0666  1 !
;  590          0667  1 !       none
;  591          0668  1 !
;  592          0669  1 ! Implicit inputs
;  593          0670  1 !
;  594          0671  1 !       none
;  595          0672  1 !
;  596          0673  1 ! Output parameters
;  597          0674  1 !
;  598          0675  1 !       none
;  599          0676  1 !
;  600          0677  1 ! Implicit outputs
;  601          0678  1 !
;  602          0679  1 !       none
;  603          0680  1 !
;  604          0681  1 ! Routine value
;  605          0682  1 !
;  606          0683  1 !       novalue
;  607          0684  1 !
;  608          0685  1 !--
;  609          0686  2 BEGIN
;  610          0687  2
;  611          0688  2 LOCAL
;  612          0689  2     form_feed : BYTE INITIAL(26),
;  613          0690  2     offset : INITIAL (0),
;  614          0691  2     desc : VECTOR[2],
;  615          0692  2     status;
;  616          0693  2
;  617          0694  2
;  618          0695  2 IF .type$gen_flags[type$v_paginate]
;  619          0696  2 THEN
;  620          0697  3     BEGIN
;  621          0698  3     LOCAL pointer;
;  622          0699  3     desc[0] = .output_rab[rab$w_rsz];
;  623          0700  3     desc[1] = .output_rab[rab$l_rbf];
;  624          0701  3     IF NOT CH$FAIL(CH$FIND_CH(.desc[0],.desc[1],%CHAR(12)))
;  625          0702  3     THEN
;  626          0703  4         BEGIN
;  627          0704  4         LIB$GET_VM(desc[0],desc[1]);
;  628          0705  4         CH$MOVE(.desc[0],.output_rab[rab$l_rbf],.desc[1]);
;  629          0706  4         WHILE true DO
;  630          0707  5             BEGIN
;  631          0708  5             pointer = CH$FIND_CH(.desc[0],.desc[1],%CHAR(12));
```

```
;  632          0709  5              IF NOT CH$FAIL(.pointer)
;  633          0710  5                  THEN
;  634          0711  6                      BEGIN
;  635          0712  6                      BIND string = .pointer : VECTOR[,BYTE];
;  636          0713  6                      string[0] = .form_feed;
;  637          0714  6                      END
;  638          0715  5                  ELSE EXITLOOP;
;  639          0716  4              END;
;  640          0717  4          output_rab[rab$l_rbf] = .desc[1];
;  641          0718  3          END;
;  642          0719  3      line_counter = .line_counter+(.output_rab[rab$w_rsz]/.line_width)+1;
;  643          0720  3      IF .line_counter GTR .bottom_line
;  644          0721  3      THEN
;  645          0722  4          BEGIN
;  646          0723  4          line_counter = 2+(.output_rab[rab$w_rsz]/.line_width);
;  647          0724  4          (.LBR_PUT_SCREEN)($descriptor('Press RETURN to continue'),bottom_of_page,middle_of_line,screen_flags
;  648          0725  4          status = lib$get_command(command_desc);
;  649          0726  4          IF .type$gen_flags[type$v_control]
;  650          0727  4              THEN RETURN true;
;  651          0728  4          (.LBR_ERASE_LINE)(bottom_of_page,middle_of_line);
;  652          0729  4          IF .command_desc[dsc$w_length] NEQ 0
;  653          0730  4           OR .status EQL rms$_eof
;  654          0731  4              THEN RETURN false;
;  655          0732  4          IF .type$gen_flags[type$v_vt100]
;  656          0733  4          THEN
;  657          0734  5              BEGIN
;  658          0735  5              (.LBR_SET_CURSOR)(bottom_scroll,top_line);
;  659          0736  5              IF .output_fab[fab$v_cr]
;  660          0737  5               OR .output_fab[fab$v_prn]
;  661          0738  5                  THEN (.LBR_UP_SCROLL)();
;  662          0739  5              END
;  663          0740  4          ELSE
;  664          0741  4              (.LBR_ERASE_PAGE)(top_line,top_line);
;  665          0742  3          END;
;  666          0743  2      END;
;  667          0744  2
;  668          0745  2  status = $PUT(rab = output_rab);                    ! Put the buffer to the output file.
;  669          0746  2  IF NOT .status                                      ! If we fail, signal an error.
;  670          0747  2  THEN
;  671          0748  2      type$file_error(msg$_writeerr,output_fab);
;  672          0749  2
;  673          0750  2  RETURN .status
;  674          0751  1  END;


                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

6F  74  20  4E  52  55  54  45  52  20  73  73  65  72  50  000C0 P.AAQ:  .ASCII  \Press RETURN to continue\                ;
                    65  75  6E  69  74  6E  6F  63  20  000CF                                                          ;
                                                00000018  000D8 P.AAP:  .LONG   24                                     ;
                                                00000000' 000DC          .ADDRESS P.AAQ                             ;

                                                                .EXTRN  SYS$PUT

                                                                .PSECT  $CODE$,NOWRT,2
```

```
                                    00FC 00000 WRITE_LINE:
                                               .WORD   Save R2,R3,R4,R5,R6,R7              ; 0652
                    57    0000'  CF  9E 00002          MOVAB   LINE_COUNTER, R7                   ;
                    5E           08  C2 00007          SUBL2   #8, SP                             ;
                    56           1A  90 0000A          MOVB    #26, FORM_FEED                     ; 0686
                                 50  D4 0000D          CLRL    OFFSET                             ;
       03    FBF4   C7           03  E0 0000F          BBS     #3, TYPE$GEN_FLAGS, 1$             ; 0695
                               00DB  31 00015          BRW     10$                                ;
                    6E    0182   C7  3C 00018 1$:      MOVZWL  OUTPUT_RAB+34, DESC                ; 0699
             04     AE    0188   C7  D0 0001D          MOVL    OUTPUT_RAB+40, DESC+4              ; 0700
  04   BE           6E           0C  3A 00023          LOCC    #12, DESC, @DESC+4                 ; 0701
                                 02  12 00028          BNEQ    2$                                 ;
                                 51  D4 0002A          CLRL    R1                                 ;
                                 51  D5 0002C 2$:      TSTL    R1                                 ;
                                 2C  13 0002E          BEQL    6$                                 ;
                          04     AE  9F 00030          PUSHAB  DESC+4                             ; 0704
                          04     AE  9F 00033          PUSHAB  DESC                               ;
       00000000G    00           02  FB 00036          CALLS   #2, LIB$GET_VM                     ;
  04   BE    0188   D7           6E  28 0003D          MOVC3   DESC, @OUTPUT_RAB+40, @DESC+4      ; 0705
  04   BE           6E           0C  3A 00044 3$:      LOCC    #12, DESC, @DESC+4                 ; 0708
                                 02  12 00049          BNEQ    4$                                 ;
                                 51  D4 0004B          CLRL    R1                                 ;
                                 51  D5 0004D 4$:      TSTL    POINTER                            ; 0709
                                 05  13 0004F          BEQL    5$                                 ;
                    61           56  90 00051          MOVB    FORM_FEED, (POINTER)               ; 0713
                                 EE  11 00054          BRB     3$                                 ; 0709
             0188   C7    04     AE  D0 00056 5$:      MOVL    DESC+4, OUTPUT_RAB+40              ; 0717
                    51    0182   C7  3C 0005C 6$:      MOVZWL  OUTPUT_RAB+34, R1                  ; 0719
                    51    FBE4   C7  C6 00061          DIVL2   LINE_WIDTH, R1                     ;
       50           67           51  C1 00066          ADDL3   R1, LINE_COUNTER, R0               ;
                    67    01     A0  9E 0006A          MOVAB   1(R0), LINE_COUNTER                ;
             FBD8   C7           67  D1 0006E          CMPL    LINE_COUNTER, BOTTOM_LINE          ; 0720
                                 7E  15 00073          BLEQ    10$                                ;
                    67    02     A1  9E 00075          MOVAB   2(R1), LINE_COUNTER                ; 0723
                          FBEC   C7  9F 00079          PUSHAB  SCREEN_FLAGS                       ; 0724
                          FBE8   C7  9F 0007D          PUSHAB  MIDDLE_OF_LINE                     ;
                          FBE0   C7  9F 00081          PUSHAB  BOTTOM_OF_PAGE                     ;
                          0000'  CF  9F 00085          PUSHAB  P.AAP                              ;
             0000'  DF           04  FB 00089          CALLS   #4, @LBR_PUT_SCREEN                ;
                          14     A7  9F 0008E          PUSHAB  COMMAND_DESC                       ; 0725
       00000000G    00           01  FB 00091          CALLS   #1, LIB$GET_COMMAND                ;
                    52           50  D0 00098          MOVL    R0, STATUS                         ;
       04    FBF4   C7           05  E1 0009B          BBC     #5, TYPE$GEN_FLAGS, 7$             ; 0726
                    50           01  D0 000A1          MOVL    #1, R0                             ; 0727
                                     04 000A4          RET                                        ;
                          FBE8   C7  9F 000A5 7$:      PUSHAB  MIDDLE_OF_LINE                     ; 0728
                          FBE0   C7  9F 000A9          PUSHAB  BOTTOM_OF_PAGE                     ;
             0000'  DF           02  FB 000AD          CALLS   #2, @LBR_ERASE_LINE                ;
                          14     A7  B5 000B2          TSTW    COMMAND_DESC                       ; 0729
                                 60  12 000B5          BNEQ    12$                                ;
       0001827A     8F           52  D1 000B7          CMPL    STATUS, #98938                     ; 0730
                                 57  13 000BE          BEQL    12$                                ;
       20    FBF4   C7           04  E1 000C0          BBC     #4, TYPE$GEN_FLAGS, 9$             ; 0732
                          FBD4   C7  9F 000C6          PUSHAB  TOP_LINE                           ; 0735
                          FBDC   C7  9F 000CA          PUSHAB  BOTTOM_SCROLL                      ;
             0000'  DF           02  FB 000CE          CALLS   #2, @LBR_SET_CURSOR                ;
       06    012E   C7           01  E0 000D3          BBS     #1, OUTPUT_FAB+30, 8$              ; 0736
```

```
                14      012E  C7              02  E1 000D9          BBC     #2, OUTPUT_FAB+30, 10$          ; 0737
                        0000' DF              00  FB 000DF 8$:      CALLS   #0, @LBR_UP_SCROLL              ; 0738
                                              0D  11 000E4          BRB     10$                             ; 0732
                                      FBD4    C7  9F 000E6 9$:      PUSHAR  TOP_LINE                        ; 0741
                                      FBD4    C7  9F 000EA          PUSHAB  TOP_LINE                        ;
                        0000' DF              02  FB 000EE          CALLS   #2, @LBR_ERASE_PAGE             ;
                                      0160    C7  9F 000F3 10$:     PUSHAB  OUTPUT_RAB                      ; 0745
                    00000000G 00              01  FB 000F7          CALLS   #1, SYS$PUT                     ;
                              52              50  D0 000FE          MOVL    R0, STATUS                      ;
                              0F              52  E8 00101          BLBS    STATUS, 11$                     ; 0746
                                      0110    C7  9F 00104          PUSHAB  OUTPUT_FAB                      ; 0748
                                009510D4      8F  DD 00108          PUSHL   #9769172                        ;
                        0000V CF              02  FB 0010E          CALLS   #2, TYPE$FILE_ERROR             ;
                              50              52  D0 00113 11$:     MOVL    STATUS, R0                      ; 0750
                                              04  00116          RET                                     ;
                                              50  D4 00117 12$:     CLRL    R0                              ; 0751
                                              04  00119          RET                                     ;

; Routine Size:  282 bytes,    Routine Base:  $CODE$ + 03A5
```

```
;  676          0752  1 ROUTINE ast_routine :NOVALUE =
;  677          0753  1
;  678          0754  1 !++
;  679          0755  1 ! Functional descr'iption
;  680          0756  1 !
;  681          0757  1 !  This ast routine is called when a user types CNTRL/C.  Its purpose
;  682          0758  1 !  is to set a flag so that we skip to the next file in the file spec.
;  683          0759  1 !
;  684          0760  1 ! Calling sequence
;  685          0761  1 !
;  686          0762  1 !       write_line ()
;  687          0763  1 !
;  688          0764  1 ! Input parameters
;  689          0765  1 !
;  690          0766  1 !       none
;  691          0767  1 !
;  692          0768  1 ! Implicit inputs
;  693          0769  1 !
;  694          0770  1 !       none
;  695          0771  1 !
;  696          0772  1 ! Output parameters
;  697          0773  1 !
;  698          0774  1 !       none
;  699          0775  1 !
;  700          0776  1 ! Implicit outputs
;  701          0777  1 !
;  702          0778  1 !       none
;  703          0779  1 !
;  704          0780  1 ! Routine value
;  705          0781  1 !
;  706          0782  1 !       novalue
;  707          0783  1 !
;  708          0784  1 !--
;  709          0785  2 BEGIN
;  710          0786  2
;  711          0787  2 LOCAL
;  712          0788  2     status;
;  713          0789  2
;  714          0790  2 type$gen_flags[type$v_controlc] = TRUE;          ! User typed CNTRL/C.
;  715          0791  2
;  716          0792  2 status = $CANCEL (CHAN = .channel);              ! Flush any I/O on queue.
;  717          0793  2 IF NOT .status
;  718          0794  2     THEN SIGNAL(msg$_syserror,.status);
;  719          0795  2
;  720        P 0796  2 status = $QIOW  (CHAN = .channel,                ! Reenable CNTRL/C handler.
;  721        P 0797  2                  FUNC = (IO$_SETMODE OR IO$M_CTRLCAST),
;  722        P 0798  2                  P1   = ast_routine,
;  723          0799  2                  P3   = %X'3');
;  724          0800  2 IF NOT .status
;  725          0801  2     THEN SIGNAL(msg$_syserror,.status);
;  726          0802  2
;  727          0803  2 RETURN;
;  728          0804  1 END;


                                                                .EXTRN  SYS$CANCEL, SYS$QIOW
```

```
                                000C 00000 AST_ROUTINE:
                                           .WORD   Save R2,R3                      ; 0752
                 53 00000000G   00  9E 00002        MOVAB   LIB$SIGNAL, R3
           0000' CF             20  88 00009        BISB2   #32, TYPE$GEN_FLAGS     ; 0790
                 7E      0000'  CF  3C 0000E        MOVZWL  CHANNEL, -(SP)          ; 0792
00000000G        00             01  FB 00013        CALLS   #1, SYS$CANCEL
                 52             50  D0 0001A        MOVL    R0, STATUS
                 0B             52  E8 0001D        BLBS    STATUS, 1$              ; 0793
                                52  DD 00020        PUSHL   STATUS                  ; 0794
                    009511B4    8F  DD 00022        PUSHL   #9769396
                 63             02  FB 00028        CALLS   #2, LIB$SIGNAL
                                7E  7C 0002B 1$:    CLRQ    -(SP)                   ; 0799
                 7E             03  7D 0002D        MOVQ    #3, -(SP)
                                7E  D4 00030        CLRL    -(SP)
                       CB       AF  9F 00032        PUSHAB  AST_ROUTINE
                                7E  7C 00035        CLRQ    -(SP)
                                7E  D4 00037        CLRL    -(SP)
                 7E      0123   8F  3C 00039        MOVZWL  #291, -(SP)
                 7E      0000'  CF  3C 0003E        MOVZWL  CHANNEL, -(SP)
                                7E  D4 00043        CLRL    -(SP)
00000000G        00             0C  FB 00045        CALLS   #12, SYS$QIOW
                 52             50  D0 0004C        MOVL    R0, STATUS
                 0B             52  E8 0004F        BLBS    STATUS, 2$              ; 0800
                                52  DD 00052        PUSHL   STATUS                  ; 0801
                    009511B4    8F  DD 00054        PUSHL   #9769396
                 63             02  FB 0005A        CALLS   #2, LIB$SIGNAL
                                04  0005D 2$:      RET                             ; 0804

; Routine Size:  94 bytes,    Routine Base:  $CODE$ + 04BF

;  729          0805  1
```

```
;  731     0806  1 ROUTINE type$get_cmdqual : NOVALUE =
;  732     0807  1
;  733     0808  1 !++
;  734     0809  1 ! Functional description
;  735     0810  1 !
;  736     0811  1 !       This routine gets the command qualifiers for the command line.
;  737     0812  1 !
;  738     0813  1 ! Calling sequence
;  739     0814  1 !
;  740     0815  1 !       type$get_cmdqual()
;  741     0816  1 !
;  742     0817  1 ! Input parameters
;  743     0818  1 !
;  744     0819  1 !       none
;  745     0820  1 !
;  746     0821  1 ! Implicit inputs
;  747     0822  1 !
;  748     0823  1 !       none
;  749     0824  1 !
;  750     0825  1 ! Output parameters
;  751     0826  1 !
;  752     0827  1 !       none
;  753     0828  1 !
;  754     0829  1 ! Implicit outputs
;  755     0830  1 !
;  756     0831  1 !       none
;  757     0832  1 !
;  758     0833  1 ! Routine value
;  759     0834  1 !
;  760     0835  1 !       novalue
;  761     0836  1 !
;  762     0837  1 !--
;  763     0838  2 BEGIN
;  764     0839  2
;  765     0840  2 LOCAL
;  766     0841  2     bitmap,
;  767     0842  2     lbrindex,
;  768     0843  2     status;
;  769     0844  2
;  770     0845  2 bitmap= lib$m_cqf_confirm  OR  lib$m_cqf_exclude  OR     ! Bitmap of qualifiers
;  771     0846  2         lib$m_cqf_before   OR  lib$m_cqf_since    OR     !  that the common qualifier
;  772     0847  2         lib$m_cqf_created  OR  lib$m_cqf_modified OR     !  package is to check for.
;  773     0848  2         lib$m_cqf_expired  OR  lib$m_cqf_backup   OR
;  774     0849  2         lib$m_cqf_byowner;
;  775     0850  2
;  776     0851  2
;  777     0852  3 IF NOT (status = lib$qual_file_parse(bitmap,type$context))! Get common qualifiers
;  778     0853  2 THEN
;  779     0854  2     SIGNAL_STOP(.status);
;  780     0855  2
;  781     0856  2 type$gen_flags[type$v_paginate] = cli$present($descriptor('PAGE'));
;  782     0857  2
;  783     0858  2 !
;  784     0859  2 ! Find the image symbols only if we need them
;  785     0860  2 !
;  786     0861  2
;  787     0862  2 IF .type$gen_flags[type$v_paginate] THEN
```

```
;  788          0863  3   BEGIN
;  789          0864  3     lbrindex = 0;
;  790          0865  3     WHILE .lbr_addr_head[.lbrindex] NEQ 0
;  791          0866  4     DO BEGIN
;  792          0867  4        status = LIB$FIND_IMAGE_SYMBOL($descriptor('SCRSHR'),
;  793          0868  4                 lbr_names_head[.lbrindex*2],.lbr_addr_head[.lbrindex]);
;  794          0869  4        IF NOT .status
;  795          0870  4            THEN SIGNAL_STOP (.status);
;  796          0871  .        lbrindex = .lbrindex + 1;
;  797          0872  3        END;
;  798          0873  2   END;
;  799          0874  1 END;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                                    45  47  41  50  000E0 P.AAS:  .ASCII  \PAGE\                                    ;
                                      00000004  000E4 P.AAR:  .LONG   4                                           ;
                                      00000000' 000E8          .ADDRESS P.AAS                                     ;
                                52  43  53  52  43  53  000EC P.AAU:  .ASCII  \SCRSHR\                             ;
                                                    000F2          .BLKB   2                                       ;
                                      00000006  000F4 P.AAT:  .LONG   6                                           ;
                                      00000000' 000F8          .ADDRESS P.AAU                                     ;


                                                        .PSECT  $CODE$,NOWRT,2

                                                001C 00000 TYPE$GET_CMDQUAL:
                                                           .WORD   Save R2,R3,R4                                    ; 0806
                                    54 00000000G   00   9E 00002          MOVAB   LIB$STOP, R4                      ;
                                    7E      01FF   8F   3C 00009          MOVZWL  #511, BITMAP                      ; 0848
                                           0000'   CF   9F 0000E          PUSHAB  TYPE$CONTEXT                      ; 0852
                                             04    AE   9F 00012          PUSHAB  BITMAP                            ;
                          00000000G        00         02   FB 00015      CALLS   #2, LIB$QUAL_FILE_PARSE           ;
                                           53         50   D0 0001C      MOVL    R0, STATUS                        ;
                                           05         53   E8 0001F      BLBS    STATUS, 1$                        ;
                                                      53   DD 00022      PUSHL   STATUS                            ; 0854
                                           64         01   FB 00024      CALLS   #1, LIB$STOP                      ;
                                          0000'       CF   9F 00027 1$:  PUSHAB  P.AAR                             ; 0856
                          00000000G        00         01   FB 0002B      CALLS   #1, CLI$PRESENT                   ;
0000'  CF                      01         03         50   F0 00032      INSV    R0, #3, #1, TYPE$GEN_FLAGS        ;
                          2F      0000'    CF         03   E1 00039      BBC     #3, TYPE$GEN_FLAGS, 4$            ; 0862
                                                      52   D4 0003F      CLRL    LBRINDEX                          ; 0864
                                           50  0000'CF42   D0 00041 2$:  MOVL    LBR_ADDR_HEAD[LBRINDFX], R0       ; 0865
                                                      25   13 00047      BEQL    4$                                ;
                                                      50   DD 00049      PUSHL   R0                                ; 0868
                                    50                52   01   78 0004B      ASHL    #1, LBRINDEX, R0             ;
                                                0000'CF40   DF 0004F      PUSHAL  LBR_NAMES_HEAD[R0]                ;
                                                   0000'    CF   9F 00054      PUSHAB  P.AAT                        ; 0867
                          00000000G        00         03   FB 00058      CALLS   #3, LIB$FIND_IMAGE_SYMBOL         ; 0868
                                           53         50   D0 0005F      MOVL    R0, STATUS                        ;
                                           05         53   E8 00062      BLBS    STATUS, 3$                        ; 0869
                                                      53   DD 00065      PUSHL   STATUS                            ; 0870
                                           64         01   FB 00067      CALLS   #1, LIB$STOP                      ;
                                                      52   D6 0006A 3$:  INCL    LBRINDEX                          ; 0871
```

```
                                D3  11 0006C         BRB     2$                      ; 0865
                                    04 0006E 4$:     RET                             ; 0874

; Routine Size:  111 bytes,    Routine Base:  $CODE$ + 051D

;  800          0875  1
```

```
;  802     0876  1 ROUTINE get_term_info : NOVALUE =
;  803     0877  2 BEGIN
;  804     0878  2
;  805     0879  2 !++
;  806     0880  2 ! FUNCTIONAL DESCRIPTION:
;  807     0881  2 !
;  808     0882  2 !       Determine the file characteristics, i.e., page height and line width,
;  809     0883  2 !       of the file specified by the input fab.  Also, check to see if device
;  810     0884  2 !       is suitable for page breaks.
;  811     0885  2 !
;  812     0886  2 ! INPUTS:
;  813     0887  2 !
;  814     0888  2 !       none
;  815     0889  2 !
;  816     0890  2 ! OUTPUTS:
;  817     0891  2 !
;  818     0892  2 !       none
;  819     0893  2 !
;  820     0894  2 ! ROUTINE VALUE:
;  821     0895  2 !
;  822     0896  2 !       Always true.
;  823     0897  2 !
;  824     0898  2 !--
;  825     0899  2 LOCAL
;  826     0900  2     status;
;  827     0901  2
;  828     0902  2 MACRO
;  829     0903  2     ddp$b_pagelen = 3,0,8,0%;                   ! *** Hardwired page length offset ***
;  830     0904  2
;  831     0905  2 LITERAL
;  832     0906  2     getdvilen = 4*12 + 4;
;  833     0907  2
;  834     0908  2 LOCAL
;  835     0909  2     getdvidesc : $BBLOCK [getdvilen],
;  836     0910  2     devnamdesc : $BBLOCK [dsc$c_s_bln],
;  837     0911  2     devbufsiz  : INITIAL(0),
;  838     0912  2     devclass   : INITIAL(0),
;  839     0913  2     devdepend : $BBLOCK [4],
;  840     0914  2     devdepend2: $BBLOCK [4];
;  841     0915  2
;  842     0916  2
;  843     0917  2 !
;  844     0918  2 ! Do a $GETDVI to get the parameters of interest.
;  845     0919  2 !
;  846     0920  2 CH$FILL (0, getdvilen, getdvidesc);                 ! Init the $GETDVI buffer
;  847     0921  2
;  848     0922  2 getdvidesc [0,0,16,0] = 4;                          ! Get the device class
;  849     0923  2 getdvidesc [2,0,16,0] = dvi$_devclass;              !
;  850     0924  2 getdvidesc [4,0,32,0] = devclass;                   !
;  851     0925  2 getdvidesc [8,0,32,0] = getdvidesc [0,0,16,0];      !
;  852     0926  2
;  853     0927  2 getdvidesc [12,0,16,0] = 4;                         ! Get the device dependent chars
;  854     0928  2 getdvidesc [14,0,16,0] = dvi$_devdepend;            !
;  855     0929  2 getdvidesc [16,0,32,0] = devdepend;                 !
;  856     0930  2 getdvidesc [20,0,32,0] = getdvidesc [12,0,16,0];    !
;  857     0931  2
;  858     0932  2 getdvidesc [24,0,16,0] = 4;                         ! Get the device type
```

```
;  859          0933  2 getdvidesc [26,0,16,0] = dvi$_devdepend2;                                    !
;  860          0934  2 getdvidesc [28,0,32,0] = devdepend2;                                        !
;  861          0935  2 getdvidesc [32,0,32,0] = getdvidesc [24,0,16,0];                           !
;  862          0936  2
;  863          0937  2 getdvidesc [36,0,16,0] = 4;                                                ! Get the device buffer size
;  864          0938  2 getdvidesc [38,0,16,0] = dvi$_devbufsiz;                                   !
;  865          0939  2 getdvidesc [40,0,32,0] = devbufsiz;                                       !
;  866          0940  2 getdvidesc [44,0,32,0] = getdvidesc [36,0,16,0];                          !
;  867          0941  2
;  868        P 0942  2 status = $GETDVI (DEVNAM = $descriptor('SYS$COMMAND'),                    ! Get the device characteristics
;  869          0943  2                   ITMLST = getdvidesc);
;  870          0944  2
;  871          0945  2 IF NOT .status
;  872          0946  2 THEN
;  873          0947  3     BEGIN
;  874          0948  3     type$gen_flags[type$v_paginate] = false;
;  875          0949  3     RETURN;
;  876          0950  2     END;
;  877          0951  2
;  878          0952  2 IF .type$gen_flags[type$v_paginate]
;  879          0953  2 THEN
;  880          0954  3     BEGIN
;  881          0955  4     IF (.devclass NEQ dc$_term)                                            ! If output device is not a terminal
;  882          0956  3     THEN
;  883          0957  4         BEGIN
;  884          0958  4         type$gen_flags[type$v_paginate] = false;
;  885          0959  4         END
;  886          0960  3     ELSE
;  887          0961  4         BEGIN
;  888          0962  4         IF .devdepend2[tt2$v_deccrt]
;  889          0963  4             THEN type$gen_flags[type$v_vt100] = true;
;  890          0964  4         top_line = 1;
;  891          0965  4         line_width = MINU (.devbufsiz, 132);                               ! Get the listing width
;  892          0966  4         middle_of_line = (.line_width - 22)/2;                             ! Calculate middle of line
;  893          0967  4         bottom_of_page = .devdepend [ddp$b_pagelen];                       ! Get the listing height
;  894          0968  4         bottom_line = .bottom_of_page - 2;
;  895          0969  3         END;
;  896          0970  2     END;
;  897          0971  2
;  898          0972  2 !
;  899          0973  2 ! Enable CNTRL/C handler if this is a terminal.
;  900          0974  2 !
;  901          0975  2
;  902          0976  3 IF (.devclass EQL dc$_term)                                                ! If output device is a terminal
;  903          0977  2 THEN
;  904          0978  3     BEGIN
;  905        P 0979  3     status = $ASSIGN (DEVNAM = $descriptor('SYS$COMMAND'),                 ! Get a channel
;  906          0980  3                       CHAN   = channel);
;  907          0981  3     IF NOT .status
;  908          0982  3         THEN SIGNAL(msg$_syserror,0,.status);
;  909          0983  3
;  910        P 0984  3     status = $QIOW (CHAN = .channel,                                       ! Enable CNTRL/C handling
;  911        P 0985  3                     FUNC = (IO$_SETMODE OR IO$M_CTRLCAST),
;  912        P 0986  3                     P1   = ast_routine,
;  913          0987  3                     P3   = %X'3');
;  914          0988  3     IF NOT .status
;  915          0989  3         THEN SIGNAL(msg$_syserror,0,.status);
```

```
;  916          0990  2     END;
;  917          0991  2
;  918          0992  1 END;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                44  4E  41  4D  4D  4F  43  24  53  59  53  000FC P.AAW:  .ASCII  \SYS$COMMAND\                          ;
                                                        00107           .BLKB   1                                       ;
                                            0000000B    00108 P.AAV:    .LONG   11                                      ;
                                            00000000'   0010C           .ADDRESS P.AAW                                  ;
                44  4E  41  4D  4D  4F  43  24  53  59  53  00110 P.AAY:  .ASCII  \SYS$COMMAND\                          ;
                                                        0011B           .BLKB   1                                       ;
                                            0000000B    0011C P.AAX:    .LONG   11                                      ;
                                            00000000'   00120           .ADDRESS P.AAY                                  ;

                                                        .EXTRN  SYS$GETDVI, SYS$ASSIGN

                                                        .PSECT  $CODE$,NOWRT,2

                                                00FC 00000 GET_TERM_INFO:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7                                  ; 0876
                                    57 00000000G    00  9E 00002            MOVAB   LIB$SIGNAL, R7                      ;
                                    56      0000'   CF  9E 00009            MOVAB   TYPE$GEN_FLAGS, R6                  ;
                                    5E        B8    AE  9E 0000E            MOVAB   -72(SP), SP                         ;
                                              04    AE  D4 00012            CLRL    DEVBUFSIZ                           ; 0877
                                                    7E  D4 00015            CLRL    DEVCLASS                            ;
                34          00            6E        00  2C 00017            MOVC5   #0, (SP), #0, #52, GETDVIDESC       ; 0920
                                                  18    AE  0001C                                                       ;
                                        18  AE 00040004 8F  D0 0001E            MOVL    #262148, GETDVIDESC                 ; 0922
                                        1C  AE          6E  9E 00026            MOVAB   DEVCLASS, GETDVIDESC+4              ; 0924
                                        20  AE          18  AE 9E 0002A         MOVAB   GETDVIDESC, GETDVIDESC+8            ; 0925
                                        24  AE 000A0004    8F D0 0002F         MOVL    #655364, GETDVIDESC+12              ; 0927
                                        28  AE          0C  AE 9E 00037         MOVAB   DEVDEPEND, GETDVIDESC+16            ; 0929
                                        2C  AE          24  AE 9E 0003C         MOVAB   GETDVIDESC+12, GETDVIDESC+20        ; 0930
                                        30  AE 001C0004    8F D0 00041         MOVL    #1835012, GETDVIDESC+24             ; 0932
                                        34  AE          04  AE 9E 00049         MOVAB   DEVDEPEND2, GETDVIDESC+28           ; 0934
                                        38  AE          30  AE 9E 0004E         MOVAB   GETDVIDESC+24, GETDVIDESC+32        ; 0935
                                        3C  AE 00080004    8F D0 00053         MOVL    #524292, GETDVIDESC+36              ; 0937
                                        40  AE          08  AE 9E 0005B         MOVAB   DEVBUFSIZ, GETDVIDESC+40            ; 0939
                                        44  AE          3C  AE 9E 00060         MOVAB   GETDVIDESC+36, GETDVIDESC+44        ; 0940
                                                        7E  7C 00065            CLRQ    -(SP)                               ; 0943
                                                        7E  7C 00067            CLRQ    -(SP)                               ;
                                                  28    AE  9F 00069            PUSHAB  GETDVIDESC                          ;
                                                0000'   CF  9F 0006C            PUSHAB  P.AAV                               ;
                                                        7E  7C 00070            CLRQ    -(SP)                               ;
                            00000000G   00              08  FB 00072            CALLS   #8, SYS$GETDVI                      ;
                                        52              50  D0 00079            MOVL    R0, STATUS                          ;
                                        04              52  E8 0007C            BLBS    STATUS, 1$                          ; 0945
                                        66              08  8A 0007F            BICB2   #8, TYPE$GEN_FLAGS                  ; 0948
                                                        04  00082              RET                                         ; 0947
                44                      66              03  E1 00083 1$:       BBC     #3, TYPE$GEN_FLAGS, 5$              ; 0952
                            00000042    8F              6E  D1 00087            CMPL    DEVCLASS, #66                       ; 0955
                                                        05  13 0008E            BEQL    2$                                  ;
                                        66              08  8A 00090            BICB2   #8, TYPE$GEN_FLAGS                  ; 0958
                                                        36  11 00093            BRB     5$                                  ; 0955
```

```
             03     07  AE          05 E1 00095 2$:    BBC     #5, DEVDEPEND2+3, 3$              ; 0962
                        66          10 88 0009A        BISB2   #16, TYPE$GEN_FLAGS               ; 0963
                    E0  A6          01 D0 0009D 3$:    MOVL    #1, TOP_LINE                      ; 0964
                        50      08  AE D0 000A1        MOVL    DEVBUFSIZ, R0                     ; 0965
              00000084  8F          50 D1 000A5        CMPL    R0, #132                          ;
                        04          1B 000AC           BLEQU   4$                                ;
                        50      84  8F 9A 000AE        MOVZBL  #132, R0                          ;
                    F0  A6          50 D0 000B2 4$:    MOVL    R0, LINE_WIDTH                    ;
             50     F0  A6          16 C3 000B6        SUBL3   #22, LINE_WIDTH, R0               ; 0966
         F4  A6         50          02 C7 000BB        DIVL3   #2, R0, MIDDLE_OF_LINE            ;
                    EC  A6      0F  AE 9A 000C0        MOVZBL  DEVDEPEND+3, BOTTOM_OF_PAGE       ; 0967
         E4  A6     EC  A6          02 C3 000C5        SUBL3   #2, BOTTOM_OF_PAGE, BOTTOM_LINE   ; 0968
              00000042  8F          6E D1 000CB 5$:    CMPL    DEVCLASS, #66                     ; 0976
                                    57 12 000D2        BNEQ    7$                                ;
                                    7E 7C 000D4        CLRQ    -(SP)                             ; 0980
                                DC  A6 9F 000D6        PUSHAB  CHANNEL                           ;
                             0000'  CF 9F 000D9        PUSHAB  P.AAX                             ;
              00000000G 00          04 FB 000DD        CALLS   #4, SYS$ASSIGN                    ;
                        52          50 D0 000E4        MOVL    R0, STATUS                        ;
                        0D          52 E8 000E7        BLBS    STATUS, 6$                        ; 0981
                                    52 DD 000EA        PUSHL   STATUS                            ; 0982
                                    7E D4 000EC        CLRL    -(SP)                             ;
                          009511B4  8F DD 000EE        PUSHL   #9769396                          ;
                        67          03 FB 000F4        CALLS   #3, LIB$SIGNAL                    ;
                                    7E 7C 000F7 6$:    CLRQ    -(SP)                             ; 0987
                        7E          03 7D 000F9        MOVQ    #3, -(SP)                         ;
                                    7E D4 000FC        CLRL    -(SP)                             ;
                              FE31  CF 9F 000FE        PUSHAB  AST_ROUTINE                       ;
                                    7E 7C 00102        CLRQ    -(SP)                             ;
                                    7E D4 00104        CLRL    -(SP)                             ;
                        7E    0123  8F 3C 00106        MOVZWL  #291, -(SP)                       ;
                        7E      DC  A6 3C 0010B        MOVZWL  CHANNEL, -(SP)                    ;
                                    7E D4 0010F        CLRL    -(SP)                             ;
              00000000G 00          0C FB 00111        CALLS   #12, SYS$QIOW                     ;
                        52          50 D0 00118        MOVL    R0, STATUS                        ;
                        0D          52 E8 0011B        BLBS    STATUS, 7$                        ; 0988
                                    52 DD 0011E        PUSHL   STATUS                            ; 0989
                                    7E D4 00120        CLRL    -(SP)                             ;
                          009511B4  8F DD 00122        PUSHL   #9769396                          ;
                        67          03 FB 00128        CALLS   #3, LIB$SIGNAL                    ;
                                       04 0012B 7$:    RET                                       ; 0992

; Routine Size:  300 bytes,    Routine Base:  $CODE$ + 058C

;  919          0993  1
```

```
;  921          0994  1 ROUTINE type$open_output  =
;  922          0995  1
;  923          0996  1 !++
;  924          0997  1 ! Functional description:
;  925          0998  1 !
;  926          0999  1 !       This routine open the output file.
;  927          1000  1 !
;  928          1001  1 ! Input parameters:
;  929          1002  1 !
;  930          1003  1 !       none
;  931          1004  1 !
;  932          1005  1 ! Implicit inputs:
;  933          1006  1 !
;  934          1007  1 !       none
;  935          1008  1 !
;  936          1009  1 ! Output parameters:
;  937          1010  1 !
;  938          1011  1 !       none
;  939          1012  1 !
;  940          1013  1 ! Implicit outputs:
;  941          1014  1 !
;  942          1015  1 !       none
;  943          1016  1 !
;  944          1017  1 ! Routine value:
;  945          1018  1 !
;  946          1019  1 !       none
;  947          1020  1 !
;  948          1021  1 !--
;  949          1022  1
;  950          1023  2 BEGIN
;  951          1024  2
;  952          1025  2 LOCAL
;  953          1026  2     status;
;  954          1027  2
;  955          1028  2
;  956        P 1029  2 $FAB_INIT (FAB = output_fab,                  ! Initialize the output FAB.
;  957        P 1030  2            CTX = output_rab,                  ! Save address of output RAB.
;  958        P 1031  2            FAC = PUT,                         ! File access is write.
;  959        P 1032  2            ORG = SEQ,                         ! File organization is sequential.
;  960        P 1033  2            NAM = output_nam,                  ! Address of output NAM block.
;  961        P 1034  2            FNS = %CHARCOUNT('SYS$OUTPUT'),    ! Default name is SYS$OUTPUT
;  962        P 1035  2            FNA = UPLIT BYTE('SYS$OUTPUT'),
;  963        P 1036  2            FOP = <SUP,OFP,SQO>,               ! File options are supersede, output file parse, seq
;  964        P 1037  2            MRS = .input_fab[fab$w_mrs],       ! Maximum recored size from $GET.
;  965          1038  2            FSZ = .input_fab[fab$b_fsz]);      ! Fixed control area size.
;  966          1039  2
;  967        P 1040  2 $RAB_INIT (RAB = output_rab,                  ! Initialize output RAB.
;  968        P 1041  2            RAC = SEQ,                         ! Record access is sequential.
;  969        P 1042  2            RHB = rh_buffer,                   ! Record header buffer address
;  970          1043  2            FAB = output_fab);                 ! Address of output FAB.
;  971          1044  2
;  972        P 1045  2 $NAM_INIT (NAM = output_nam,                  ! Initalize output NAM.
;  973        P 1046  2            ESS = nam$c_maxrss,                ! Expanded string size
;  974        P 1047  2            ESA = type$output_rsa,             ! Expanded string address
;  975        P 1048  2            RSS = nam$c_maxrss,                ! Resultant string size
;  976        P 1049  2            RSA = type$output_rsa,             ! Resultant string address
;  977          1050  2            RLF = input_nam);                  ! Related file NAM block.
```

```
;  978          1051  2
;  979          1052  2
;  980          1053  2 type$gen_flags[type$v_sysoutput] = TRUE;                  ! Assume writing to sys$output
;  981          1054  2 output_fab[fab$b_rfm] = .input_fab[fab$b_rfm];           ! Init output record format to input format.
;  982          1055  2 output_fab[fab$b_rat] = .input_fab[fab$b_rat];           ! Init outpur record attr to input attr.
;  983          1056  2
;  984          1057  2 status = cli$present($descriptor('OUTPUT'));
;  985          1058  2 IF .status THEN
;  986          1059  3     BEGIN
;  987          1060  3     type$gen_flags[type$v_sysoutput] = false;            ! Not writing to sys$output
;  988          1061  3     cli$get_value($descriptor('OUTPUT'), output_desc); ! Get output file
;  989          1062  3     output_fab[fab$b_fns] = .output_desc[dsc$w_length];
;  990          1063  3     output_fab[fab$l_fna] = .output_desc[dsc$a_pointer];
;  991          1064  3     END
;  992          1065  2 ELSE
;  993          1066  3     BEGIN
;  994          1067  3     IF .status EQL cli$_negated
;  995          1068  3     THEN
;  996          1069  4         BEGIN
;  997          1070  4         type$gen_flags[type$v_sysoutput] = false;            ! Not writing to sys$output
;  998          1071  4         output_fab[fab$b_fns] = %CHARCOUNT('NL:');       ! Default name is NL:
;  999          1072  4         output_fab[fab$l_fna] = UPLIT BYTE('NL:');
; 1000          1073  3         END;
; 1001          1074  2     END;
; 1002          1075  2
; 1003          1076  2
; 1004          1077  2 IF (NOT $CREATE(FAB=output_fab)) OR                      ! If we fail CREATING or
; 1005          1078  3    (NOT $CONNECT(RAB=output_rab))                        ! CONNECTING to the output file
; 1006          1079  2 THEN                                                     ! then signal an error and return.
; 1007          1080  3     BEGIN
; 1008          1081  3     type$file_error(msg$_openout,output_fab);
; 1009          1082  3     RETURN false;
; 1010          1083  2     END;
; 1011          1084  2
; 1012          1085  2 RETURN true;
; 1013          1086  1 END;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                54  55  50  54  55  4F  24  53  59  53  00124 P.AAZ:  .ASCII  \SYS$OUTPUT\                    ;
                                54  55  50  54  55  4F  0012E P.ABB:  .ASCII  \OUTPUT\                      ;
                                            00000006    00134 P.ABA:  .LONG   6                             ;
                                            00000000'   00138         .ADDRESS P.ABB                    ;
                                54  55  50  54  55  4F  0013C P.ABD:  .ASCII  \OUTPUT\                      ;
                                                00142         .BLKB   2                                   ;
                                            00000006    00144 P.ABC:  .LONG   6                             ;
                                            00000000'   00148         .ADDRESS P.ABD                    ;
                                        3A  4C  4E      0014C P.ABE:  .ASCII  \NL:\                       ;

                                                        $RMS_PTR=           OUTPUT_FAB
                                                        $RMS_PTR=           OUTPUT_RAB
                                                        $RMS_PTR=           OUTPUT_NAM
                                                        .EXTRN  SYS$CREATE

                                                        .PSECT  $CODE$,NOWRT,2
```

```
                                     007C 00000 TYPE$OPEN_OUTPUT:
                                                      .WORD   Save R2,R3,R4,R5,R6                 ; 0994
                     56     0000'  CF  9E 00002       MOVAB   $RMS_PTR, R6                        ;
0050  8F      00     6E            00  2C 00007       MOVC5   #0, (SP), #0, #80, $RMS_PTR         ; 1038
                                   66     0000E                                                   ;
                     66     5003   8F  B0 0000F       MOVW    #20483, $RMS_PTR                    ;
              04     A6 20000044   8F  D0 00014       MOVL    #536870980, $RMS_PTR+4              ;
              16     A6            01  90 0001C       MOVB    #1, $RMS_PTR+22                     ;
              18     A6       50   A6  9E 00020       MOVAB   OUTPUT_RAB, $RMS_PTR+24             ;
                              1D   A6  94 00025       CLRB    $RMS_PTR+29                         ;
              1F     A6            02  90 00028       MOVB    #2, $RMS_PTR+31                     ;
              28     A6     0094   C6  9E 0002C       MOVAB   OUTPUT_NAM, $RMS_PTR+40             ;
              2C     A6     0000'  CF  9E 00032       MOVAB   P.AAZ, $RMS_PTR+44                  ;
              34     A6            0A  90 00038       MOVB    #10, $RMS_PTR+52                    ;
              36     A6     FF42   C6  B0 0003C       MOVW    INPUT_FAB+54, $RMS_PTR+54           ;
              3F     A6     FF4B   C6  90 00042       MOVB    INPUT_FAB+63, $RMS_PTR+63           ;
0044  8F      00     6E            00  2C 00048       MOVC5   #0, (SP), #0, #68, $RMS_PTR         ; 1043
                              50   A6     0004F                                                   ;
              50     A6     4401   8F  B0 00051       MOVW    #17409, $RMS_PTR                    ;
                              6E   A6  94 00057       CLRB    $RMS_PTR+30                         ;
              7C     A6     FDEC   C6  9E 0005A       MOVAB   RH_BUFFER, $RMS_PTR+44              ;
            008C     C6            66  9E 00060       MOVAB   OUTPUT_FAB, $RMS_PTR+60             ;
0060  8F      00     6E            00  2C 00065       MOVC5   #0, (SP), #0, #96, $RMS_PTR         ; 1050
                            0094   C6     0006C                                                   ;
            0094     C6     6002   8F  B0 0006F       MOVW    #24578, $RMS_PTR                    ;
            0096     C6            01  8E 00076       MNEGB   #1, $RMS_PTR+2                      ;
            0098     C6     FCEC   C6  9E 0007B       MOVAB   TYPE$OUTPUT_RSA, $RMS_PTR+4         ;
            009E     C6            01  8E 00082       MNEGB   #1, $RMS_PTR+10                     ;
            00A0     C6     FCEC   C6  9E 00087       MOVAB   TYPE$OUTPUT_RSA, $RMS_PTR+12        ;
            00A4     C6       A0   C6  9E 0008E       MOVAB   INPUT_NAM, $RMS_PTR+16              ;
            FAE4     C6            04  88 00094       BISB2   #4, TYPE$GEN_FLAGS                  ; 1053
              1E     A6     FF2A   C6  B0 00099       MOVW    INPUT_FAB+30, OUTPUT_FAB+30         ; 1055
                            0000'  CF  9F 0009F       PUSHAB  P.ABA                               ; 1057
       00000000G     00            01  FB 000A3       CALLS   #1, CLI$PRESENT                     ;
                     22            50  E9 000AA       BLBC    STATUS, 1$                          ; 1058
            FAE4     C6            04  8A 000AD       BICB2   #4, TYPE$GEN_FLAGS                  ; 1060
                            FEFC   C6  9F 000B2       PUSHAB  OUTPUT_DESC                         ; 1061
                            0000'  CF  9F 000B6       PUSHAB  P.ABC                               ;
       00000000G     00            02  FB 000BA       CALLS   #2, CLI$GET_VALUE                   ;
              34     A6     FEFC   C6  90 000C1       MOVB    OUTPUT_DESC, OUTPUT_FAB+52          ; 1062
              2C     A6     FF00   C6  D0 000C7       MOVL    OUTPUT_DESC+4, OUTPUT_FAB+44        ; 1063
                                   1B  11 000CD       BRB     2$                                  ; 1058
                     51 00000000G  00  9E 000CF 1$:   MOVAB   CLI$_NEGATED, R1                    ; 1067
                     51            50  D1 000D6       CMPL    STATUS, R1                          ;
                                   0F  12 000D9       BNEQ    2$                                  ;
            FAE4     C6            04  8A 000DB       BICB2   #4, TYPE$GEN_FLAGS                  ; 1070
              34     A6            03  90 000E0       MOVB    #3, OUTPUT_FAB+52                   ; 1071
              2C     A6     0000'  CF  9E 000E4       MOVAB   P.ABE, OUTPUT_FAB+44                ; 1072
                                   56  DD 000EA 2$:   PUSHL   R6                                  ; 1077
       00000000G     00            01  FB 000EC       CALLS   #1, SYS$CREATE                      ;
                     0D            50  E9 000F3       BLBC    R0, 3$                              ;
                              50   A6  9F 000F6       PUSHAB  OUTPUT_RAB                          ; 1078
       00000000G     00            01  FB 000F9       CALLS   #1, SYS$CONNECT                     ;
                     0F            50  E8 00100       BLBS    R0, 4$                              ;
                                   56  DD 00103 3$:   PUSHL   R6                                  ; 1081
                        009510A4   8F  DD 00105       PUSHL   #9769124                            ;
```

```
                 0000V  CF    02 FB 0010B        CALLS   #2, TYPE$FILE_ERROR      ;
                              04 11 00110        BRB     5$                       ; 1082
                        50    01 D0 00112 4$:    MOVL    #1, R0                   ; 1085
                                 04 00115        RET                              ;
                              50 D4 00116 5$:    CLRL    R0                       ; 1086
                                 04 00118        RET                              ;

; Routine Size:  281 bytes,    Routine Base:  $CODE$ + 06B8

; 1014        1087  1
```

```
;1016     1088  1 GLOBAL ROUTINE type$search_error (fab_block) : NOVALUE =
;1017     1089  1
;1018     1090  1 !++
;1019     1091  1 ! Functional description:
;1020     1092  1 !
;1021     1093  1 !       This routine reports an error as a result of searching for the
;1022     1094  1 !       next file to be typed.
;1023     1095  1 !
;1024     1096  1 ! Calling sequence:
;1025     1097  1 !
;1026     1098  1 !       type_error (fab_block.ra.v)
;1027     1099  1 !
;1028     1100  1 ! Input parameters:
;1029     1101  1 !
;1030     1102  1 !       fab_block       - the FAB associated with the file
;1031     1103  1 !
;1032     1104  1 ! Implicit inputs:
;1033     1105  1 !
;1034     1106  1 !       none
;1035     1107  1 !
;1036     1108  1 ! Output parameters:
;1037     1109  1 !
;1038     1110  1 !       none
;1039     1111  1 !
;1040     1112  1 ! Implicit outputs:
;1041     1113  1 !
;1042     1114  1 !       none
;1043     1115  1 !
;1044     1116  1 ! Routine value:
;1045     1117  1 !
;1046     1118  1 !       none
;1047     1119  1 !
;1048     1120  1 ! Side effects:
;1049     1121  1 !
;1050     1122  1 !       none
;1051     1123  1 !
;1052     1124  1 !--
;1053     1125  1
;1054     1126  2 BEGIN
;1055     1127  2
;1056     1128  2 type$file_error(msg$_searchfail,.fab_block);          ! Report specified RMS error.
;1057     1129  2
;1058     1130  1 END;


                                                0000 00000          .ENTRY  TYPE$SEARCH_ERROR, Save nothing                 ;1088
                                          04    AC   DD 00002       PUSHL   FAB_BLOCK                                       ;1128
                                    00951238    8F   DD 00005       PUSHL   #9769528                                        ;
                          0000V     CF          02   FB 0000B       CALLS   #2, TYPE$FILE_ERROR                             ;
                                                04 00010            RET                                                     ;1130

; Routine Size:  17 bytes,    Routine Base:  $CODE$ + 07D1
```

: 1059 1131 1

```
;  1061          1132  1 GLOBAL ROUTINE type$file_error (message_id, fab_block) : NOVALUE =
;  1062          1133  1
;  1063          1134  1 !++
;  1064          1135  1 ! Functional description
;  1065          1136  1 !
;  1066          1137  1 !       This RMS error action routine sends an error message to the user.
;  1067          1138  1 !
;  1068          1139  1 ! Calling sequence
;  1069          1140  1 !
;  1070          1141  1 !       type$file_error (message_id.rv, fab_block.ra.v)
;  1071          1142  1 !
;  1072          1143  1 ! Input parameters
;  1073          1144  1 !
;  1074          1145  1 !       message_id      - The message code for the message to send.
;  1075          1146  1 !       fab_block       - Address of the FAB block of the file for which the error occurred
;  1076          1147  1 !
;  1077          1148  1 ! Implicit inputs
;  1078          1149  1 !
;  1079          1150  1 !       The associated NAM block.
;  1080          1151  1 !
;  1081          1152  1 ! Output parameters
;  1082          1153  1 !
;  1083          1154  1 !       none
;  1084          1155  1 !
;  1085          1156  1 ! Implicit outputs
;  1086          1157  1 !
;  1087          1158  1 !       none
;  1088          1159  1 !
;  1089          1160  1 ! Routine value
;  1090          1161  1 !
;  1091          1162  1 !       novalue
;  1092          1163  1 !
;  1093          1164  1 ! Side effects
;  1094          1165  1 !
;  1095          1166  1 !       none
;  1096          1167  1 !
;  1097          1168  1 !--
;  1098          1169  1
;  1099          1170  2 BEGIN
;  1100          1171  2
;  1101          1172  2 MAP
;  1102          1173  2     fab_block    : REF $BBLOCK;
;  1103          1174  2
;  1104          1175  2 BIND
;  1105          1176  2     rab_block = .fab_block [fab$l_ctx]  : $BBLOCK,      ! Associated RAB block address
;  1106          1177  2     nam_block = .fab_block [fab$l_nam]  : $BBLOCK;      ! Associated NAM block address
;  1107          1178  2
;  1108          1179  2 LOCAL
;  1109          1180  2     status                : VECTOR [2],
;  1110          1181  2     name_desc    : VECTOR [2];                  ! String descriptor for the file name
;  1111          1182  2
;  1112          1183  2 !
;  1113          1184  2 ! Fill in the file name descriptor with the most complete name possible.
;  1114          1185  2 !
;  1115          1186  2
;  1116          1187  2     IF .nam_block [nam$b_rsl] NEQ 0                    ! If a resultant name string exists,
;  1117          1188  2     THEN                                               !
```

```
3             BEGIN                                                  !
3             name_desc [0] = .nam_block [nam$b_rsl];                ! then fill in the resultant name length
3             name_desc [1] = .nam_block [nam$l_rsa];                ! and address.
3             END
2         ELSE
2             IF .nam_block [nam$b_esl] NEQ 0                        ! If RMS created an expanded string
2             THEN                                                   ! but couldn't open the file,
3                 BEGIN
3                 name_desc [0] = .nam_block [nam$b_esl];            ! then fill in the expanded name length
3                 name_desc [1] = .nam_block [nam$l_esa];            ! and address.
3                 END
2             ELSE                                                   ! Otherwise, no RMS name information is available.
3                 BEGIN
3                 name_desc [0] = .fab_block [fab$b_fns];            ! So use the file name lenc
3                 name_desc [1] = .fab_block [fab$l_fna];            ! and length passed by the CLI.
2                 END;
2
2 IF NOT .fab_block[fab$l_sts]                                       ! Check to see if the FAB
2 THEN                                                               !  or RAB contains the error
3     BEGIN                                                          !  status, and retrieve it.
3     status[0] = .fab_block [fab$l_sts];                            ! The primary RMS completion code,
3     status[1] = .fab_block [fab$l_stv];                            !  and the scondary RMS completion code.
3     END
2 ELSE
3     BEGIN                                                          ! Error status is located in RAB.
3     status[0] = .rab_block [rab$l_sts];                            ! The primary RMS completion code,
3     status[1] = .rab_block [rab$l_stv];                            !  and the scondary RMS completion code.
2     END;
2
2 !
2 ! Signal the error condition.
2 !
2
2     SIGNAL (                                                       ! Signal error with the following arguments:
2             .message_id,                                           !    the message identifier,
2             1,                                                     !    the number of message arguments,
2             name_desc,                                             !    the address of input name descriptor,
2             .status[0],                                            !    the primary RMS completion code,
2             .status[1]);                                           !    and the scondary RMS completion code.
2
2
1     END;



                                 0004 00000     .ENTRY  TYPE$FILE_ERROR, Save R2              ; 1132
            5E             10     C2 00002     SUBL2   #16, SP                               ;
            51      08     AC     D0 00005     MOVL    FAB_BLOCK, R1                         ; 1176
            52      18     A1     D0 00009     MOVL    24(R1), R2                            ;
            50      28     A1     D0 0000D     MOVL    40(R1), R0                            ; 1177
                    03     A0     95 00011     TSTB    3(R0)                                 ; 1187
                           0B     13 00014     BEQL    1$                                    ;
            6E      03     A0     9A 00016     MOVZBL  3(R0), NAME_DESC                      ; 1190
      04    AE      04     A0     D0 0001A     MOVL    4(R0), NAME_DESC+4                    ; 1191
                           19     11 0001F     BRB     3$                                    ; 1187
```

```
                          0B  A0  95 00021 1$:     TSTB    11(R0)                         ; 1194
                              0B  13 00024         BEQL    2$                             ;
               6E         0B  A0  9A 00026         MOVZBL  11(R0), NAME_DESC              ; 1197
          04   AE         0C  A0  D0 0002A         MOVL    12(R0), NAME_DESC+4            ; 1198
                              09  11 0002F         BRB     3$                             ; 1194
               6E         34  A1  9A 00031 2$:     MOVZBL  52(R1), NAME_DESC              ; 1202
          04   AE         2C  A1  D0 00035         MOVL    44(R1), NAME_DESC+4            ; 1203
               07         08  A1  E8 0003A 3$:     BLBS    8(R1), 4$                      ; 1206
          08   AE         08  A1  7D 0003E         MOVQ    8(R1), STATUS                  ; 1209
                              05  11 00043         BRB     5$                             ; 1206
          08   AE         08  A2  7D 00045 4$:     MOVQ    8(R2), STATUS                  ; 1214
                          0C  AE  DD 0004A 5$:     PUSHL   STATUS+4                       ; 1227
                          0C  AE  DD 0004D         PUSHL   STATUS                         ; 1226
                          08  AE  9F 00050         PUSHAB  NAME_DESC                      ; 1222
                              01  DD 00053         PUSHL   #1                             ;
                          04  AC  DD 00055         PUSHL   MESSAGE_ID                     ; 1223
     00000000G  00            05  FB 00058         CALLS   #5, LIB$SIGNAL                 ;
                                  04 0005F         RET                                    ; 1230

; Routine Size:  96 bytes,    Routine Base:  $CODE$ + 07E2

; 1160          1231  1
```

```
;  1162          1232  1 ROUTINE condit_handler (signal_array, mechan_array) =
;  1163          1233  1
;  1164          1234  1 !++
;  1165          1235  1 ! Functional description
;  1166          1236  1 !
;  1167          1237  1 !       This routine is the condition handler for the main routine.  It
;  1168          1238  1 !       saves the most severe condition as the exit status.
;  1169          1239  1 !
;  1170          1240  1 ! Calling sequence
;  1171          1241  1 !
;  1172          1242  1 !       condit_handler (signal_array.ra.v, mechan_array.ra.v)
;  1173          1243  1 !
;  1174          1244  1 ! Input parameters
;  1175          1245  1 !
;  1176          1246  1 !       signal_array    - the address of the signal array for the condition
;  1177          1247  1 !       mechan_array    - the address of the mechanism array for the condition
;  1178          1248  1 !
;  1179          1249  1 ! Implicit inputs
;  1180          1250  1 !
;  1181          1251  1 !       none
;  1182          1252  1 !
;  1183          1253  1 ! Output parameters
;  1184          1254  1 !
;  1185          1255  1 !       none
;  1186          1256  1 !
;  1187          1257  1 ! Implicit outputs
;  1188          1258  1 !
;  1189          1259  1 !       TYPE$EXIT_STATUS        -  Contains the most severe status encountered.
;  1190          1260  1 !
;  1191          1261  1 ! Routine value
;  1192          1262  1 !
;  1193          1263  1 !       SS$_RESIGNAL
;  1194          1264  1 !
;  1195          1265  1 !--
;  1196          1266  1
;  1197          1267  2 BEGIN
;  1198          1268  2
;  1199          1269  2 MAP
;  1200          1270  2     signal_array        : REF $BBLOCK;
;  1201          1271  2
;  1202          1272  2 BIND
;  1203          1273  2     signame = signal_array [chf$l_sig_name] : $BBLOCK;  ! Get the condition name
;  1204          1274  2
;  1205          1275  2
;  1206          1276  2 !
;  1207          1277  2 ! Update the "most severe error" if the current error is more severe.
;  1208          1278  2 !
;  1209          1279  2
;  1210          1280  2 IF
;  1211          1281  2     NOT ,signame                                        ! If an error signal
;  1212          1282  4     AND ((.signame[sts$v_severity]                      ! and severity is worse
;  1213          1283  4         GTRU .type$exit_status[sts$v_severity])
;  1214          1284  3         OR .type$exit_status[sts$v_severity])           ! or no errors yet
;  1215          1285  2 THEN
;  1216          1286  2     type$exit_status = .signame;                        ! then save it for exit
;  1217          1287  2
;  1218          1288  2
```

```
; 1219          1289  2 RETURN SS$_RESIGNAL;                                  ! Resignal to get message
; 1220          1290  1 END;


                                          0004 00000 CONDIT_HANDLER:
                                                           .WORD   Save R2                                   ; 1232
                        52      0000'  CF   9E 00002       MOVAB   TYPE$EXIT_STATUS, R2                      ;
             50     04  AC             04   C1 00007       ADDL3   #4, SIGNAL_ARRAY, R0                      ; 1273
                        12             60   E8 0000C       BLBS    (R0), 2$                                  ; 1281
      51     62         03             00   EF 0000F       EXTZV   #0, #3, TYPE$EXIT_STATUS, R1              ; 1283
      51     60         03             00   ED 00014       CMPZV   #0, #3, (R0), R1                          ;
                                       03   1A 00019       BGTRU   1$                                        ;
                        03             62   E9 0001B       BLBC    TYPE$EXIT_STATUS, 2$                      ; 1284
                        62             60   D0 0001E 1$:   MOVL    (R0), TYPE$EXIT_STATUS                    ; 1286
                        50      0918   8F   3C 00021 2$:   MOVZWL  #2328, R0                                 ; 1289
                                            04 00026       RET                                               ; 1290

; Routine Size:  39 bytes,    Routine Base:  $CODE$ + 0842

; 1221          1291  1
```

```
; 1223          1292  1 END
; 1224          1293  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL, LIB$STOP

;                               PSECT SUMMARY

;       Name                    Bytes                           Attributes

;  $GLOBAL$                     1588  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  LBR_ADDRESSES                  32  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  LBR_NAMES                      56  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $OWN$                          28  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $PLIT$                        335  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                       2153  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       154         1       581         00:01.0



;                                   COMMAND QUALIFIERS

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

; Size:          2153 code + 2039 data bytes
; Run Time:          00:40.2
; Elapsed Time:      01:58.5
; Lines/CPU Min:      1932
; Lexemes/CPU-Min:  32746
; Memory Used:  203 pages
; Compilation Complete
```

TYPEMAIN
LIS

COBPROLOG
REQ

COBACCDAT
LIS

COBACCDWK
LIS

UNLOCK
LIS

UTILSUBS
LIS

INTPAR
SDL

COBDEF
REQ

COBACCDAY
LIS

COBACCECV
LIS

COBRTL

COBLNK
REQ

COBRTL
MAP