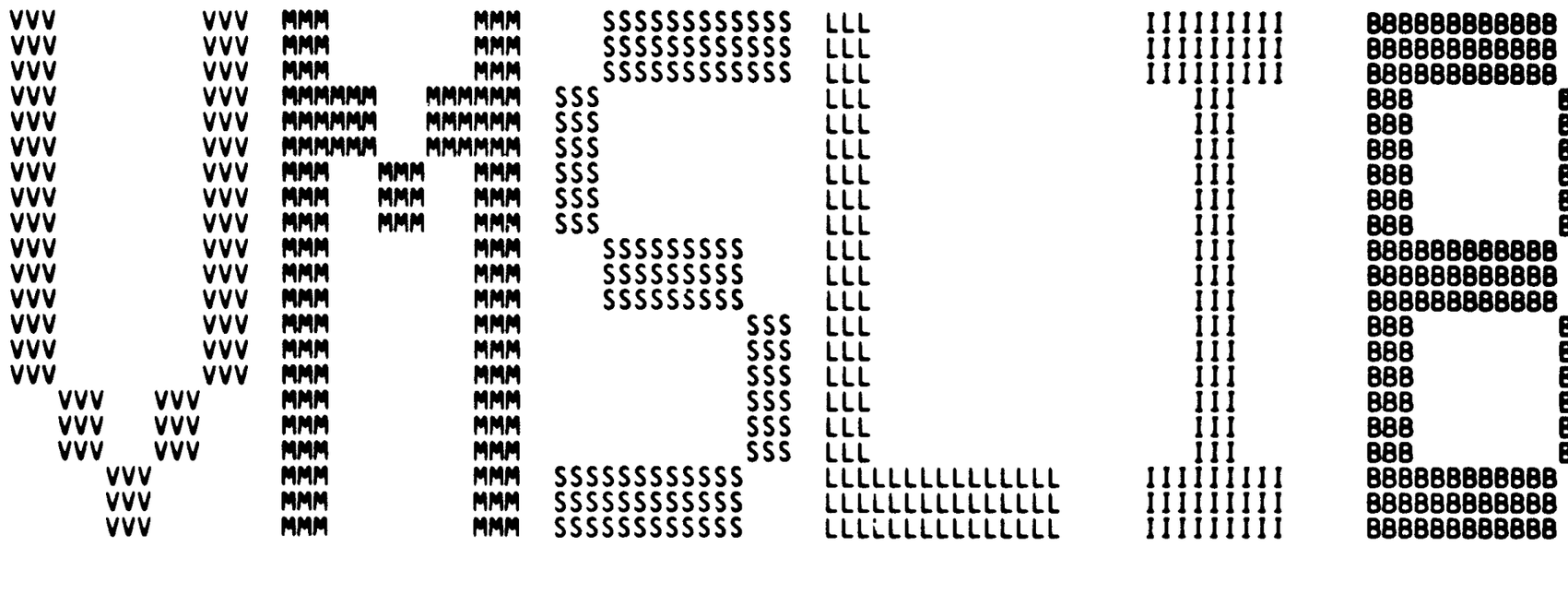
VVV VVV MMM MMM SSSSSSSSSSS LLL IIIIIIIII BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSSSS LLL IIIIIIIII BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSSSS LLL IIIIIIIII BBBBBBBBBBB
VVV VVV MMMMMM MMMMMM SSS LLL III BBB BBB
VVV VVV MMMMMM MMMMMM SSS LLL III BBB BBB
VVV VVV MMMMMM MMMMMM SSS LLL III BBB BBB
VVV VVV MMM MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSSSSSSSS LLL III BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSS LLL III BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSS LLL III BBBBBBBBBBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSS LLL III BBB BBB
VVV VVV MMM MMM SSSSSSSSSSS LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBB
VVV VVV MMM MMM SSSSSSSSSSS LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBB
VVV MMM MMM SSSSSSSSSSS LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBB

```
LL              IIIIII    BBBBBBBB      CCCCCCCC    QQQQQQ    UU      UU    AAAAAA    LL
LL              IIIIII    BBBBBBBB      CCCCCCCC    QQQQQQ    UU      UU    AAAAAA    LL
LL                II      BB      BB  CC          QQ      QQ  UU      UU  AA      AA  LL
LL                II      BB      BB  CC          QQ      QQ  UU      UU  AA      AA  LL
LL                II      BB      BB  CC          QQ      QQ  UU      UU  AA      AA  LL
LL                II      BB      BB  CC          QQ      QQ  UU      UU  AA      AA  LL
LL                II      BBBBBBBB    CC          QQ      QQ  UU      UU  AA      AA  LL
LL                II      BBBBBBBB    CC          QQ      QQ  UU      UU  AA      AA  LL
LL                II      BB      BB  CC          QQ  QQ  QQ  UU      UU  AAAAAAAAAA  LL
LL                II      BB      BB  CC          QQ  QQ  QQ  UU      UU  AAAAAAAAAA  LL
LL                II      BB      BB  CC          QQ    QQ    UU      UU  AA      AA  LL              ....
LL                II      BB      BB  CC          QQ    QQ    UU      UU  AA      AA  LL              ....
LLLLLLLLLL      IIIIII    BBBBBBBB      CCCCCCCC    QQQQ  QQ  UUUUUUUUUU  AA      AA  LLLLLLLLLL      ....
LLLLLLLLLL      IIIIII    BBBBBBBB      CCCCCCCC    QQQQ  QQ  UUUUUUUUUU  AA      AA  LLLLLLLLLL      ....

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

```
;     1          0001  0 MODULE common_file_qualifiers(
;     2          0002  0                 %TITLE 'Common file qualifier'
;     3          0003  0                 IDENT = 'V04-000'
;     4          0004  0                 ) =
;     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 !++
;    34          0034  1 ! FACILITY:
;    35          0035  1 !       VAX/VMS Run-Time Library
;    36          0036  1 !
;    37          0037  1 ! ABSTRACT:
;    38          0038  1 !       This module contains routines which perform command qualifier parsing
;    39          0039  1 !       and analysis for a set of qualifiers which select files based on
;    40          0040  1 !       information contained in the RMS control blocks.
;    41          0041  1 !
;    42          0042  1 ! ENVIRONMENT:
;    43          0043  1 !       VAX/VMS user mode
;    44          0044  1 !
;    45          0045  1 !--
;    46          0046  1 !
;    47          0047  1 ! AUTHOR:  Tamar Krichevsky,    CREATION DATE: 29-Jun-1982
;    48          0048  1 !          (with acknowledgements to Marty Jack and BACKUP)
;    49          0049  1 !
;    50          0050  1 ! MODIFIED BY:
;    51          0051  1 !
;    52          0052  1 !       V03-013 TSK0012         Tamar Krichevky          19-Jul-1984
;    53          0053  1 !               Use the most complete name when signaling invalid qualifier
;    54          0054  1 !               value for the /EXCLUDE qualifier.  In other words, if there
;    55          0055  1 !               is and resultant or expanded name string, use those, instead
;    56          0056  1 !               of the value on the command line.
;    57          0057  1 !
```

```
;    58          0058  1 !       V03-012 TSK0011         Tamar Krichevky         28-Mar-1984
;    59          0059  1 !               Check the return status from TPARSE, after it attempts to
;    60          0060  1 !               parse the value given by the /BY_OWNER qualifier.  If the
;    61          0061  1 !               return status is SS$_NOSUCHID, then signal that -- instead of
;    62          0062  1 !               CLI$_INVQUALVAL.
;    63          0063  1 !
;    64          0064  1 !       V03-011 TSK0010         Tamar Krichevky         5-Mar-1984
;    65          0065  1 !               Replace a call to CLEAN_UP, in LIB$QUAL_FILE_MATCH, with a
;    66          0066  1 !               direct return to the calling routine.  This avoids referencing
;    67          0067  1 !               the varaible XAB_HOLDER before it is initialized.
;    68          0068  1 !
;    69          0069  1 !               Also, pass back the appropriate value after the answer to the
;    70          0070  1 !               /CONFIRM prompt has been evaluated.
;    71          0071  1 !
;    72          0072  1 !       V03-010 TSK0009         Tamar Krichevky         7-Feb-1984
;    73          0073  1 !               Make sure /BEFORE or /SINCE is specified whenever any of
;    74          0074  1 !               the time qualifiers are given (/CREATED, /EXPIRED...)
;    75          0075  1 !               Also, retrun LIB$_QUIPRO for all files encountered after
;    76          0076  1 !               ^Z or "QUIT" is given as response to a /CONFIRM prompt.
;    77          0077  1 !
;    78          0078  1 !       V03-009 LMP0140         L. Mark Pilant,         24-Aug-1983  1:16
;    79          0079  1 !               Add support for alphanumeric UICs.
;    80          0080  1 !
;    81          0081  1 !       V03-008 TSK0008         Tamar Krichevky         14-Mar-83
;    82          0082  1 !               Change addressing modes to longword relative to avoid
;    83          0083  1 !               truncation errors when linking with large images.
;    84          0084  1 !
;    85          0085  1 !       V03-007 TSK0007         Tamar Krichevky         14-Mar-83
;    86          0086  1 !               Change "qdesc_byowner" reference to "qdesc_by_owner".
;    87          0087  1 !
;    88          0088  1 !       V03-006 TSK0006         Tamar Krichevky         11-Mar-83
;    89          0089  1 !               By popular demand the /BYOWNER qualifier is changed to
;    90          0090  1 !               /BY_OWNER.  Also, fix logic so that the presence of the
;    91          0091  1 !               confirm prompt arguement is only checked after it is determined
;    92          0092  1 !               that it is needed.
;    93          0093  1 !
;    94          0094  1 !       V03-005 TSK0005         Tamar Krichevky         11-FEB-83
;    95          0095  1 !               Change the /UIC qualifier to /BYOWNER.
;    96          0096  1 !
;    97          0097  1 !       V03-004 TSK0004         Tamar Krichevky         29-Dec-82
;    98          0098  1 !               Parse the file name passed to LIB$QUAL_FILE_MATCH and use
;    99          0099  1 !               the expanded name string, if the resultant name string has
;   100          0100  1 !               not been defined (the file has not been open), as the
;   101          0101  1 !               canidate string for MATCH.
;   102          0102  1 !
;   103          0103  1 !       V03-003 TSK0003         Tamar Krichevky         23-NOV-82
;   104          0104  1 !               General clean up and modification.  Several people had
;   105          0105  1 !               suggestions such as removing OWN storage, freeing virtual
;   106          0106  1 !               memory, not signalling RMS returns, etc.  The following lists
;   107          0107  1 !               the changes:
;   108          0108  1 !               - Remove OWN storage, used for TPARSE, and replace it with
;   109          0109  1 !                 action routines to make these routines re-entrant.
;   110          0110  1 !               - Clean up PARSE_EXCL_SPEC a little bit: fix some if statements,
;   111          0111  1 !                 take out extra $PARSE call, etc.
;   112          0112  1 !               - Replace the miriad of IF statements in LEGAL_ANSWER with
;   113          0113  1 !                 a table and loop.
;   114          0114  1 !               - Insert PSECT declarations to make the code PIC and sharable.
```

```
;  115          0115  1 !                - Return RMS file errors instead of signalling them.
;  116          0116  1 !                - Change signalled conditions to have the CLI facility code,
;  117          0117  1 !                  instead of LIB.
;  118          0118  1 !                - Add the LIB$QUAL_FILE_END routine, which cleans up the virtual
;  119          0119  1 !                  memory allocated by LIB$QUAL_FILE_PARSE and used by
;  120          0120  1 !                  LIB$QUAL_FILE_MATCH.
;  121          0121  1 !
;  122          0122  1 !        V03-002 TSK0002         Tamar Krichevsky         8-NOV-82
;  123          0123  1 !                Change logical expression from "IF stmt = 0" to "IF stmt EQL 0".
;  124          0124  1 !
;  125          0125  1 !        V03-001 TSK0001         Tamar Krichevsky        11-Oct-82
;  126          0126  1 !                Change the bit field names to match those in STARLET.  To
;  127          0127  1 !                prevent potential duplicate declarations the format for Common
;  128          0128  1 !                Qualifier Flags names is LIB$V_CQF_...
;  129          0129  1 !**
```

```
;   131          0130  1 !
;   132          0131  1 ! PSECTS
;   133          0132  1 !
;   134          0133  1 PSECT
;   135          0134  1     CODE = $CODE (READ, NOWRITE, EXECUTE, SHARE, PIC,
;   136          0135  1         ADDRESSING_MODE (LONG_RELATIVE)),
;   137          0136  1     PLIT = $CODE (READ, NOWRITE, EXECUTE, SHARE, PIC,
;   138          0137  1         ADDRESSING_MODE (LONG_RELATIVE)),
;   139          0138  1     OWN = $DATA (READ, WRITE, NOEXECUTE, NOSHARE, PIC,
;   140          0139  1         ADDRESSING_MODE (LONG_RELATIVE)),
;   141          0140  1     GLOBAL = $DATA (READ, WRITE, NOEXECUTE, NOSHARE, PIC,
;   142          0141  1         ADDRESSING_MODE (LONG_RELATIVE))
;   143          0142  1     ;
;   144          0143  1
;   145          0144  1 !
;   146          0145  1 ! SWITCHES:
;   147          0146  1 !
;   148          0147  1 SWITCHES ADDRESSING_MODE
;   149          0148  1     (EXTERNAL = GENERAL, NONEXTERNAL = LONG_RELATIVE)
;   150          0149  1     ;
;   151          0150  1 !
;   152          0151  1 ! TABLE OF CONTENTS
;   153          0152  1 !
;   154          0153  1 FORWARD ROUTINE
;   155          0154  1     get_vm,                         ! Allocate some virtual memory
;   156          0155  1     get_zero_vm,                    ! Allocate & clear some virtual memory
;   157          0156  1     parse_excl_spec : NOVALUE,      ! Parse the file spec given in /EXCLUDE
;   158          0157  1     set_uic_value,                  ! Retrieve the UIC values given by /BY_OWNNER
;   159          0158  1     clean_up,                       ! Clean_up RMS blocks and return
;   160          0159  1     legal_answer,                   ! Check answer to /CONFIRM prompt
;   161          0160  1     find_rms_blocks : NOVALUE,      ! Locates needed XAB's in a chain
;   162          0161  1     lib$confirm_act,                ! /CONFIRM action routine
;   163          0162  1     lib$qual_file_parse,            ! Parse the common file qualifiers
;   164          0163  1     lib$qual_file_match,            ! Evaluate file match criteria
;   165          0164  1     lib$qual_file_end               ! Free up any virtual memory allocated
;   166          0165  1     ;
;   167          0166  1
;   168          0167  1
;   169          0168  1 !
;   170          0169  1 ! INCLUDE FILES
;   171          0170  1 !
;   172          0171  1 LIBRARY
;   173          0172  1     'SYS$LIBRARY:TPAMAC'
;   174          0173  1     ;
;   175          0174  1 LIBRARY
;   176          0175  1     'SYS$LIBRARY:STARLET'
;   177          0176  1     ;
;   178          0177  1
;   179          0178  1 !
;   180          0179  1 ! Structure declaration used for system defined structures. This structure
;   181          0180  1 ! is byte sized.
;   182          0181  1 !
;   183          0182  1
;   184          0183  1 STRUCTURE
;   185          0184  1
;   186          0185  1         BBLOCKVECTOR [I, O, P, S, E; N, BS] =
;   187          0186  1             [N*BS]
```

```
;  188          0187  1              ((BBLOCKVECTOR+I*BS)+0)<P,S,E>
;  189          0188  1          ;
;  190          0189  1
```

```
;  192          0190  1 !
;  193          0191  1 ! MACROS
;  194          0192  1 !
;  195          0193  1 MACRO
;  196          0194  1
;  197          0195  1     ! Field definitions for the QUALIFIER database. (QUALIFIER bblock)
;  198          0196  1     !
;  199          0197  1     !
;  200          0198  1     ! The following bit fields in the QUALIFIER area are set if a qualifier
;  201          0199  1     ! is legal (selected by the caller) and also present on the command line.
;  202          0200  1     !
;  203          0201  1     confirm_present     =  0,  0,  1, 0 %,      ! /CONFIRM
;  204          0202  1     exclude_present     =  0,  1,  1, 0 %,      ! /EXCLUDE
;  205          0203  1     before_present      =  0,  2,  1, 0 %,      ! /BEFORE
;  206          0204  1     since_present       =  0,  3,  1, 0 %,      ! /SINCE
;  207          0205  1     created_present     =  0,  4,  1, 0 %,      ! /CREATED
;  208          0206  1     modified_present    =  0,  5,  1, 0 %,      ! /MODIFIED
;  209          0207  1     expired_present     =  0,  6,  1, 0 %,      ! /EXPIRED
;  210          0208  1     backup_present      =  0,  7,  1, 0 %,      ! /BACKUP
;  211          0209  1     byowner_present     =  0,  8,  1, 0 %,      ! /BY_OWNER
;  212          0210  1                                                 !
;  213          0211  1     !
;  214          0212  1     ! These fields point to or contain values that appear on the command
;  215          0213  1     ! line.
;  216          0214  1     !
;  217          0215  1     exclude_list        =  4,  0, 32, 0 %,      ! List head for /EXCLUDE list
;  218          0216  1     before_value        =  8,  0,  0, 0 %,      ! /BEFORE time value, quadword
;  219          0217  1     since_value         = 16,  0,  0, 0 %,      ! /SINCE time value, quadword
;  220          0218  1     uic_value           = 24,  0, 32, 0 %,      ! /BY_OWNER UIC value
;  221          0219  1     uic_member          = 24,  0, 16, 0 %,      !       member portion
;  222          0220  1     uic_group           = 26,  0, 16, 0 %,      !       group portion
;  223          0221  1     uic_wldmem          = 28,  0,  1, 0 %,      !       member value was wild
;  224          0222  1     uic_wldgrp          = 28,  1,  1, 0 %,      !       group value was wild
;  225          0223  1     uic_grp_parsed      = 28,  2,  1, 0 %,      ! Group portion has been parsed
;  226          0224  1     !
;  227          0225  1     ! Field defiintions for flags, which when set effect the operaton of
;  228          0226  1     ! LIB$QUAL_FILE_MATCH.
;  229          0227  1     !
;  230          0228  1     callers_fab         = 28,  3,  1, 0 %,      ! caller passed a file name not  a FAB
;  231          0229  1     file_open           = 28,  4,  1, 0 %,      ! caller has file already open
;  232          0230  1     display_hdr         = 28,  5,  1, 0 %,      ! information in file header will be looked at
;  233          0231  1     quit_processing     = 28,  6,  1, 0 %       ! Response to /CONFIRM prompt was ^Z or QUIT
;  234          0232  1     ;
;  235          0233  1
;  236          0234  1
;  237          0235  1 MACRO
;  238          0236  1
;  239          0237  1     ! Field descriptions for varying string descriptor.
;  240          0238  1     !
;  241          0239  1     var_str_len         =  0,  0, 16, 0 %,      ! current length of string
;  242          0240  1     var_str_body        =  2,  0,  0, 0 %,      ! body of string
;  243          0241  1
;  244          0242  1
;  245          0243  1     ! These MACROS describe an element of the exclude list.  This list
;  246          0244  1     ! contains the file specifications given in the /EXCLUDE qualifier.
;  247          0245  1     ! An element consists of a pointer to the next element and a descriptor
;  248          0246  1     ! for the file specification.
```

```
;  249          0247  1     !
;  250          0248  1     next_excl_blk       =  0,  0, 32, 0 %,      ! Pointer to next element
;  251          0249  1     excl_spec_desc      =  4,  0,  0, 0 %       ! Descriptor in element
;  252          0250  1     ;
```

```
;  254          0251  1     ;
;  255          0252  1 !
;  256          0253  1 ! EQUATED SYMBOLS
;  257          0254  1 !
;  258          0255  1
;  259          0256  1 LITERAL
;  260          0257  1     true        = 1,
;  261          0258  1     false       = 0,
;  262          0259  1     success     = 1,
;  263          0260  1     failure     = 0,
;  264          0261  1     quad        = 8                 ! Size of a quad word, for declarations
;  265          0262  1     ;
;  266          0263  1
;  267          0264  1 LITERAL
;  268          0265  1     qual_blk_len = 32,              ! Length qualifier database
;  269          0266  1     excl_blk_len = 12               ! Length of exclude list element
;  270          0267  1     ;
;  271          0268  1
;  272          0269  1
;  273          0270  1 !
;  274          0271  1 ! EXTERNAL REFERENCES
;  275          0272  1 !
;  276          0273  1 EXTERNAL ROUTINE
;  277          0274  1     CLI$GET_VALUE,              ! Get a qualifier or value from the command line
;  278          0275  1     CLI$PRESENT,                ! Determine if entity is present
;  279          0276  1     LIB$GET_VM,                 ! Allocate some virtual memory
;  280          0277  1     LIB$CVT_TIME,               ! Convert an ASCII time to 64-bits
;  281          0278  1     LIB$TPARSE,                 ! Table driven parser
;  282          0279  1     LIB$SIG_TO_RET,             ! Convert a signal to a return
;  283          0280  1     LIB$GET_COMMAND,            ! Get a sring from SYS$COMMAND
;  284          0281  1     STR$UPCASE,                 ! Up case a string
;  285          0282  1     MATCH,                      ! Compare a file spec to a pattern
;  286          0283  1     LIB$FREE_VM                 ! Deallocate virtual memory
;  287          0284  1     ;
;  288          0285  1
;  289          0286  1
;  290          0287  1 EXTERNAL LITERAL
;  291          0288  1     LIB$_FILFAIMAT,
;  292          0289  1     LIB$_INTLOGERR,
;  293          0290  1     LIB$_INVARG,
;  294          0291  1     LIB$_INVXAB,
;  295          0292  1     LIB$_NEGANS,
;  296          0293  1     LIB$_QUIPRO,
;  297          0294  1     LIB$_QUICONACT
;  298          0295  1     ;
;  299          0296  1
;  300        P 0297  1 $SHR_MSGDEF( CLI, 3, GLOBAL,
;  301        P 0298  1     (CONFQUAL, SEVERE),
;  302        P 0299  1     (INVQUAVAL, SEVERE),
;  303        P 0300  1     (QUALMISS, SEVERE),
;  304        P 0301  1     (NOSUCHID, SEVERE)
;  305          0302  1     );
```

```
;  307          0303  1 ! Address of the qualifier data base has been placed in the TPA$L_PARAM field
;  308          0304  1 ! of the TPARSE parameter block.  THIS FIELD MUST NOT BE ALTERED by any of the
;  309          0305  1 ! state transitions or the action routine will not work.
;  310          0306  1 !
;  311          0307  1 $INIT_STATE( cq_uic_states, cq_uic_keys );
;  312          0308  1
;  313        P 0309  1 $STATE(
;  314        P 0310  1     (TPA$_IDENT,,set_uic_value)
;  315          0311  1     );
;  316        P 0312  1 $STATE(
;  317        P 0313  1     (TPA$_EOS,TPA$_EXIT)
;  318          0314  1     );
;  319          0315  1
;  320          0316  1
```

```
;  322     0317  1 %SBTTL 'GET_VM'
;  323     0318  1 ROUTINE get_vm( size ) =
;  324     0319  1
;  325     0320  1 !++
;  326     0321  1 !
;  327     0322  1 ! FUNCTIONAL DESCRIPTION:
;  328     0323  1 !
;  329     0324  1 !       This routine calls LIB$GET_VM to allocate a block of virtual memory.
;  330     0325  1 !
;  331     0326  1 ! FORMAL PARAMETERS:
;  332     0327  1 !
;  333     0328  1 !       SIZE    :  The size of the area, in bytes
;  334     0329  1 !
;  335     0330  1 ! IMPLICIT INPUTS:
;  336     0331  1 !
;  337     0332  1 !       None
;  338     0333  1 !
;  339     0334  1 ! IMPLICIT OUTPUTS:
;  340     0335  1 !
;  341     0336  1 !       None
;  342     0337  1 !
;  343     0338  1 ! ROUTINE VALUE:
;  344     0339  1 !
;  345     0340  1 !       Address of the allocated area
;  346     0341  1 !
;  347     0342  1 ! COMPLETION CODES:
;  348     0343  1 !
;  349     0344  1 !       Any completion code from LIB$GET_VM
;  350     0345  1 !
;  351     0346  1 ! SIDE EFFECTS:
;  352     0347  1 !
;  353     0348  1 !       If allocation fails, fatal error is signalled
;  354     0349  1 !
;  355     0350  1 !--
;  356     0351  1
;  357     0352  2 BEGIN
;  358     0353  2
;  359     0354  2 LOCAL
;  360     0355  2     rtn_status,                         ! Status returns from LIB$GET_VM call
;  361     0356  2     area_loc                            ! Pointer to the allocated area
;  362     0357  2     ;
;  363     0358  2
;  364     0359  2
;  365     0360  2
;  366     0361  2 ! Allocate some virtual memory; check the return status.  If it isn't
;  367     0362  2 ! successful, signal.
;  368     0363  2 !
;  369     0364  3 IF NOT ( rtn_status = LIB$GET_VM( size, area_loc ))
;  370     0365  2 THEN
;  371     0366  2     SIGNAL( .rtn_status );
;  372     0367  2
;  373     0368  2
;  374     0369  2 ! Return the location of the memory.
;  375     0370  2 !
;  376     0371  2 RETURN .area_loc;
;  377     0372  2
;  378     0373  1 END;                                    ! End of routine get_vm
```

```
                                        .TITLE  COMMON_FILE_QUALIFIERS Common file qualifier
                                        .IDENT  \V04-000\

                                        .PSECT  _LIB$STATE$,NOWRT,  SHR,  PIC,1

                        00000 CQ_UIC_STATES::
                                        .BLKB   0
                    85EC 00000 ;TPA$TYPE
                                U.2:    .WORD   -31252                                  ;
                00000000V 00002 ;TPA$ACTION
                                U.3:    .LONG   <<SET_UIC_VALUE-U.3>-4>                 ;
                    15F7 00006 ;TPA$TYPE
                                U.4:    .WORD   5623                                    ;
                    FFFF 00008 ;TPA$TARGET
                                U.5:    .WORD   -1                                      ;

                                        .PSECT  _LIB$KEY0$,NOWRT,  SHR,  PIC,1

                        00000 CQ_UIC_KEYS::
                                        .BLKB   0
                        00000 ;TPA$KEY0
                                U.1:    .BLKB   0

                                CLI$_CONFQUAL==     201444
                                CLI$_INVQUAVAL==    201516
                                CLI$_QUALMISS==     201540
                                CLI$_NOSUCHID==     201564
                                        .EXTRN  CLI$GET_VALUE, CLI$PRESENT
                                        .EXTRN  LIB$GET_VM, LIB$CVT_TIME
                                        .EXTRN  LIB$TPARSE, LIB$SIG_TO_RET
                                        .EXTRN  LIB$GET_COMMAND
                                        .EXTRN  STR$UPCASE, MATCH
                                        .EXTRN  LIB$FREE_VM, LIB$_FILFAIMAT
                                        .EXTRN  LIB$_INTOVFERR, LIB$_INVARG
                                        .EXTRN  LIB$_INVXAB, LIB$_NEGANS
                                        .EXTRN  LIB$_QUIPRO, LIB$_QUICONACT

                                        .PSECT  $CODE,NOWRT,  SHR,  PIC,2

                            0000 00000 GET_VM:  .WORD   Save nothing                    ; 0318
                        5E          04  C2 00002        SUBL2   #4, SP                  ;
                                    5E  DD 00005        PUSHL   SP                      ; 0364
                                04  AC  9F 00007        PUSHAB  SIZE                    ;
                00000000G   00          02  FB 0000A    CALLS   #2, LIB$GET_VM          ;
                            09          50  E8 00011    BLBS    RTN_STATUS, 1$          ;
                                        50  DD 00014    PUSHL   RTN_STATUS              ; 0366
                00000000G   00          01  FB 00016    CALLS   #1, LIB$SIGNAL          ;
                            50          6E  D0 0001D 1$:  MOVL   AREA_LOC, R0           ; 0371
                                            04 00020    RET                             ; 0373

; Routine Size:  33 bytes,    Routine Base:  $CODE + 0000
```

```
;  380          0374  1 %SBTTL 'GET_ZERO_VM'
;  381          0375  1 ROUTINE get_zero_vm( size ) =
;  382          0376  1
;  383          0377  1 !++
;  384          0378  1 !
;  385          0379  1 ! FUNCTIONAL DESCRIPTION:
;  386          0380  1 !
;  387          0381  1 !       This routine calls LIB$GET_VM to allocate a block of virtual memory
;  388          0382  1 !       and clears the allocted area.
;  389          0383  1 !
;  390          0384  1 ! FORMAL PARAMETERS:
;  391          0385  1 !
;  392          0386  1 !       SIZE    :  The size of the area, in bytes
;  393          0387  1 !
;  394          0388  1 ! IMPLICIT INPUTS:
;  395          0389  1 !
;  396          0390  1 !       None
;  397          0391  1 !
;  398          0392  1 ! IMPLICIT OUTPUTS:
;  399          0393  1 !
;  400          0394  1 !       None
;  401          0395  1 !
;  402          0396  1 ! ROUTINE VALUE:
;  403          0397  1 !
;  404          0398  1 !       Address of the allocated area
;  405          0399  1 !
;  406          0400  1 ! COMPLETION CODES:
;  407          0401  1 !
;  408          0402  1 !       Any completion code from LIB$GET_VM
;  409          0403  1 !
;  410          0404  1 ! SIDE EFFECTS:
;  411          0405  1 !
;  412          0406  1 !       If allocation fails, fatal error is signalled
;  413          0407  1 !
;  414          0408  1 !--
;  415          0409  1
;  416          0410  2 BEGIN
;  417          0411  2
;  418          0412  2 LOCAL
;  419          0413  2     rtn_status,                         ! Status returns from LIB$GET_VM call
;  420          0414  2     area_loc                            ! Pointer to the allocated area
;  421          0415  2     ;
;  422          0416  2
;  423          0417  2
;  424          0418  2 ! Allocate some virtual memory; check the return status.  If it isn't
;  425          0419  2 ! successful, signal.
;  426          0420  2 !
;  427          0421  3 IF NOT ( rtn_status = LIB$GET_VM( size, area_loc ))
;  428          0422  2 THEN
;  429          0423  2     SIGNAL( .rtn_status );
;  430          0424  2
;  431          0425  2
;  432          0426  2 ! Zero out the area just allocated.
;  433          0427  2 !
;  434          0428  2 CH$FILL( 0, .size, .area_loc );
;  435          0429  2
;  436          0430  2
```

```
;  437          0431  2 ! Return the location of the memory.
;  438          0432  2 !
;  439          0433  2 RETURN .area_loc;
;  440          0434  2
;  441          0435  1 END;                                    ! End of routine get_zero_vm


                                                 003C 00000 GET_ZERO_VM:
                                                            .WORD   Save R2,R3,R4,R5                    ; 0375
                                      5E                 04  C2 00002          SUBL2   #4, SP                      ;
                                                         5E  DD 00005          PUSHL   SP                          ; 0421
                                                   04    AC  9F 00007          PUSHAB  SIZE                        ;
                                 00000000G  00           02  FB 0000A          CALLS   #2, LIB$GET_VM              ;
                                            09           50  E8 00011          BLBS    RTN_STATUS, 1$              ;
                                                         50  DD 00014          PUSHL   RTN_STATUS                  ; 0423
                                 00000000G  00           01  FB 00016          CALLS   #1, LIB$SIGNAL              ;
      04    AC                       00    6E           00  2C 0001D 1$:       MOVC5   #0, (SP), #0, SIZE, @AREA_LOC ; 0428
                                                   00    BE      00023         
                                      50                 6E  D0 00025          MOVL    AREA_LOC, R0                ; 0433
                                                            04 00028          RET                                 ; 0435

; Routine Size:  41 bytes,    Routine Base:  $CODE + 0021
```

```
;  443          0436  1 %SBTTL 'PARSE_EXCL_SPEC'
;  444          0437  1 ROUTINE parse_excl_spec( src_desc    : REF $BBLOCK,
;  445          0438  1                         esl_desc    : REF $BBLOCK,
;  446          0439  1                         rln_desc    : REF $BBLOCK,
;  447          0440  1                         qual_desc   : REF $BBLOCK ) : NOVALUE =
;  448          0441  1
;  449          0442  1 !++
;  450          0443  1 !
;  451          0444  1 ! FUNCTIONAL DESCRIPTION:
;  452          0445  1 !
;  453          0446  1 !       This routine parses and checks a value specified for the /EXCLUDE
;  454          0447  1 !       qualifier.
;  455          0448  1 !
;  456          0449  1 ! FORMAL PARAMETERS:
;  457          0450  1 !
;  458          0451  1 !       src_desc        :  pointer to the string return by the CLI (PL/I
;  459          0452  1 !                          varying).
;  460          0453  1 !       rln_desc        :  descriptor for related file name string
;  461          0454  1 !       qual_desc       :  descriptor for /EXCLUDE qualifier; used for
;  462          0455  1 !                          signalling an error.
;  463          0456  1 !
;  464          0457  1 ! IMPLICIT INPUTS:
;  465          0458  1 !
;  466          0459  1 !       None
;  467          0460  1 !
;  468          0461  1 ! OUTPUTS PARAMETERS:
;  469          0462  1 !
;  470          0463  1 !       esl_desc        : descrptor for the expanded string returned by the
;  471          0464  1 !                         $PARSE operation.
;  472          0465  1 !
;  473          0466  1 ! IMPLICIT OUTPUTS:
;  474          0467  1 !
;  475          0468  1 !       None
;  476          0469  1 !
;  477          0470  1 ! ROUTINE VALUE:
;  478          0471  1 !
;  479          0472  1 !       None
;  480          0473  1 !
;  481          0474  1 ! COMPLETION CODES:
;  482          0475  1 !
;  483          0476  1 !       None
;  484          0477  1 !
;  485          0478  1 ! SIDE EFFECTS:
;  486          0479  1 !
;  487          0480  1 !       Virtual memory is allocated to hold the expanded name string.
;  488          0481  1 !
;  489          0482  1 !--
;  490          0483  1
;  491          0484  2 BEGIN
;  492          0485  2
;  493          0486  2 LOCAL
;  494          0487  2     rtn_status,                     ! Return status from external calls
;  495          0488  2     fab :                           ! Local FAB, for parsing file name
;  496          0489  2         $FAB_DECL,
;  497          0490  2     nam :                           ! Local NAM block, for parsing
;  498          0491  2         $NAM_DECL,
;  499          0492  2     rel_nam :                       ! Local related NAM block, for parsing
```

```
;  500          0493  2          $NAM_DECL,
;  501          0494  2      esa :                               ! Area to hold expanded name string
;  502          0495  2          VECTOR[ NAM$C_MAXRSS, BYTE ],
;  503          0496  2      desc:                               ! Descriptor used when signaling errors
;  504          0497  2          $BBLOCK[ DSC$K_S_BLN ] INITIAL (REP DSC$K_S_BLN OF BYTE (0))
;  505          0498  2      ;
;  506          0499  2
;  507          0500  2
;  508          0501  2
;  509          0502  2
;  510          0503  2
;  511          0504  2  ! Initialize the RMS control blocks for the $PARSE operation.
;  512          0505  2  !
;  513        P 0506  2  $FAB_INIT(
;  514        P 0507  2            FAB = fab,
;  515        P 0508  2            FNA = .src_desc[ DSC$A_POINTER ],
;  516        P 0509  2            FNS = .src_desc[ DSC$W_LENGTH ],
;  517        P 0510  2            DNM = '[000000...]*.*;*',
;  518        P 0511  2            NAM = nam
;  519          0512  2           );
;  520          0513  2
;  521        P 0514  2  $NAM_INIT(
;  522        P 0515  2            NAM = nam,
;  523        P 0516  2            ESA = esa,
;  524        P 0517  2            ESS = NAM$C_MAXRSS
;  525          0518  2           );
;  526          0519  2
;  527          0520  2  IF .rln_desc NEQ 0
;  528          0521  2  THEN
;  529          0522  3      BEGIN
;  530          0523  3      nam[ NAM$L_RLF ] = rel_nam;
;  531        P 0524  3      $NAM_INIT(
;  532        P 0525  3                NAM = rel_nam,
;  533        P 0526  3                RSA = .rln_desc[ DSC$A_POINTER ],
;  534        P 0527  3                RSS = .rln_desc[ DSC$W_LENGTH ]
;  535          0528  3               );
;  536          0529  2      END;
;  537          0530  2
;  538          0531  2
;  539          0532  2
;  540          0533  2  ! Parse the file name using $PARSE.
;  541          0534  2  !
;  542          0535  3  IF NOT (rtn_status = $PARSE( FAB = fab ))
;  543          0536  2  THEN
;  544          0537  3      BEGIN
;  545          0538  3      !
;  546          0539  3      ! The $PARSE operation was not successful.  Use the most complete name
;  547          0540  3      ! possible in signaling the error.
;  548          0541  3      !
;  549          0542  3      IF .nam[ NAM$B_RSL ] NEQ 0
;  550          0543  3      THEN
;  551          0544  4          BEGIN
;  552          0545  4          !
;  553          0546  4          ! There is a resultant name string, use it.
;  554          0547  4          !
;  555          0548  4          desc[ DSC$A_POINTER ] = .nam[ NAM$L_RSA ];
;  556          0549  4          desc[ DSC$W_LENGTH ]  = .nam[ NAM$B_RSL ];
```

```
;  557     0550  4          END
;  558     0551  3      ELSE
;  559     0552  4          BEGIN
;  560     0553  4          IF .nam[ NAM$B_ESL ] NEQ 0
;  561     0554  4          THEN
;  562     0555  5              BEGIN
;  563     0556  5              !
;  564     0557  5              ! There is an expanded name string, use it.
;  565     0558  5              !
;  566     0559  5              desc[ DSC$A_POINTER ] = .nam[ NAM$L_ESA ];
;  567     0560  5              desc[ DSC$W_LENGTH ]  = .nam[ NAM$B_ESL ];
;  568     0561  5              END
;  569     0562  4          ELSE
;  570     0563  5              BEGIN
;  571     0564  5              !
;  572     0565  5              ! Use the qualifier value given on the command line.
;  573     0566  5              !
;  574     0567  5              desc[ DSC$A_POINTER ] = .src_desc[ DSC$A_POINTER ];
;  575     0568  5              desc[ DSC$W_LENGTH ]  = .src_desc[ DSC$W_LENGTH ];
;  576     0569  4              END;                                    ! inner else stmt
;  577     0570  3          END;                                        ! outer else stmt
;  578     0571  3      SIGNAL(CLI$_INVQUAVAL, 2, desc, .qual_desc, .rtn_status );
;  579     0572  2      END;
;  580     0573  2
;  581     0574  2 ! Check to make sure an acceptable name was returned.  No null strings, node
;  582     0575  2 ! names, quoted strings or device names.
;  583     0576  2 !
;  584     0577  2 IF .nam[ NAM$B_ESL ] EQL 0
;  585     0578  2          OR
;  586     0579  3      ( .nam[ NAM$L_FNB ] AND
;  587     0580  2          ( NAM$M_EXP_DEV  OR  NAM$M_NODE  OR  NAM$M_QUOTED )) NEQ 0
;  588     0581  2 THEN
;  589     0582  2      SIGNAL(CLI$_INVQUAVAL, 2, .src_desc, .qual_desc );
;  590     0583  2
;  591     0584  2
;  592     0585  2 ! Initalize the descriptor for the expanded name string.
;  593     0586  2 !
;  594     0587  2 esl_desc[ DSC$W_LENGTH ] = .nam[ NAM$B_ESL ];
;  595     0588  2 esl_desc[ DSC$B_DTYPE ]  = DSC$K_DTYPE_T;
;  596     0589  2 esl_desc[ DSC$B_CLASS ]  = DSC$K_CLASS_S;
;  597     0590  2
;  598     0591  2
;  599     0592  2 ! Allocate virtual memory to hold the expanded name string. Have the descriptor
;  600     0593  2 ! point to this new memory.
;  601     0594  2 !
;  602     0595  2 esl_desc[ DSC$A_POINTER ] = get_vm( .nam[ NAM$B_ESL ] );
;  603     0596  2
;  604     0597  2
;  605     0598  2 ! Copy the expanded name string into the allocated memory.
;  606     0599  2 !
;  607     0600  2 CH$MOVE(
;  608     0601  2          .nam[ NAM$B_ESL ],
;  609     0602  2          esa,
;  610     0603  2          .esl_desc[ DSC$A_POINTER ]
;  611     0604  2         );
;  612     0605  2
;  613     0606  1 END;                                    ! End of routine parse_excl_spec
```

```
3B 2A 2E 2A 5D 2E 2E 2E 30 30 30 30 30 30 5B  0004A P.AAA:  .ASCII  \[000000...]*.*;*\
                                              2A  00059

                                                          .EXTRN  SYS$PARSE

                                              03FC 00000 PARSE_EXCL_SPEC:
                                                          .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9                ; 0437
                                  59      EB  AF  9E 00002        MOVAB   P.AAA, R9                           ;
                                  58 00000000G 00 9E 00006        MOVAB   LIB$SIGNAL, R8                      ;
                                  5E      FDF0 CE 9E 0000D        MOVAB   -528(SP), SP                        ;
                                               7E 7C 00012        CLRQ    DESC                                ; 0484
    0050  8F        00            6E           00 2C 00014        MOVC5   #0, (SP), #0, #80, $RMS_PTR         ; 0512
                                         B0     AD     0001B
                              B0  AD      5003  8F B0 0001D        MOVW    #20483, $RMS_PTR                    ;
                              C6  AD            02 90 00023        MOVB    #2, $RMS_PTR+22                     ;
                              CF  AD            02 90 00027        MOVB    #2, $RMS_PTR+31                     ;
                              D8  AD      FF50  CD 9E 0002B        MOVAB   NAM, $RMS_PTR+40                    ;
                                  57        04  AC D0 00031        MOVL    SRC_DESC, R7                        ;
                              DC  AD        04  A7 D0 00035        MOVL    4(R7), $RMS_PTR+44                  ;
                              E0  AD            69 9E 0003A        MOVAB   P.AAA, $RMS_PTR+48                  ;
                              E4  AD            67 90 0003E        MOVB    (R7), $RMS_PTR+52                   ;
                              E5  AD            10 90 00042        MOVB    #16, $RMS_PTR+53                    ;
    0060  8F        00            6E           00 2C 00046        MOVC5   #0, (SP), #0, #96, $RMS_PTR         ; 0518
                                         FF50   CD     0004D
                          FF50  CD      6002  8F B0 00050        MOVW    #24578, $RMS_PTR                    ;
                          FF5A  CD            01 8E 00057        MNEGB   #1, $RMS_PTR+10                     ;
                          FF5C  CD        08  AE 9E 0005C        MOVAB   ESA, $RMS_PTR+12                    ;
                                  56        0C  AC D0 00062        MOVL    RLN_DESC, R6                        ; 0520
                                              23 13 00066        BEQL    1$                                  ;
                          FF60  CD      0108  CE 9E 00068        MOVAB   REL_NAM, NAM+16                     ; 0523
0060  8F        00                6E          00 2C 0006F        MOVC5   #0, (SP), #0, #96, $RMS_PTR         ; 0528
                                        0108  CE     00076
                          0108  CE      6002  8F B0 00079        MOVW    #24578, $RMS_PTR                    ;
                          010A  CE            66 90 00080        MOVB    (R6), $RMS_PTR+2                    ;
                          FEF4  CD        04  A6 D0 00085        MOVL    4(R6), $RMS_PTR+4                   ;
                                          B0  AD 9F 0008B 1$:     PUSHAB  FAB                                 ; 0535
                    00000000G 00              01 FB 0008E        CALLS   #1, SYS$PARSE                       ;
                              3C              50 E8 00095        BLBS    RTN_STATUS, 6$                      ;
                                  51    FF53  CD 9A 00098        MOVZBL  NAM+3, R1                           ; 0542
                                              08 13 0009D        BEQL    2$                                  ;
                              04  AE    FF54  CD D0 0009F        MOVL    NAM+4, DESC+4                       ; 0548
                                              0D 11 000A5        BRB     3$                                  ; 0549
                                  51    FF5B  CD 9A 000A7 2$:     MOVZBL  NAM+11, R1                          ; 0553
                                              0B 13 000AC        BEQL    4$                                  ;
                              04  AE    FF5C  CD D0 000AE        MOVL    NAM+12, DESC+4                      ; 0559
                                  6E          51 B0 000B4 3$:     MOVW    R1, DESC                            ; 0560
                                              08 11 000B7        BRB     5$                                  ; 0553
                              04  AE      04  A7 D0 000B9 4$:     MOVL    4(R7), DESC+4                       ; 0567
                                  6E          67 B0 000BE        MOVW    (R7), DESC                          ; 0568
                                              50 DD 000C1 5$:     PUSHL   RTN_STATUS                          ; 0571
                                        10    AC DD 000C3        PUSHL   QUAL_DESC                           ;
                                        08    AE 9F 000C6        PUSHAB  DESC                                ;
                                              02 DD 000C9        PUSHL   #2                                  ;
                                 0003132C     8F DD 000CB        PUSHL   #201516                             ;
```

```
                                68            05 FB 000D1        CALLS    #5, LIB$SIGNAL              ;
                                53      FF5B  CD 9A 000D4 6$:    MOVZBL   NAM+11, R3                  ; 0577
                                              0A 13 000D9        BEQL     7$                          ;
                   00060080     8F        84  AD D3 000DB        BITL     NAM+52, #393344             ; 0580
                                              10 13 000E3        BEQL     8$                          ;
                                          10  AC DD 000E5 7$:    PUSHL    QUAL_DESC                   ; 0582
                                              57 DD 000E8        PUSHL    R7                          ;
                                              02 DD 000EA        PUSHL    #2                          ;
                                  0003132C    8F DD 000EC        PUSHL    #201516                     ;
                                68            04 FB 000F2        CALLS    #4, LIB$SIGNAL              ;
                                52        08  AC D0 000F5 8$:    MOVL     ESL_DESC, R2                ; 0587
                                62            53 B0 000F9        MOVW     R3, (R2)                    ;
                         02     A2      010E  8F B0 000FC        MOVW     #270, 2(R2)                 ; 0588
                                              53 DD 00102        PUSHL    R3                          ; 0595
                       FE9D     CF            01 FB 00104        CALLS    #1, GET_VM                  ;
                         04     A2            50 D0 00109        MOVL     R0, 4(R2)                   ;
          04    B2       08     AE            53 28 0010D        MOVC3    R3, ESA, @4(R2)             ; 0603
                                                 04 00113        RET                                  ; 0606

; Routine Size:  276 bytes,    Routine Base:  $CODE + 005A
```

```
;  615          0607  1 %SBTTL 'SET_UIC_VALUE'
;  616          0608  1 ROUTINE set_uic_value =
;  617          0609  1
;  618          0610  1 !++
;  619          0611  1 !
;  620          0612  1 ! FUNCTIONAL DESCRIPTION:
;  621          0613  1 !
;  622          0614  1 !       This routine places the group and member numbers, given on the
;  623          0615  1 !       line, in the qualifier data-base.
;  624          0616  1 !
;  625          0617  1 ! FORMAL PARAMETERS:
;  626          0618  1 !
;  627          0619  1 !       None
;  628          0620  1 !
;  629          0621  1 ! IMPLICIT INPUTS:
;  630          0622  1 !
;  631          0623  1 !       AP, most notable the offset TPA$L_PARAM, which contains the address
;  632          0624  1 !       of the qualifier data-base.
;  633          0625  1 !
;  634          0626  1 ! IMPLICIT OUTPUTS:
;  635          0627  1 !
;  636          0628  1 !       None
;  637          0629  1 !
;  638          0630  1 ! ROUTINE VALUE:
;  639          0631  1 !
;  640          0632  1 !       True, unless something is screwed up.
;  641          0633  1 !
;  642          0634  1 ! COMPLETION CODES:
;  643          0635  1 !
;  644          0636  1 !       None
;  645          0637  1 !
;  646          0638  1 ! SIDE EFFECTS:
;  647          0639  1 !
;  648          0640  1 !       The UIC fields in the qualifier database are changed.
;  649          0641  1 !
;  650          0642  1 !--
;  651          0643  1
;  652          0644  2 BEGIN
;  653          0645  2
;  654          0646  2 BUILTIN
;  655          0647  2     AP
;  656          0648  2     ;
;  657          0649  2
;  658          0650  2 LOCAL
;  659          0651  2     qualifier : REF $BBLOCK,
;  660          0652  2     number
;  661          0653  2     ;
;  662          0654  2
;  663          0655  2 MAP
;  664          0656  2     AP : REF $BBLOCK
;  665          0657  2     ;
;  666          0658  2
;  667          0659  2
;  668          0660  2
;  669          0661  2 qualifier = .AP[ TPA$L_PARAM ];
;  670          0662  2 number    = .AP[ TPA$L_NUMBER ];
;  671          0663  2
```

```
;  672          0664  2 qualifier[uic_value] = .number;
;  673          0665  2 if .qualifier[uic_group] eql uic$k_wild_group then qualifier[uic_wldgrp] = true;
;  674          0666  2 if .qualifier[uic_member] eql uic$k_wild_member then qualifier[uic_wldmem] = true;
;  675          0667  2
;  676          0668  2 RETURN true;
;  677          0669  2
;  678          0670  1 END;                                    ! End of routine set_uic_value


                                        0000 00000 SET_UIC_VALUE:
                                                   .WORD   Save nothing                          ; 0608
                        50      20  AC  D0 00002            MOVL    32(AP), QUALIFIER            ; 0661
                        51      1C  AC  D0 00006            MOVL    28(AP), NUMBER               ; 0662
                    18  A0          51  D0 0000A            MOVL    NUMBER, 24(QUALIFIER)        ; 0664
                3FFF    8F      1A  A0  B1 0000E            CMPW    26(QUALIFIER), #16383        ; 0665
                                    04  12 00014            BNEQ    1$                           ;
                    1C  A0          02  88 00016            BISB2   #2, 28(QUALIFIER)            ;
                FFFF    8F      18  A0  B1 0001A 1$:        CMPW    24(QUALIFIER), #65535        ; 0666
                                    04  12 00020            BNEQ    2$                           ;
                    1C  A0          01  88 00022            BISB2   #1, 28(QUALIFIER)            ;
                        50          01  D0 00026 2$:        MOVL    #1, R0                       ; 0668
                                        04 00029            RET                                  ; 0670

; Routine Size:  42 bytes,    Routine Base:  $CODE + 016E
```

```
;  680          0671  1 %SBTTL 'CLEAN_UP'
;  681          0672  1 ROUTINE clean_up( qualifier  : REF $BBLOCK,
;  682          0673  1                  rtn_status,
;  683          0674  1                  fab        : REF $BBLOCK,
;  684          0675  1                  xab_holder ) =
;  685          0676  1 !++
;  686          0677  1 !
;  687          0678  1 ! FUNCTIONAL DESCRIPTION:
;  688          0679  1 !
;  689          0680  1 !       clean_up puts the user's FAB back in order -- returns the user's
;  690          0681  1 !       XAB chain to the FAB, if the file was not open when LIB$QUAL_FILE_MATCH
;  691          0682  1 !       was called and then closes the file.  If the status was anything
;  692          0683  1 !       unexpected the condition is signalled.
;  693          0684  1 !
;  694          0685  1 ! FORMAL PARAMETERS:
;  695          0686  1 !
;  696          0687  1 !       fab             :  User's FAB
;  697          0688  1 !       rtn_status      :  Status to exit procedure with
;  698          0689  1 !       xab_holder      :  Pointer to user's xab chain
;  699          0690  1 !
;  700          0691  1 ! IMPLICIT INPUTS:
;  701          0692  1 !
;  702          0693  1 !       None
;  703          0694  1 !
;  704          0695  1 ! IMPLICIT OUTPUTS:
;  705          0696  1 !
;  706          0697  1 !       None
;  707          0698  1 !
;  708          0699  1 ! ROUTINE VALUE:
;  709          0700  1 !
;  710          0701  1 !       rtn_status
;  711          0702  1 !
;  712          0703  1 ! COMPLETION CODES:
;  713          0704  1 !
;  714          0705  1 !       rtn_status
;  715          0706  1 !
;  716          0707  1 ! SIDE EFFECTS:
;  717          0708  1 !
;  718          0709  1 !       None
;  719          0710  1 !
;  720          0711  1 !--
;  721          0712  1
;  722          0713  2 BEGIN
;  723          0714  2
;  724          0715  2 IF NOT .qualifier[ file_open ]
;  725          0716  2 THEN
;  726          0717  3     BEGIN
;  727          0718  3
;  728          0719  3     ! If the file is currently open and was not open before, close it and
;  729          0720  3     ! return the original XAB chain.
;  730          0721  3     !
;  731          0722  3     IF .fab[ FAB$W_IFI ] NEQ 0
;  732          0723  3     THEN $CLOSE( FAB = .fab );
;  733          0724  3
;  734          0725  3     fab[ FAB$L_XAB ] = .xab_holder;
;  735          0726  2     END;
;  736          0727  2
```

```
;  737          0728  3 IF NOT ( .rtn_status OR
;  738          0729  3          .rtn_status EQL LIB$_FILFAIMAT OR
;  739          0730  3          .rtn_status EQL LIB$_QUIPRO )
;  740          0731  2 THEN SIGNAL( .rtn_status );
;  741          0732  2
;  742          0733  2 RETURN .rtn_status;
;  743          0734  2
;  744          0735  1 END;                                    ! End of routine clean_up


                                                        .EXTRN  SYS$CLOSE

                                        0004 00000 CLEAN_UP:
                                                        .WORD   Save R2                         ; 0672
                                50      04   AC  D0 00002      MOVL    QUALIFIER, R0          ; 0715
                             17   1C    A0      04  E0 00006  BBS     #4, 28(R0), 2$         ;
                                52      0C   AC  D0 0000B      MOVL    FAB, R2                ; 0722
                                        02   A2  B5 0000F      TSTW    2(R2)                  ;
                                             09  13 00012      BEQL    1$                     ;
                                52           DD 00014        PUSHL   R2                     ; 0723
                    00000000G   00           01  FB 00016    CALLS   #1, SYS$CLOSE          ;
                          24    A2      10   AC  D0 0001D 1$: MOVL    XAB_HOLDER, 36(R2)     ; 0725
                                52      08   AC  D0 00022 2$: MOVL    RTN_STATUS, R2         ; 0728
                                1B           52  E8 00026    BLBS    R2, 3$                 ;
                    00000000G   8F           52  D1 00029    CMPL    R2, #LIB$_FILFAIMAT    ; 0729
                                             12  13 00030    BEQL    3$                     ;
                    00000000G   8F           52  D1 00032    CMPL    R2, #LIB$_QUIPRO       ; 0730
                                             09  13 00039    BEQL    3$                     ;
                                52           DD 0003B        PUSHL   R2                     ; 0731
                    00000000G   00           01  FB 0003D    CALLS   #1, LIB$SIGNAL         ;
                                50           52  D0 00044 3$: MOVL    R2, R0                 ; 0733
                                             04  00047        RET                            ; 0735

; Routine Size:  72 bytes,    Routine Base:  $CODE + 0198
```

```
;  746          0736  1 %SBTTL 'LEGAL_ANSWER'
;  747          0737  1 ROUTINE legal_answer( desc : REF $BBLOCK ) =
;  748          0738  1
;  749          0739  1 !++
;  750          0740  1 !
;  751          0741  1 ! FUNCTIONAL DESCRIPTION:
;  752          0742  1 !
;  753          0743  1 !       legal_answer checks the user's response to the /CONFIRM prompt to
;  754          0744  1 !       make sure it was a legal response.
;  755          0745  1 !
;  756          0746  1 ! FORMAL PARAMETERS:
;  757          0747  1 !
;  758          0748  1 !       desc    :  Address of descriptor for the user's answer
;  759          0749  1 !
;  760          0750  1 ! IMPLICIT INPUTS:
;  761          0751  1 !
;  762          0752  1 !       None
;  763          0753  1 !
;  764          0754  1 ! IMPLICIT OUTPUTS:
;  765          0755  1 !
;  766          0756  1 !       None
;  767          0757  1 !
;  768          0758  1 ! ROUTINE VALUE:
;  769          0759  1 !
;  770          0760  1 !       SS$_NORMAL      :  positive answer
;  771          0761  1 !       LIB$_NEGANS     :  negative answer
;  772          0762  1 !       LIB$_QUIPRO     :  quit processing
;  773          0763  1 !       LIB$_QUICONACT  :  continue processing, but cease prompting
;  774          0764  1 !
;  775          0765  1 ! COMPLETION CODES:
;  776          0766  1 !
;  777          0767  1 !       None
;  778          0768  1 !
;  779          0769  1 ! SIDE EFFECTS:
;  780          0770  1 !
;  781          0771  1 !       The response string will be up cased.
;  782          0772  1 !
;  783          0773  1 !--
;  784          0774  1
;  785          0775  2 BEGIN
;  786          0776  2
;  787          0777  2 MACRO
;  788          0778  2
;  789          0779  2     ! This macro creates an entry in the answer table.
;  790          0780  2     !
;  791        M 0781  2     table_entry[ ans_str, rtn_val ] =
;  792        M 0782  2             WORD (%CHARCOUNT(ans_str)),  ! max length of valid response
;  793        M 0783  2             UPLIT BYTE ( ans_str ),      ! pointer to answer string
;  794          0784  2             rtn_val %,                   ! return value, if a match occurs
;  795          0785  2
;  796          0786  2     ans_len = 0, 0, 16, 0 %,             ! length field
;  797          0787  2     ans_str = 2, 0, 32, 0 %,             ! string address field
;  798          0788  2     ans_rtn = 6, 0, 32, 0 %              ! return value field
;  799          0789  2     ;
;  800          0790  2
;  801          0791  2
;  802          0792  2 LITERAL
```

```
;   803          0793  2     entry_len  = 10                        ! length table entry in bytes
;   804          0794  2     ;
;   805          0795  2
;   806          0796  2 BIND
;   807        P 0797  2     ans_table = PLIT( table_entry(
;   808        P 0798  2                       'TRUE',  SS$_NORMAL,
;   809        P 0799  2                       'YES',   SS$_NORMAL,
;   810        P 0800  2                       '1',     SS$_NORMAL,
;   811        P 0801  2                       'FALSE', LIB$_NEGANS,
;   812        P 0802  2                       'NO',    LIB$_NEGANS,
;   813        P 0803  2                       '0',     LIB$_NEGANS,
;   814        P 0804  2                       'ALL',   LIB$_QUICONACT,
;   815          0805  2                       'QUIT',  LIB$_QUIPRO )) : bblockvector[,entry_len]
;   816          0806  2     ;
;   817          0807  2
;   818          0808  2
;   819          0809  2 LOCAL
;   820          0810  2     str_len,                                ! Length of user's response string
;   821          0811  2     entry_count                             ! number of entries in the table
;   822          0812  2     ;
;   823          0813  2
;   824          0814  2
;   825          0815  2
;   826          0816  2
;   827          0817  2 ! <CR> defaults to a NO of FALSE; that is, the user does not want this
;   828          0818  2 ! file operated on.
;   829          0819  2 !
;   830          0820  2 IF .desc[ DSC$W_LENGTH ] EQL 0
;   831          0821  2 THEN RETURN LIB$_NEGANS;
;   832          0822  2
;   833          0823  2
;   834          0824  2 ! Up case the string.
;   835          0825  2 !
;   836          0826  2 STR$UPCASE( .desc, .desc );
;   837          0827  2 str_len = .desc[ DSC$W_LENGTH ];
;   838          0828  2
;   839          0829  2
;   840          0830  2 ! See what the user's response was, and return the appropriate value.  First,
;   841          0831  2 ! determine the number of entries in the table and then compare the user's
;   842          0832  2 ! response to the the table entries.
;   843          0833  2 !
;   844          0834  2 entry_count = (.(ans_table - 4) * 4) / entry_len;
;   845          0835  2 INCR i FROM 0 TO .entry_count - 1
;   846          0836  2 DO
;   847          0837  3     IF( CH$EQL( .str_len, .desc[ DSC$A_POINTER ],
;   848          0838  3         MINU( .str_len, .ans_table[ .i, ans_len ]), .ans_table[ .i, ans_str ]))
;   849          0839  2     THEN  RETURN .ans_table[ .i, ans_rtn ];
;   850          0840  2
;   851          0841  2
;   852          0842  2 ! The answer given was not legal.
;   853          0843  2 !
;   854          0844  2 RETURN false;
;   855          0845  2
;   856          C346  1 END;                                        ! End of routine legal_answer
```

```
                                45 55 52 54  001E0 P.AAC:  .ASCII  \TRUE\
                                   53 45 59  001E4 P.AAD:  .ASCII  \YES\
                                      31     001E7 P.AAE:  .ASCII  \1\
                             45 53 4C 41 46  001E8 P.AAF:  .ASCII  \FALSE\
                                   4F 4E     001ED P.AAG:  .ASCII  \NO\
                                      30     001EF P.AAH:  .ASCII  \0\
                                4C 4C 41     001F0 P.AAI:  .ASCII  \ALL\
                                54 49 55 51  001F3 P.AAJ:  .ASCII  \QUIT\
                                             001F7         .BLKB   1
                                   00000014  001F8         .LONG   20
                                       0004  001FC P.AAB:  .WORD   4
                                   00000000' 001FE         .ADDRESS P.AAC
                                   00000001  00202         .LONG   1
                                       0003  00206         .WORD   3
                                   00000000' 00208         .ADDRESS P.AAD
                                   00000001  0020C         .LONG   1
                                       0001  00210         .WORD   1
                                   00000000' 00212         .ADDRESS P.AAE
                                   00000001  00216         .LONG   1
                                       0005  0021A         .WORD   5
                                   00000000' 0021C         .ADDRESS P.AAF
                                   00000000G 00220         .LONG   LIB$_NEGANS
                                       0002  00224         .WORD   2
                                   00000000' 00226         .ADDRESS P.AAG
                                   00000000G 0022A         .LONG   LIB$_NEGANS
                                       0001  0022E         .WORD   1
                                   00000000' 00230         .ADDRESS P.AAH
                                   00000000G 00234         .LONG   LIB$_NEGANS
                                       0003  00238         .WORD   3
                                   00000000' 0023A         .ADDRESS P.AAI
                                   00000000G 0023E         .LONG   LIB$_QUICONACT
                                       0004  00242         .WORD   4
                                   00000000' 00244         .ADDRESS P.AAJ
                                   00000000G 00248         .LONG   LIB$_QUIPRO

                                                  ANS_TABLE=       P.AAB


                                             03FC 00000 LEGAL_ANSWER:
                                                        .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9                ; 0737
                                59       AB    AF  9E 00002        MOVAB   ANS_TABLE, R9
                                57       04    AC  D0 00006        MOVL    DESC, R7                        ; 0820
                                               67  B5 0000A        TSTW    (R7)
                                               08  12 0000C        BNEQ    1$
                                50 00000000G   8F  D0 0000E        MOVL    #LIB$_NEGANS, R0                ; 0821
                                               04  00015        RET
                                               57  DD 00016 1$:    PUSHL   R7                              ; 0826
                                               57  DD 00018        PUSHL   R7
                          00000000G            00  FB 0001A        CALLS   #2, STR$UPCASE
                                               55  3C 00021        MOVZWL  (R7), STR_LEN                   ; 0827
                             50       FC       A9  78 00024        ASHL    #2, ANS_TABLE-4, R0             ; 0834
                             58                50  C7 00029        DIVL3   #10, R0, ENTRY_COUNT
                                               54  CE 0002D        MNEGL   #1, I                           ; 0837
                                               2D  11 00030        BRB     4$
                             56                54  C5 00032 2$:    MULL3   #10, I, R6                      ; 0838
                                               50  D0 00036        MOVL    STR_LEN, R0
                   50              6946        10  ED 00039        CMPZV   #0, #16, ANS_TABLE[R6], R0
```

```
                                                    06  1E 0003F          BGEQU   3$                                          ;
                                                  6946  9F 00041          PUSHAB  ANS_TABLE[R6]                               ;
                                        50          9E  3C 00044          MOVZWL  @(SP)+, R0                                  ;
                                                02 A946  9F 00047 3$:     PUSHAB  ANS_TABLE+2[R6]                             ;
                                        51          9E  D0 0004B          MOVL    @(SP)+, R1                                  ;
        50              00      04      B7          55  2D 0004E          CMPC5   STR_LEN, @4(R7), #0, R0, (R1)               ; 0837
                                                    61      00054                                                              ;
                                                    08  12 00055          BNEQ    4$                                          ;
                                                06 A946  9F 00057          PUSHAB  ANS_TABLE+6[R6]                             ; 0839
                                        50          9E  D0 0005B          MOVL    @(SP)+, R0                                  ;
                                                        04 0005E          RET                                                 ;
                        CF              54          58  F2 0005F 4$:     AOBLSS  ENTRY_COUNT, I, 2$                          ; 0837
                                                    50  D4 00063          CLRL    R0                                          ; 0844
                                                        04 00065          RET                                                 ; 0846

; Routine Size:  102 bytes,    Routine Base:  $CODE + 024C
```

```
;  858     0847  1 %SBTTL 'FIND_RMS_BLOCK'
;  859     0848  1 ROUTINE find_rms_blocks( qualifier     : REF $BBLOCK,
;  860     0849  1                          fab           : REF $BBLOCK,
;  861     0850  1                          xabdat,
;  862     0851  1                          xabpro,
;  863     0852  1                          lcl_xabdat    : REF $BBLOCK,
;  864     0853  1                          lcl_xabpro    : REF $BBLOCK,
;  865     0854  1                          xab_holder )  : NOVALUE =
;  866     0855  1
;  867     0856  1 !++
;  868     0857  1 !
;  869     0858  1 ! FUNCTIONAL DESCRIPTION:
;  870     0859  1 !
;  871     0860  1 !       FIND_RMS_BLOCKS locates the RMS control blocks attached to the
;  872     0861  1 !       user's FAB.  If the file isn't open, local XAB's are used.
;  873     0862  1 !
;  874     0863  1 ! FORMAL PARAMETERS:
;  875     0864  1 !
;  876     0865  1 !       QUALIFIER   :  Pointer to the qualiier data base
;  877     0866  1 !       FAB         :  The user's fab
;  878     0867  1 !       LCL_XABDAT  :  Address of local date XAB, used if file is closed
;  879     0868  1 !       LCL_XABPRO  :  Address of local protection XAB, used if file is closed
;  880     0869  1 !       XAB_HOLDER  :  Pointer to the user's XAB chain
;  881     0870  1 !
;  882     0871  1 ! FORMAL OUTPUTS
;  883     0872  1 !
;  884     0873  1 !       XABDAT      :  Pointer to the date XAB
;  885     0874  1 !       XABPRO      :  Pointer to the protection XAB
;  886     0875  1 !
;  887     0876  1 ! IMPLICIT INPUTS:
;  888     0877  1 !
;  889     0878  1 !       None
;  890     0879  1 !
;  891     0880  1 ! IMPLICIT OUTPUTS:
;  892     0881  1 !
;  893     0882  1 !       None
;  894     0883  1 !
;  895     0884  1 ! ROUTINE VALUE:
;  896     0885  1 !
;  897     0886  1 !       None
;  898     0887  1 !
;  899     0888  1 ! COMPLETION CODES:
;  900     0889  1 !
;  901     0890  1 !       Any RMS code or LIB$INVXAB (Invalid XAB chain)
;  902     0891  1 !
;  903     0892  1 ! SIDE EFFECTS:
;  904     0893  1 !
;  905     0894  1 !       File will be opened, if it is not already open.
;  906     0895  1 !
;  907     0896  1 !--
;  908     0897  1
;  909     0898  2 BEGIN
;  910     0899  2
;  911     0900  2 LOCAL
;  912     0901  2     rtn_status,                     ! Status returned from externla calls
;  913     0902  2     current_xab :                   ! XAB being evaluated
;  914     0903  2         REF $BBLOCK,
```

```
;  915     0904  2      xabdat_fnd,                     ! Boolean, date XAB was located
;  916     0905  2      xabpro_fnd                      ! Boolean, protection XAB was located
;  917     0906  2      ;
;  918     0907  2
;  919     0908  2
;  920     0909  2
;  921     0910  2  ! Is the file already open?
;  922     0911  2  !
;  923     0912  2  IF NOT .qualifier[ file_open ]
;  924     0913  2  THEN
;  925     0914  3      BEGIN
;  926     0915  3
;  927     0916  3      ! Have the FAB point to the local XAB's and save their addresses.
;  928     0917  3      !
;  929     0918  3      fab[ FAB$L_XAB ]          = .lcl_xabdat;
;  930     0919  3      .xabdat                   = .lcl_xabdat;
;  931     0920  3      lcl_xabdat[ XAB$L_NXT ]   = .lcl_xabpro;
;  932     0921  3      .xabpro                   = .lcl_xabpro;
;  933     0922  3
;  934     0923  3
;  935     0924  3      ! Open the file and display the header information.
;  936     0925  3      !
;  937     0926  4      IF NOT (rtn_status = $OPEN( FAB = .fab ))
;  938     0927  3      THEN clean_up( .qualifier, .rtn_status, .fab, .xab_holder );
;  939     0928  3
;  940     0929  3      END
;  941     0930  2  ELSE
;  942     0931  3      BEGIN
;  943     0932  3
;  944     0933  3      ! Assume neither XAB is needed.  Get the first XAB in the chain.
;  945     0934  3      !
;  946     0935  3      xabdat_fnd  = NOT (.qualifier[ before_present ] or .qualifier[ since_present ]);
;  947     0936  3      xabpro_fnd  = NOT .qualifier[ byowner_present ];
;  948     0937  3      current_xab = .fab[ FAB$L_XAB ];
;  949     0938  3
;  950     0939  3
;  951     0940  3      ! For each XAB in the chain, check to see if it is a protection or date XAB.
;  952     0941  3      !
;  953     0942  4      WHILE NOT( .xabdat_fnd and .xabpro_fnd )
;  954     0943  3      DO
;  955     0944  4          BEGIN
;  956     0945  4
;  957     0946  4          ! If we have encountered the end of the XAB chain, then a necessary XAB                    ;
;  958     0947  4          ! was not found.  Signal the condition.
;  959     0948  4          !
;  960     0949  4          IF .currer _xab EQL 0
;  961     0950  4          THEN clean_up( .qualifier, LIB$_INVXAB, .fab, .xab_holder );
;  962     0951  4
;  963     0952  4          IF .current_xab[ XAB$B_COD ] EQL XAB$C_DAT
;  964     0953  4          THEN
;  965     0954  5              BEGIN
;  966     0955  5
;  967     0956  5              ! The date XAB has been found.  Save it's address and note that it
;  968     0957  5              ! was found.
;  969     0958  5              !
;  970     0959  5              .xabdat     = .current_xab;
;  971     0960  5              xabdat_fnd = true;
```

```
4                   END;
4
4               IF .current_xab[ XAB$B_COD ] EQL XAB$C_PRO
4               THEN
5                   BEGIN
5
5                   ! The protection XAB has been found.  Save it's address and note
5                   ! that it was found.
5                   !
5                   .xabpro     = .current_xab;
5                   xabpro_fnd = true;
4                   END;
4
4               ! Get the address of the next XAB in the chain.
4               !
4               current_xab = .current_xab[ XAB$L_NXT ];
4
3               END;                                    ! while loop
2           END;                                        ! wlse statement
1   END;                                                ! End of routine find_rms_blocks


                                                .EXTRN  SYS$OPEN

                                007C 00000 FIND_RMS_BLOCKS:
                                                .WORD   Save R2,R3,R4,R5,R6                         ; 0848
                  56     FEE0   CF   9E 00002   MOVAB   CLEAN_UP, R6                                 ;
                  53       08   AC   D0 00007   MOVL    FAB, R3                                      ; 0918
                  54       04   AC   D0 0000B   MOVL    QUALIFIER, R4                                ; 0912
         31       1C   A4        04   E0 0000F   BBS     #4, 28(R4), 1$                               ;
                  50       14   AC   D0 00014   MOVL    LCL_XABDAT, R0                               ; 0918
             24   A3            50   D0 00018   MOVL    R0, 36(R3)                                   ;
             0C   BC            50   D0 0001C   MOVL    R0, @XABDAT                                  ; 0919
             04   A0       18   AC   D0 00020   MOVL    LCL_XABPRO, 4(R0)                            ; 0920
             10   BC       18   AC   D0 00025   MOVL    LCL_XABPRO, @XABPRO                          ; 0921
                           08   AC   DD 0002A   PUSHL   FAB                                          ; 0926
      00000000G   00            01   FB 0002D   CALLS   #1, SYS$OPEN                                 ;
                  62            50   E8 00034   BLBS    RTN_STATUS, 7$                               ;
                           1C   AC   DD 00037   PUSHL   XAB_HOLDER                                   ; 0927
                           08   AC   DD 0003A   PUSHL   FAB                                          ;
                                50   DD 0003D   PUSHL   RTN_STATUS                                   ;
                                54   DD 0003F   PUSHL   R4                                           ;
                  66            04   FB 00041   CALLS   #4, CLEAN_UP                                 ;
                                     04 00044   RET                                                  ; 0912
   52        64   01            02   EF 00045 1$: EXTZV   #2, #1, (R4), R2                           ; 0935
   50        64   01            03   EF 0004A   EXTZV   #3, #1, (R4), R0                             ;
                  52            50   C8 0004F   BISL2   R0, R2                                       ;
                  52            52   D2 00052   MCOML   R2, XABDAT_FND                               ;
   55    01  A4   01            00   EF 00055   EXTZV   #0, #1, 1(R4), XABPRO_FND                    ; 0936
                  55            55   D2 0005B   MCOML   XABPRO_FND, XABPRO_FND                       ;
                  53       24   A3   D0 0005E   MOVL    36(R3), CURRENT_XAB                          ; 0937
                  03            52   E9 00062 2$: BLBC    XABDAT_FND, 3$                             ; 0942
                  31            55   E8 00065   BLBS    XABPRO_FND, 7$                               ;
                                11   12 00068 3$: BNEQ    4$                                         ; 0949
                           1C   AC   DD 0006A   PUSHL   XAB_HOLDER                                   ; 0950
                           08   AC   DD 0006D   PUSHL   FAB                                          ;
```

```
                          00000000G  8F  DD 00070          PUSHL   #LIB$_INVXAB                          ;
                                     54  DD 00076          PUSHL   R4                                    ;
                              66     04  FB 00078          CALLS   #4, CLEAN_UP                          ;
                              12     63  91 0007B 4$:      CMPB    (CURRENT_XAB), #18                    ; 0952
                                     07  12 0007E          BNEQ    5$                                    ;
                          0C  BC     53  D0 00080          MOVL    CURRENT_XAB, @XABDAT                  ; 0959
                              52     01  D0 00084          MOVL    #1, XABDAT_FND                        ; 0960
                              13     63  91 00087 5$:      CMPB    (CURRENT_XAB), #19                    ; 0963
                                     07  12 0008A          BNEQ    6$                                    ;
                          10  BC     53  D0 0008C          MOVL    CURRENT_XAB, @XABPRO                  ; 0970
                              55     01  D0 00090          MOVL    #1, XABPRO_FND                        ; 0971
                              53  04 A3  D0 00093 6$:      MOVL    4(CURRENT_XAB), CURRENT_XAB           ; 0976
                                     C9  11 00097          BRB     2$                                    ; 0942
                                         04 00099 7$:      RET                                           ; 0980

; Routine Size:  154 bytes,    Routine Base:  $CODE + 02B2
```

```
;  993          0981  1 %SBTTL 'LIB$QUAL_FILE_PARSE'
;  994          0982  1 GLOBAL ROUTINE lib$qual_file_parse ( flags : REF $BBLOCK, context ) =
;  995          0983  1
;  996          0984  1 !++
;  997          0985  1 !
;  998          0986  1 ! FUNCTIONAL DESCRIPTION:
;  999          0987  1 !
; 1000          0988  1 !       This routine parses a selected set of command qualifiers for the
; 1001          0989  1 !       calling utility.  A database, which describes the results of the
; 1002          0990  1 !       parse, is set up for LIB$QUAL_FILE_MATCH to use in determining if a
; 1003          0991  1 !       file matches the necessary criteria.
; 1004          0992  1 !
; 1005          0993  1 ! FORMAL PARAMETERS:
; 1006          0994  1 !
; 1007          0995  1 !       flags           the address of a longword of flag-bits.  Each bit is
; 1008          0996  1 !                       associated with a specific qualifier.  If the bit
; 1009          0997  1 !                       is set, then the qualifier should be parsed.
; 1010          0998  1 !
; 1011          0999  1 ! OUTPUT PARAMETERS
; 1012          1000  1 !
; 1013          1001  1 !       context         the address of a longword to hold the location the
; 1014          1002  1 !                       qualifier data base;  the contents of this longword
; 1015          1003  1 !                       is passed, by the caller, to LIB$QUAL_FILE_MATCH.
; 1016          1004  1 !
; 1017          1005  1 ! IMPLICIT INPUTS:
; 1018          1006  1 !
; 1019          1007  1 !       None
; 1020          1008  1 !
; 1021          1009  1 ! IMPLICIT OUTPUTS:
; 1022          1010  1 !
; 1023          1011  1 !       None
; 1024          1012  1 !
; 1025          1013  1 ! ROUTINE VALUE:
; 1026          1014  1 !
; 1027          1015  1 !       SS$_NORMAL      the routine completed successfully.
; 1028          1016  1 !
; 1029          1017  1 !       LIB$_INVARG     invalid argument; a bit in the flags parameter was set
; 1030          1018  1 !                       and there was no qualifier associated with it.
; 1031          1019  1 !
; 1032          1020  1 ! COMPLETION CODES:
; 1033          1021  1 !
; 1034          1022  1 !       CLI$_INVQUAVAL  an unusable value was given on the command line for
; 1035          1023  1 !                       any of the following qualifiers: /EXCLUDE, /BEFORE,
; 1036          1024  1 !                       /SINCE, /BY_OWNER.  (i.e. /BEFORE=mintchip or /EXCLUDE=A*A.B)
; 1037          1025  1 !
; 1038          1026  1 !       CLI$_CONFQUAL   more than one of the following appeared on the command
; 1039          1027  1 !                       line at the same time: /CREATED, /MODIFIED, /EXPIRED,
; 1040          1028  1 !                       /BACKUP.
; 1041          1029  1 !
; 1042          1030  1 !       Any completion status from LIB$GET_VM.
; 1043          1031  1 !
; 1044          1032  1 ! SIDE EFFECTS:
; 1045          1033  1 !
; 1046          1034  1 !       Virtual memory may be allocated to hold the /EXCLUDE list.
; 1047          1035  1 !
; 1048          1036  1 !--
; 1049          1037  1
```

```
;  1050          1038  2 BEGIN
;  1051          1039  2
;  1052          1040  2 MACRO
;  1053          1041  2
;  1054          1042  2     ! Generate string descriptors
;  1055          1043  2     !
;  1056          1044  2     qdesc[n] = BIND %NAME('qdesc_', n) = $DESCRIPTOR( n ) %
;  1057          1045  2     ;
;  1058          1046  2
;  1059          1047  2
;  1060          1048  2 LITERAL
;  1061          1049  2     max_string   =  1024                ! maximum length of a command line
;  1062          1050  2     ;
;  1063          1051  2
;  1064          1052  2
;  1065          1053  2 ! Declare the string descriptors which contain the qualifier names.
;  1066          1054  2 !
;  1067        P 1055  2 qdesc(
;  1068        P 1056  2         'CONFIRM',                    ! /CONFIRM
;  1069        P 1057  2         'EXCLUDE',                    ! /EXCLUDE
;  1070        P 1058  2         'BEFORE',                     ! /BEFORE
;  1071        P 1059  2         'SINCE',                      ! /SINCE
;  1072        P 1060  2         'CREATED',                    ! /CREATED
;  1073        P 1061  2         'MODIFIED',                   ! /MODIFIED
;  1074        P 1062  2         'EXPIRED',                    ! /EXPIRED
;  1075        P 1063  2         'BACKUP',                     ! /BACKUP
;  1076        P 1064  2         'BY_OWNER'                    ! /BY_OWNER
;  1077          1065  2       );
;  1078          1066  2
;  1079          1067  2
;  1080          1068  2 BUILTIN
;  1081          1069  2     NULLPARAMETER
;  1082          1070  2     ;
;  1083          1071  2
;  1084          1072  2
;  1085          1073  2 LOCAL
;  1086          1074  2     status,                           ! return status from external routines
;  1087          1075  2     qualifier :                       ! qualifier database
;  1088          1076  2         REF $BBLOCK,
;  1089          1077  2     buffer :                          ! buffer for string values
;  1090          1078  2         $BBLOCK[ max_string ],
;  1091          1079  2     cli_desc :                        ! descriptor for values returned by CLI
;  1092          1080  2         $BBLOCK[ dsc$c_vs_bln ],
;  1093          1081  2     excl_desc_blk :                   ! descriptor for file spec given by /EXCLUDE
;  1094          1082  2         REF $BBLOCK,
;  1095          1083  2     last_excl_blk :                   ! pointer to last descriptor in list
;  1096          1084  2         REF $BBLOCK INITIAL( 0 ),
;  1097          1085  2     previous_excl_desc :              ! pointer to previous descriptor in list
;  1098          1086  2         INITIAL( 0 ),
;  1099          1087  2     tpa_param :                       ! parameter block for LIB$TPARSE
;  1100          1088  2         $BBLOCK[ TPA$K_LENGTH0 ]
;  1101          1089  2         INITIAL( REP TPA$K_LENGTH0 OF BYTE( 0 ))
;  1102          1090  2     ;
;  1103          1091  2
;  1104          1092  2
;  1105          1093  2
;  1106          1094  2
```

```
;  1107          1095  2
;  1108          1096  2 ! Are any bits set in the flags parameter which should not be?
;  1109          1097  2 !
;  1110          1098  2 IF .flags[ LIB$V_CQF_UNASSIGNED ] NEQU 0
;  1111          1099  2 THEN RETURN( LIB$_INVARG );
;  1112          1100  2
;  1113          1101  2
;  1114          1102  2 ! The CONTEXT argument MUST be specified.
;  1115          1103  2 !
;  1116          1104  2 IF NULLPARAMETER( 2 )
;  1117          1105  2 THEN RETURN( LIB$_INVARG );
;  1118          1106  2
;  1119          1107  2
;  1120          1108  2 ! Allocate virtual memory for the qualifier data base.  Pass the address
;  1121          1109  2 ! back to the user.
;  1122          1110  2 !
;  1123          1111  2 qualifier = get_zero_vm( qual_blk_len );
;  1124          1112  2 .context  = .qualifier;
;  1125          1113  2
;  1126          1114  2 ! Intialize the descriptor as a dynamic string descriptor which points to
;  1127          1115  2 ! nothing.
;  1128          1116  2 !
;  1129          1117  2 cli_desc[ DSC$W_MAXSTRLEN ]     = 0;
;  1130          1118  2 cli_desc[ DSC$B_DTYPE ]         = DSC$K_DTYPE_T;
;  1131          1119  2 cli_desc[ DSC$B_CLASS ]         = DSC$K_CLASS_D;
;  1132          1120  2 cli_desc[ DSC$A_POINTER ]       = 0;
;  1133          1121  2
;  1134          1122  2 ! Process the /CONFIRM qualifier, if it has been selected. Set a flag, if it is
;  1135          1123  2 ! on the command line.
;  1136          1124  2 !
;  1137          1125  2 IF .flags[ LIB$V_CQF_CONFIRM ]
;  1138          1126  2 THEN
;  1139          1127  2     qualifier[ confirm_present ] = CLI$PRESENT( qdesc_confirm );
;  1140          1128  2
;  1141          1129  2
;  1142          1130  2 ! Process the /EXCLUDE qualifier, if it has been selected.
;  1143          1131  2 !
;  1144          1132  2 IF .flags[ LIB$V_CQF_EXCLUDE ]
;  1145          1133  2 THEN
;  1146          1134  2
;  1147          1135  2     ! For each file specification in the list:
;  1148          1136  2     !
;  1149          1137  2     WHILE CLI$GET_VALUE( qdesc_exclude, cli_desc ) DO
;  1150          1138  3     BEGIN
;  1151          1139  3
;  1152          1140  3         ! Set a flag, if it is on the command line.
;  1153          1141  3         !
;  1154          1142  3         qualifier[ exclude_present ] = true;
;  1155          1143  3
;  1156          1144  3
;  1157          1145  3         ! Allocate some virtual memory to hold the descriptor for the file
;  1158          1146  3         ! specification.  Place it at the end of the exclude list.
;  1159          1147  3         !
;  1160          1148  3         excl_desc_blk = get_zero_vm( excl_blk_len );
;  1161          1149  3
;  1162          1150  3         IF .last_excl_blk EQL 0
;  1163          1151  3         THEN
```

```
;  1164     1152  3            qualifier[ exclude_list ] = .excl_desc_blk
;  1165     1153  3        ELSE
;  1166     1154  4            BEGIN
;  1167     1155  4            last_excl_blk[ next_excl_blk ] = .excl_desc_blk;
;  1168     1156  4            previous_excl_desc = last_excl_blk[ excl_spec_desc ];
;  1169     1157  4            END;
;  1170     1158
;  1171     1159  3        last_excl_blk = .excl_desc_blk;
;  1172     1160
;  1173     1161
;  1174     1162  3        ! Parse the file specifiction and fill in the descriptor.
;  1175     1163  3        !
;  1176     1164  3        parse_excl_spec( cli_desc,
;  1177     1165  3                         excl_desc_blk[ excl_spec_desc ],
;  1178     1166  3                         .previous_excl_desc,
;  1179     1167  3                         qdesc_exclude );
;  1180     1168
;  1181     1169  2    END;                                 ! while loop to get exclude file specs
;  1182     1170  2
;  1183     1171  2
;  1184     1172  2 ! Process the /BEFORE qualifier, if it has been selected.
;  1185     1173  2 !
;  1186     1174  2 IF .flags[ LIB$V_CQF_BEFORE ]
;  1187     1175  2 THEN
;  1188     1176  2
;  1189     1177  2     ! Set a flag, if it appears on the command line; and retreive the value
;  1190     1178  2     !
;  1191     1179  2     IF (qualifier[ before_present ] = CLI$GET_VALUE( qdesc_before, cli_desc ))
;  1192     1180  2     THEN
;  1193     1181  3         BEGIN
;  1194     1182  3
;  1195     1183  3         ! Convert the time into 64-bit format.  Signal, if this didn't work.
;  1196     1184  3         !
;  1197     1185  3         IF NOT LIB$CVT_TIME( cli_desc, qualifier[ before_value ] )
;  1198     1186  3         THEN
;  1199     1187  3             SIGNAL( CLI$_INVQUAVAL, 2, cli_desc, qdesc_before );
;  1200     1188  3
;  1201     1189  3         ! The file header needs to be read; therefore the file must be
;  1202     1190  3         ! opened to evaluate the selection criteria.
;  1203     1191  3         !
;  1204     1192  3         qualifier[ display_hdr ] = true;
;  1205     1193  2         END;
;  1206     1194  2
;  1207     1195  2
;  1208     1196  2 ! Process the /SINCE qualifier, if it has been selected.
;  1209     1197  2 !
;  1210     1198  2 IF .flags[ LIB$V_CQF_SINCE ]
;  1211     1199  2 THEN
;  1212     1200  2
;  1213     1201  2     ! Set a flag, if it appears on the command line; and retreive the value
;  1214     1202  2     !
;  1215     1203  2     IF (qualifier[ since_present ] = CLI$GET_VALUE( qdesc_since, cli_desc ))
;  1216     1204  2     THEN
;  1217     1205  3         BEGIN
;  1218     1206  3
;  1219     1207  3         ! Convert the time into 64-bit format.  Signal, if this didn't work.
;  1220     1208  3         !
```

```
;  1221     1209  3          IF NOT LIB$CVT_TIME( cli_desc, qualifier[ since_value ] )
;  1222     1210  3          THEN
;  1223     1211  3              SIGNAL( CLI$_INVQUAVAL, 2, cli_desc, qdesc_since );
;  1224     1212  3
;  1225     1213  3          qualifier[ display_hdr ] = true;
;  1226     1214  2          END;
;  1227     1215  2
;  1228     1216  2
;  1229     1217  2 ! Process the /CREATED qualifier, if it has been selected. Set a flag, if it is
;  1230     1218  2 ! on the command line.
;  1231     1219  2 !
;  1232     1220  2 IF .flags[ LIB$V_CQF_CREATED ]
;  1233     1221  2 THEN
;  1234     1222  3     BEGIN
;  1235     1223  3     qualifier[ created_present ] = CLI$PRESENT( qdesc_created );
;  1236     1224  3
;  1237     1225  3     qualifier[ display_hdr ] = .qualifier[ created_present ] OR .qualifier[ display_hdr ];
;  1238     1226  2     END;
;  1239     1227  2
;  1240     1228  2
;  1241     1229  2 ! Process the /MODIFIED qualifier, if it has been selected. Set a flag, if it
;  1242     1230  2 ! is on the command line.
;  1243     1231  2 !
;  1244     1232  2 IF .flags[ LIB$V_CQF_MODIFIED ]
;  1245     1233  2 THEN
;  1246     1234  3     BEGIN
;  1247     1235  3     qualifier[ modified_present ] = CLI$PRESENT( qdesc_modified );
;  1248     1236  3
;  1249     1237  3     qualifier[ display_hdr ] = .qualifier[ modified_present ] OR .qualifier[ display_hdr ];
;  1250     1238  2     END;
;  1251     1239  2
;  1252     1240  2
;  1253     1241  2 ! Process the /EXPIRED qualifier, if it has been selected. Set a flag, if it is
;  1254     1242  2 ! on the command line.
;  1255     1243  2 !
;  1256     1244  2 IF .flags[ LIB$V_CQF_EXPIRED ]
;  1257     1245  2 THEN
;  1258     1246  3     BEGIN
;  1259     1247  3     qualifier[ expired_present ] = CLI$PRESENT( qdesc_expired );
;  1260     1248  3
;  1261     1249  3     qualifier[ display_hdr ] = .qualifier[ expired_present ] OR .qualifier[ display_hdr ];
;  1262     1250  2     END;
;  1263     1251  2
;  1264     1252  2
;  1265     1253  2 ! Process the /BACKUP qualifier, if it has been selected. Set a flag, if it is
;  1266     1254  2 ! on the command line.
;  1267     1255  2 !
;  1268     1256  2 IF .flags[ LIB$V_CQF_BACKUP ]
;  1269     1257  2 THEN
;  1270     1258  3     BEGIN
;  1271     1259  3     qualifier[ backup_present ] = CLI$PRESENT( qdesc_backup );
;  1272     1260  3
;  1273     1261  3     qualifier[ display_hdr ] = .qualifier[ backup_present ] OR .qualifier[ display_hdr ];
;  1274     1262  2     END;
;  1275     1263  2
;  1276     1264  2
;  1277     1265  2 ! Make sure that only one of /CREATED, /MODIFIED, /EXPIRED and /BACKUP was
```

```
;  1278          1266  2 ! given on the command line.  If none were specified default to /CREATED.
;  1279          1267  2 ! Check to be sure that /BEFORE or /SINCE was specified if one of the date
;  1280          1268  2 ! qualifiers was given.
;  1281          1269  2 !
;  1282          1270  2 CASE
;  1283          1271  2     .qualifier[ created_present ]  +
;  1284          1272  2     .qualifier[ modified_present ] +
;  1285          1273  2     .qualifier[ expired_present ]  +
;  1286          1274  2     .qualifier[ backup_present ]
;  1287          1275  2 FROM 0 TO 1 OF SET
;  1288          1276  2
;  1289          1277  2     [ 0 ]         : qualifier[ created_present ] = true;
;  1290          1278  3     [ 1 ]         : IF NOT (.qualifier[ since_present] OR .qualifier[ before_present])
;  1291          1279  2                   THEN SIGNAL( CLI$_QUALMISS, 1, $DESCRIPTOR('/BEFORE or /SINCE') );
;  1292          1280  2     [ OUTRANGE ] : SIGNAL( CLI$_CONFQUAL );
;  1293          1281  2 TES;
;  1294          1282  2
;  1295          1283  2 ! Process the /BY_OWNER qualifier, if it has been selected.
;  1296          1284  2 !
;  1297          1285  2 IF .flags[ LIB$V_CQF_BYOWNER ]
;  1298          1286  2 THEN
;  1299          1287  2
;  1300          1288  2     ! Set a flag, if it is  on the command line. Get the UIC or default it.
;  1301          1289  2     !
;  1302          1290  3     IF (qualifier[ byowner_present ] = CLI$PRESENT( qdesc_by_owner ))
;  1303          1291  2     THEN
;  1304          1292  3         BEGIN
;  1305          1293  3
;  1306          1294  3         IF NOT CLI$GET_VALUE( qdesc_by_owner, cli_desc )
;  1307          1295  3         THEN
;  1308          1296  4             BEGIN
;  1309          1297  4
;  1310          1298  4             ! Declare and initialize the item list needed for $GETJPI.
;  1311          1299  4             !
;  1312          1300  4             LOCAL
;  1313          1301  4                 item_list : VECTOR[ 4 ]
;  1314          1302  4                             INITIAL( WORD( 4, JPI$_UIC ), 0, 0, 0);
;  1315          1303  4                             item_list[ 1 ] = qualifier[ uic_value ];
;  1316          1304  4
;  1317          1305  4             ! No UIC was given; use the current process's UIC.  Since the GETJPI
;  1318          1306  4             ! is being done for the current process it will be synchronous.  If
;  1319          1307  4             ! $GETJPI becomes asynchronous for the current process, a $WAITFR
;  1320          1308  4             ! must be placed after the $GETJPI call.
;  1321          1309  4             !
;  1322          1310  5             IF NOT( status = $GETJPI( ITMLST = item_list ))
;  1323          1311  4             THEN
;  1324          1312  4                 SIGNAL( .status );
;  1325          1313  4
;  1326          1314  4             qualifier[ uic_wldmem ] = false;
;  1327          1315  4             qualifier[ uic_wldgrp ] = false;
;  1328          1316  4             END
;  1329          1317  4
;  1330          1318  3         ELSE
;  1331          1319  4             BEGIN
;  1332          1320  4
;  1333          1321  4             ! A value was given on the command line.  Prepare to parse it.
;  1334          1322  4             ! Initialize the TPARSE parameter block.  The PARAM field of the
```

```
;  1335     1323  4              ! will contain the address of the qualifier data base, so that
;  1336     1324  4              ! the action routine may access the structure.
;  1337     1325  4              !
;  1338     1326  4              tpa_param[ TPA$L_COUNT ]     = TPA$K_COUNTO;
;  1339     1327  4              tpa_param[ TPA$L_STRINGCNT ] = .cli_desc[ DSC$W_LENGTH ];
;  1340     1328  4              tpa_param[ TPA$L_STRINGPTR ] = .cli_desc[ DSC$A_POINTER ];
;  1341     1329  4              tpa_param[ TPA$L_PARAM ]     = .qualifier;
;  1342     1330  4
;  1343     1331  4              ! Parse the UIC and check the validaty of the results.
;  1344     1332  4              !
;  1345     1333  4              status = LIB$TPARSE( tpa_param, cq_uic_states, cq_uic_keys );
;  1346     1334  4              IF NOT .status
;  1347     1335  4              THEN
;  1348     1336  4                  IF .status NEQ SS$_NOSUCHID
;  1349     1337  4                  THEN
;  1350     1338  4                      SIGNAL( CLI$_INVQUAVAL, 2, cli_desc, qdesc_by_owner )
;  1351     1339  4                  ELSE
;  1352     1340  4                      SIGNAL( CLI$_NOSUCHID );
;  1353     1341  3              END;
;  1354     1342  3
;  1355     1343  3          qualifier[ display_hdr ] = .qualifier[ byowner_present ] OR .qualifier[ display_hdr ];
;  1356     1344  2          END;
;  1357     1345  2
;  1358     1346  2
;  1359     1347  2 RETURN SS$_NORMAL;
;  1360     1348  2
;  1361     1349  1 END;                                   ! End of routine lib$qual_file_parse


                                                00034C P.AAL:   .ASCII  \CONFIRM\                       ;
                                                00353          .BLKB   1
                                  00000007      00354 P.AAK:   .LONG   7                               ;
                                  00000000'     00358          .ADDRESS P.AAL                          ;
                                                0035C P.AAN:   .ASCII  \EXCLUDE\                       ;
                                                00363          .BLKB   1
                                  00000007      00364 P.AAM:   .LONG   7                               ;
                                  00000000'     00368          .ADDRESS P.AAN                          ;
                                                0036C P.AAP:   .ASCII  \BEFORE\                        ;
                                                00372          .BLKB   2
                                  00000006      00374 P.AAO:   .LONG   6                               ;
                                  00000000'     00378          .ADDRESS P.AAP                          ;
                                                0037C P.AAR:   .ASCII  \SINCE\                         ;
                                                00381          .BLKB   3
                                  00000005      00384 P.AAQ:   .LONG   5                               ;
                                  00000000'     00388          .ADDRESS P.AAR                          ;
                                                0038C P.AAT:   .ASCII  \CREATED\                       ;
                                                00393          .BLKB   1
                                  00000007      00394 P.AAS:   .LONG   7                               ;
                                  00000000'     00398          .ADDRESS P.AAT                          ;
                                                0039C P.AAV:   .ASCII  \MODIFIED\                      ;
                                  00000008      003A4 P.AAU:   .LONG   8                               ;
                                  00000000'     003A8          .ADDRESS P.AAV                          ;
                                                003AC P.AAX:   .ASCII  \EXPIRED\                       ;
                                                003B3          .BLKB   1
                                  00000007      003B4 P.AAW:   .LONG   7                               ;
                                  00000000'     003B8          .ADDRESS P.AAX                          ;
```

```
                                                50  55  4B  43  41  42  003BC P.AAZ:   .ASCII  \BACKUP\                          ;
                                                                        003C2           .BLKB   2
                                                              00000006  003C4 P.AAY:   .LONG   6                                 ;
                                                              00000000' 003C8           .ADDRESS P.AAZ                          ;
                                            52  45  4E  57  4F  5F  59  42  003CC P.ABB:   .ASCII  \BY_OWNER\                    ;
                                                              00000008  003D4 P.ABA:   .LONG   8                                 ;
                                                              00000000' 003D8           .ADDRESS P.ABB                          ;
                                                                    00#   003DC P.ABC:   .BYTE   0[36]                           ;
4E  49  53  2F  20  72  6F  20  45  52  4F  46  45  42  2F  00400 P.ABE:   .ASCII  \/BEFORE or /SINCE\                          ;
                                                          45  43  0040F                                                         ;
                                                                  00411           .BLKB   3                                     ;
                                                              00000011  00414 P.ABD:   .LONG   17                                ;
                                                              00000000' 00418           .ADDRESS P.ABE                          ;
                                                        0304  0004  0041C P.ABF:   .WORD   4, 772                               ;
                                    00000000  00000000  00000000  00420           .LONG   0, 0, 0                               ;

                                                                                  QDESC_CONFIRM=      P.AAK
                                                                                  QDESC_EXCLUDE=      P.AAM
                                                                                  QDESC_BEFORE=       P.AAO
                                                                                  QDESC_SINCE=        P.AAQ
                                                                                  QDESC_CREATED=      P.AAS
                                                                                  QDESC_MODIFIED=     P.AAU
                                                                                  QDESC_EXPIRED=      P.AAW
                                                                                  QDESC_BACKUP=       P.AAY
                                                                                  QDESC_BY_OWNER=     P.ABA
                                                                                          .EXTRN  SYS$GETJPI

                                                                    0FFC 00000            .ENTRY  LIB$QUAL_FILE_PARSE, Save R2,R3,R4,R5,R6,-  ; 0982
                                                                                                  R7,R8,R9,R10,R11
                                                  5B 00000000G  00  9E 00002            MOVAB   CLI$PRESENT, R11                    ;
                                                  5A 00000000G  00  9E 00009            MOVAB   LIB$SIGNAL, R10                     ;
                                                  59        95  AF  9E 00010            MOVAB   QDESC_BY_OWNER, R9                  ;
                                                  5E      FBC4  CE  9E 00014            MOVAB   -1084(SP), SP                       ;
                                                                57  7C 00019            CLRQ    LAST_EXCL_BLK                       ; 1038
                                          10  AE        08    A9                        24  28 0001B            MOVC3   #36, P.ABC, TPA_PARAM       ; 1083
                                                        53          04    AC  D0 00021            MOVL    FLAGS, R3                   ; 1098
                                    00                  63          17    09  ED 00025            CMPZV   #9, #23, (R3), #0           ;
                                                                      0A  12 0002A            BNEQ    1$                          ;
                                                        02          6C  91 0002C            CMPB    (AP), #2                    ; 1104
                                                                    05  1F 0002F            BLSSU   1$                          ;
                                                              08    AC  D5 00031            TSTL    8(AP)                       ;
                                                                    08  12 00034            BNEQ    2$                          ;
                                                  50 00000000G  8F  D0 00036 1$:          MOVL    #LIB$_INVARG, R0            ; 1105
                                                                    04  00003D            RET                                 ;
                                                                    20  DD 0003E 2$:       PUSHL   #32                         ; 1111
                                                FBB0  CF            01  FB 00040            CALLS   #1, GET_ZERO_VM             ;
                                                      56            50  D0 00045            MOVL    R0, QUALIFIER               ;
                                                  08  BC            56  D0 00048            MOVL    QUALIFIER, @CONTEXT         ; 1112
                                                  34  AE  020E0000  8F  D0 0004C            MOVL    #34471936, CLI_DESC         ; 1117
                                                            38      AE  D4 00054            CLRL    CLI_DESC+4                  ; 1120
                                                        0B          63  E9 00057            BLBC    (R3), 3$                    ; 1125
                                                              80    A9  9F 0005A            PUSHAB  QDESC_CONFIRM               ; 1127
                                                        6B          01  FB 0005D            CALLS   #1, CLI$PRESENT             ;
                                    66                  00          50  F0 00060            INSV    R0, #0, #1, (QUALIFIER)     ;
                                                        43  63          01  E1 00065 3$:     BBC     #1, (R3), 4$                ; 1132
                                                              34    AE  9F 00069 4$:       PUSHAB  CLI_DESC                    ; 1137
                                                              90    A9  9F 0006C           PUSHAB  QDESC_EXCLUDE               ;
```

```
                    00000000G  00          02  FB 0006F          CALLS   #2, CLI$GET_VALUE
                               33          50  E9 00076          BLBC    R0, 7$
                               66          02  88 00079          BISB2   #2, (QUALIFIER)                 1142
                                           0C  DD 0007C          PUSHL   #12                             1148
                         FB72  CF          01  FB 0007E          CALLS   #1, GET_ZERO_VM
                               52          50  D0 00083          MOVL    R0, EXCL_DESC_BLK
                                           57  D5 00086          TSTL    LAST_EXCL_BLK                   1150
                                           06  12 00088          BNEQ    5$
                           04  A6          52  D0 0008A          MOVL    EXCL_DESC_BLK, 4(QUALIFIER)     1152
                                           07  11 0008E          BRB     6$
                               67          52  D0 00090 5$:      MOVL    EXCL_DESC_BLK, (LAST_EXCL_BLK)  1155
                               58      04  A7  9E 00093          MOVAB   4(R7), PREVIOUS_EXCL_DESC       1156
                               57          52  D0 00097 6$:      MOVL    EXCL_DESC_BLK, LAST_EXCL_BLK    1159
                                       90  A9  9F 0009A          PUSHAB  QDESC_EXCLUDE                   1165
                                       58  DD 0009D              PUSHL   PREVIOUS_EXCL_DESC              1166
                                       04  A2  9F 0009F          PUSHAB  4(EXCL_DESC_BLK)                1165
                                       40  AE  9F 000A2          PUSHAB  CLI_DESC                        1164
                         FB84  CF          04  FB 000A5          CALLS   #4, PARSE_EXCL_SPEC             1165
                                           BD  11 000AA          BRB     4$                              1137
                    3A         63          02  E1 000AC 7$:      BBC     #2, (R3), 9$                    1174
                                       34  AE  9F 000B0          PUSHAB  CLI_DESC                        1179
                                       A0  A9  9F 000B3          PUSHAB  QDESC_BEFORE
                    00000000G  00          02  FB 000B6          CALLS   #2, CLI$GET_VALUE
                66  01         02          50  F0 000BD          INSV    R0, #2, #1, (QUALIFIER)
                               25          50  E9 000C2          BLBC    R0, 9$
                                       08  A6  9F 000C5          PUSHAB  8(QUALIFIER)                    1185
                                       38  AE  9F 000C8          PUSHAB  CLI_DESC
                    00000000G  00          02  FB 000CB          CALLS   #2, LIB$CVT_TIME
                               11          50  E8 000D2          BLBS    R0, 8$
                                       A0  A9  9F 000D5          PUSHAB  QDESC_BEFORE                    1187
                                       38  AE  9F 000D8          PUSHAB  CLI_DESC
                                           02  DD 000DB          PUSHL   #2
                                0003132C   8F  DD 000DD          PUSHL   #201516
                               6A          04  FB 000E3          CALLS   #4, LIB$SIGNAL
                         1C    A6          20  88 000E6 8$:      BISB2   #32, 28(QUALIFIER)              1192
                    3A         63          03  E1 000EA 9$:      BBC     #3, (R3), 11$                   1198
                                       34  AE  9F 000EE          PUSHAB  CLI_DESC                        1203
                                       B0  A9  9F 000F1          PUSHAB  QDESC_SINCE
                    00000000G  00          02  FB 000F4          CALLS   #2, CLI$GET_VALUE
                66  01         03          50  F0 000FB          INSV    R0, #3, #1, (QUALIFIER)
                               25          50  E9 00100          BLBC    R0, 11$
                                       10  A6  9F 00103          PUSHAB  16(QUALIFIER)                   1209
                                       38  AE  9F 00106          PUSHAB  CLI_DESC
                    00000000G  00          02  FB 00109          CALLS   #2, LIB$CVT_TIME
                               11          50  E8 00110          BLBS    R0, 10$
                                       B0  A9  9F 00113          PUSHAB  QDESC_SINCE                     1211
                                       38  AE  9F 00116          PUSHAB  CLI_DESC
                                           02  DD 00119          PUSHL   #2
                                0003132C   8F  DD 0011B          PUSHL   #201516
                               6A          04  FB 00121          CALLS   #4, LIB$SIGNAL
                         1C    A6          20  88 00124 10$:     BISB2   #32, 28(QUALIFIER)              1213
                    1F         63          04  E1 00128 11$:     BBC     #4, (R3), 12$                   1220
                                       C0  A9  9F 0012C          PUSHAB  QDESC_CREATED                   1223
                               6B          01  FB 0012F          CALLS   #1, CLI$PRESENT
                66         01  04          50  F0 00132          INSV    R0, #4, #1, (QUALIFIER)
                50         66  01          04  EF 00137          EXTZV   #4, #1, (QUALIFIER), R0         1225
                51    1C   A6  01          05  EF 0013C          EXTZV   #5, #1, 28(QUALIFIER), R1
```

```
                                   50          51  88 00142       BISB2   R1, R0                             ;
    1C   A6          01            05          50  F0 00145       INSV    R0, #5, #1, 28(QUALIFIER)          ;
                     1F            63          05  E1 0014B 12$:  BBC     #5, (R3), 13$                      ; 1232
                                          D0   A9  9F 0014F       PUSHAB  QDESC_MODIFIED                     ; 1235
                                   6B          01  FB 00152       CALLS   #1, CLI$PRESENT                    ;
         66          01            05          50  F0 00155       INSV    R0, #5, #1, (QUALIFIER)            ;
         50          66            01          05  EF 0015A       EXTZV   #5, #1, (QUALIFIER), R0            ; 1237
         51     1C   A6            01          05  EF 0015F       EXTZV   #5, #1, 28(QUALIFIER), R1          ;
                                   50          51  88 00165       BISB2   R1, R0                             ;
    1C   A6          01            05          50  F0 00168       INSV    R0, #5, #1, 28(QUALIFIER)          ;
                     1F            63          06  E1 0016E 13$:  BBC     #6, (R3), 14$                      ; 1244
                                          E0   A9  9F 00172       PUSHAB  QDESC_EXPIRED                      ; 1247
                                   6B          01  FB 00175       CALLS   #1, CLI$PRESENT                    ;
         66          01            06          50  F0 00178       INSV    R0, #6, #1, (QUALIFIER)            ;
         50          66            01          06  EF 0017D       EXTZV   #6, #1, (QUALIFIER), R0            ; 1249
         51     1C   A6            01          05  EF 00182       EXTZV   #5, #1, 28(QUALIFIER), R1          ;
                                   50          51  88 00188       BISB2   R1, R0                             ;
    1C   A6          01            05          50  F0 0018B       INSV    R0, #5, #1, 28(QUALIFIER)          ;
                                               63  95 00191 14$:  TSTB    (R3)                               ; 1256
                                               1F  18 00193       BGEQ    15$                                ;
                                          F0   A9  9F 00195       PUSHAB  QDESC_BACKUP                       ; 1259
                                   6B          01  FB 00198       CALLS   #1, CLI$PRESENT                    ;
         66          01            07          50  F0 0019B       INSV    R0, #7, #1, (QUALIFIER)            ;
         50          66            01          07  EF 001A0       EXTZV   #7, #1, (QUALIFIER), R0            ; 1261
         51     1C   A6            01          05  EF 001A5       EXTZV   #5, #1, 28(QUALIFIER), R1          ;
                                   50          51  88 001AB       BISB2   R1, R0                             ;
    1C   A6          01            05          50  F0 001AE       INSV    R0, #5, #1, 28(QUALIFIER)          ;
         50          66            01          04  EF 001B4 15$:  EXTZV   #4, #1, (QUALIFIER), R0            ; 1272
         51          66            01          05  EF 001B9       EXTZV   #5, #1, (QUALIFIER), R1            ;
                                   50          51  C0 001BE       ADDL2   R1, R0                             ;
         51          66            01          06  EF 001C1       EXTZV   #6, #1, (QUALIFIER), R1            ; 1273
                                   50          51  C0 001C6       ADDL2   R1, R0                             ; 1272
         51          66            01          07  EF 001C9       EXTZV   #7, #1, (QUALIFIER), R1            ; 1274
                                   50          51  C0 001CE       ADDL2   R1, R0                             ;
                     01            00          50  CF 001D1       CASEL   R0, #0, #1                         ; 1273
                                 0014        000F     001D5 16$:  .WORD   17$-16$,-                          ;
                                                                          18$-16$                            ;
                                   000312E4    8F  DD 001D9       PUSHL   #201444                            ; 1280
                                   6A          01  FB 001DF       CALLS   #1, LIB$SIGNAL                     ;
                                               1B  11 001E2       BRB     19$                                ;
                                   66          10  88 001E4 17$:  BISB2   #16, (QUALIFIER)                   ; 1277
                                               16  11 001E7       BRB     19$                                ;
                     12            66          03  E0 001E9 18$:  BBS     #3, (QUALIFIER), 19$               ; 1278
                     0E            66          02  E0 001ED       BBS     #2, (QUALIFIER), 19$               ;
                                          40   A9  9F 001F1       PUSHAB  P.ABD                              ; 1279
                                               01  DD 001F4       PUSHL   #1                                 ;
                                   00031344    8F  DD 001F6       PUSHL   #201540                            ;
                                   6A          03  FB 001FC       CALLS   #3, LIB$SIGNAL                     ;
                                   03     01   A3  E8 001FF 19$:  BLBS    1(R3), 21$                         ; 1285
                                             00B1  31 00203 20$:  BRW     26$                                ;
                                               59  DD 00206 21$:  PUSHL   R9                                 ; 1290
                                   6B          01  FB 00208       CALLS   #1, CLI$PRESENT                    ;
    01   A6          01            00          50  F0 0020B       INSV    R0, #0, #1, 1(QUALIFIER)           ;
                                   EF          50  E9 00211       BLBC    R0, 20$                            ;
                                          34   AE  9F 00214       PUSHAB  CLI_DESC                           ; 1294
                                               59  DD 00217       PUSHL   R9                                 ;
                       00000000G   00          02  FB 00219       CALLS   #2, CLI$GET_VALUE                  ;
```

```
                          2D          50  E8 00220          BLBS    R0, 23$                              ;
             6E      48   A9          10  28 00223          MOVC3   #16, P.ABF, ITEM_LIST                ; 1302
                     04   AE      18  A6  9E 00228          MOVAB   24(R6), ITEM_LIST+4                  ; 1303
                                      7E  7C 0022D          CLRQ    -(SP)                                ; 1310
                                      7E  D4 0022F          CLRL    -(SP)                                ;
                                  0C  AE  9F 00231          PUSHAB  ITEM_LIST                            ;
                                      7E  7C 00234          CLRQ    -(SP)                                ;
                                      7E  D4 00236          CLRL    -(SP)                                ;
                  00000000G  00       07  FB 00238          CALLS   #7, SYS$GETJPI                       ;
                             52       50  D0 0023F          MOVL    R0, STATUS                           ;
                             05       52  E8 00242          BLBS    STATUS, 22$                          ;
                                      52  DD 00245          PUSHL   STATUS                               ; 1312
                             6A       01  FB 00247          CALLS   #1, LIB$SIGNAL                       ;
                     1C      A6       03  8A 0024A 22$:     BICB2   #3, 28(QUALIFIER)                    ; 1315
                                      52  11 0024E          BRB     25$                                  ; 1294
                     10      AE       08  D0 00250 23$:     MOVL    #8, TPA_PARAM                        ; 1326
                     18      AE   34  AE  3C 00254          MOVZWL  CLI_DESC, TPA_PARAM+8                ; 1327
                     1C      AE   38  AE  D0 00259          MOVL    CLI_DESC+4, TPA_PARAM+12             ; 1328
                     30      AE       56  D0 0025E          MOVL    QUALIFIER, TPA_PARAM+32              ; 1329
                            00000000' EF  9F 00262          PUSHAB  CQ_UIC_KEYS                          ; 1333
                            00000000' EF  9F 00268          PUSHAB  CQ_UIC_STATES                        ;
                                  18  AE  9F 0026E          PUSHAB  TPA_PARAM                            ;
                  00000000G  00       03  FB 00271          CALLS   #3, LIB$TPARSE                       ;
                             52       50  D0 00278          MOVL    R0, STATUS                           ;
                             24       52  E8 0027B          BLBS    STATUS, 25$                          ; 1334
                  000021EC   8F       52  D1 0027E          CMPL    STATUS, #8684                        ; 1336
                                      12  13 00285          BEQL    24$                                  ;
                                      59  DD 00287          PUSHL   R9                                   ; 1338
                                  38  AE  9F 00289          PUSHAB  CLI_DESC                             ;
                                      02  DD 0028C          PUSHL   #2                                   ;
                            0003132C  8F  DD 0028E          PUSHL   #201516                              ;
                             6A       04  FB 00294          CALLS   #4, LIB$SIGNAL                       ;
                                      09  11 00297          BRB     25$                                  ;
                            0003135C  8F  DD 00299 24$:     PUSHL   #201564                              ; 1340
                             6A       01  FB 0029F          CALLS   #1, LIB$SIGNAL                       ;
        50       01   A6     01       00  EF 002A2 25$:     EXTZV   #0, #1, 1(QUALIFIER), R0             ; 1343
        51       1C   A6     01       05  EF 002A8          EXTZV   #5, #1, 28(QUALIFIER), R1            ;
                             50       51  88 002AE          BISB2   R1, R0                               ;
  1C    A6            01     05       50  F0 002B1          INSV    R0, #5, #1, 28(QUALIFIER)            ;
                             50       01  D0 002B7 26$:     MOVL    #1, R0                               ; 1347
                                          04 002BA          RET                                          ; 1349

; Routine Size:  699 bytes,    Routine Base:  $CODE + 042C
```

```
;  1363          1350  1 %SBTTL 'LIB$CONFIRM_ACT'
;  1364          1351  1 GLOBAL ROUTINE lib$confirm_act( prompt_string, prompt_args, prompt_rtn ) =
;  1365          1352  1
;  1366          1353  1 !++
;  1367          1354  1 !
;  1368          1355  1 ! FUNCTIONAL DESCRIPTION:
;  1369          1356  1 !
;  1370          1357  1 !       lib$confirm_act prompts the user with a given string and evaluates
;  1371          1358  1 !       answer.  A value is returned indicating positivte and negative
;  1372          1359  1 !       responses.  The user may also request to quit processing or continue
;  1373          1360  1 !       processing and cease prompting.
;  1374          1361  1 !
;  1375          1362  1 ! FORMAL PARAMETERS:
;  1376          1363  1 !
;  1377          1364  1 !       prompt_string:  address of a descriptor for an $FAOL control string to
;  1378          1365  1 !                       be used as a /CONFIRM prompt.
;  1379          1366  1 !
;  1380          1367  1 !       prompt_args   :  address of argument list for $FAOL call.  This parameter
;  1381          1368  1 !                       is optional.
;  1382          1369  1 !
;  1383          1370  1 !
;  1384          1371  1 !       prompt_rtn    :  address of user's prompt routine, in case
;  1385          1372  1 !                       LIB$GET_COMMAND is inappropriate.  This parameter is
;  1386          1373  1 !                       optional.  The routine is called with the address of the
;  1387          1374  1 !                       descriptor for the expanded prompt string and the
;  1388          1375  1 !                       address of the answer descriptor.
;  1389          1376  1 !
;  1390          1377  1 ! IMPLICIT INPUTS:
;  1391          1378  1 !
;  1392          1379  1 !       None
;  1393          1380  1 !
;  1394          1381  1 ! IMPLICIT OUTPUTS:
;  1395          1382  1 !
;  1396          1383  1 !       None
;  1397          1384  1 !
;  1398          1385  1 ! ROUTINE VALUE:
;  1399          1386  1 !
;  1400          1387  1 !       SS$_NORMAL      :  positive answer
;  1401          1388  1 !       LIB$_NEGANS     :  negative answer
;  1402          1389  1 !       LIB$_QUIPRO     :  quit processing
;  1403          1390  1 !       LIB$_QUICONACT  :  continue processing, but cease prompting
;  1404          1391  1 !
;  1405          1392  1 ! COMPLETION CODES:
;  1406          1393  1 !
;  1407          1394  1 !       Any $FAOL return or value returned from the caller's prompt
;  1408          1395  1 !       routine.
;  1409          1396  1 !
;  1410          1397  1 ! SIDE EFFECTS:
;  1411          1398  1 !
;  1412          1399  1 !       None
;  1413          1400  1 !
;  1414          1401  1 !--
;  1415          1402  1
;  1416          1403  2 BEGIN
;  1417          1404  2
;  1418          1405  2 BUILTIN
;  1419          1406  2     NULLPARAMETER
```

```
;  1420          1407  2     ;
;  1421          1408  2
;  1422          1409  2 LITERAL
;  1423          1410  2     prompt_max_str = 256,               ! Maximum length o, the /CONF prompt
;  1424          1411  2     answer_max_str = 8                  ! Maximum length of the answer
;  1425          1412  2     ;
;  1426          1413  2
;  1427          1414  2 LOCAL
;  1428          1415  2     rtn_status,                         ! Status returned from external calls
;  1429          1416  2     prompt_buffer :                     ! Buffer for /CONFIRM prompt
;  1430          1417  2         VECTOR [ prompt_max_str, BYTE ],
;  1431          1418  2     prompt_desc :                       ! Descriptor for the prompt string
;  1432          1419  2         $BBLOCK[ DSC$K_S_BLN ],
;  1433          1420  2     answer_buffer :                     ! Buffer for user's answer to /CONFIRM prompt
;  1434          1421  2         VECTOR [ answer_max_str, BYTE ],
;  1435          1422  2     answer_desc :                       ! Descriptor for the answer string
;  1436          1423  2         $BBLOCK[ DSC$K_S_BLN ]
;  1437          1424  2     ;
;  1438          1425  2
;  1439          1426  2
;  1440          1427  2
;  1441          1428  2 ! Initialize descriptors for the prompt and the answer.
;  1442          1429  2 !
;  1443          1430  2 prompt_desc[ DSC$W_LENGTH  ] = prompt_max_str;
;  1444          1431  2 prompt_desc[ DSC$B_DTYPE   ] = DSC$K_DTYPE_T;
;  1445          1432  2 prompt_desc[ DSC$B_CLASS   ] = DSC$K_CLASS_S;
;  1446          1433  2 prompt_desc[ DSC$A_POINTER ] = prompt_buffer;
;  1447          1434  2
;  1448          1435  2 answer_desc[ DSC$B_DTYPE   ] = DSC$K_DTYPE_T;
;  1449          1436  2 answer_desc[ DSC$B_CLASS   ] = DSC$K_CLASS_S;
;  1450          1437  2 answer_desc[ DSC$A_POINTER ] = answer_buffer;
;  1451          1438  2
;  1452          1439  2
;  1453          1440  2 ! Use $FAOL to insert the prompt arguments into the the prompt string.
;  1454          1441  2 !
;  1455        P 1442  3 IF NOT( RTN_STATUS = $FAOL( CTRSTR = .prompt_string,
;  1456        P 1443  3                             OUTLEN = prompt_desc[ DSC$W_LENGTH ],
;  1457        P 1444  3                             OUTBUF = prompt_desc,
;  1458        P 1445  3                             PRMLST = (IF NULLPARAMETER( 2 )
;  1459          1446  3                                      THEN 0 ELSE .prompt_args )))
;  1460          1447  2 THEN
;  1461          1448  2     SIGNAL( .rtn_status );
;  1462          1449  2
;  1463          1450  2
;  1464          1451  2 ! Prompt the user for consent to do whatever to the current file.  If the caller
;  1465          1452  2 ! wishes, do not use LIB$GET_COMMAND.  Do not accept anything except legal
;  1466          1453  2 ! answers!
;  1467          1454  2 !
;  1468          1455  2 DO
;  1469          1456  3     BEGIN
;  1470          1457  3     answer_desc[ DSC$W_LENGTH ] = answer_max_str;
;  1471          1458  4     IF( NULLPARAMETER( 3 ))
;  1472          1459  3     THEN
;  1473          1460  3         rtn_status = LIB$GET_COMMAND( answer_desc,
;  1474          1461  3                                       prompt_desc,
;  1475          1462  3                                       answer_desc[ DSC$W_LENGTH ] )
;  1476          1463  3     ELSE
```

```
;  1477          1464  4          IF NOT( rtn_status = (.prompt_rtn)( prompt_desc, answer_desc ))
;  1478          1465  4          THEN RETURN .rtn_status;
;  1479          1466  4
;  1480          1467  4      ! Does the user wish to quit processing (^Z for an answer)?
;  1481          1468  4      !
;  1482          1469  4      IF .rtn_status EQL RMS$_EOF
;  1483          1470  4      THEN RETURN LIB$_QUIPRO;
;  1484          1471  4
;  1485          1472  4      END
;  1486          1473  2  UNTIL ((rtn_status = legal_answer( answer_desc )) NEQ 0);
;  1487          1474  2
;  1488          1475  2  RETURN .rtn_status;
;  1489          1476  2
;  1490          1477  1  END;                                    ! End of routine lib$confirm_act


                                                        .EXTRN  SYS$FAOL

                                          0004 00000    .ENTRY  LIB$CONFIRM_ACT, Save R2                        ; 1351
                                5E      FEE8    CE  9E 00002    MOVAB   -280(SP), SP                            ;
                           10   AE  010E0100    8F  D0 00007    MOVL    #17694976, PROMPT_DESC                  ; 1430
                           14   AE        18    AE  9E 0000F    MOVAB   PROMPT_BUFFER, PROMPT_DESC+4            ; 1433
                           02   AE       010E   8F  B0 00014    MOVW    #270, ANSWER_DESC+2                     ; 1435
                           04   AE        08    AE  9E 0001A    MOVAB   ANSWER_BUFFER, ANSWER_DESC+4            ; 1437
                                02              6C  91 0001F    CMPB    (AP), #2                                ; 1446
                                                05  1F 00022    BLSSU   1$                                      ;
                                          08    AC  D5 00024    TSTL    8(AP)                                   ;
                                                04  12 00027    BNEQ    2$                                      ;
                                                7E  D4 00029 1$:    CLRL    -(SP)                               ;
                                                03  11 0002B    BRB     3$                                      ;
                                          08    AC  DD 0002D 2$:    PUSHL   PROMPT_ARGS                         ;
                                          14    AE  9F 00030 3$:    PUSHAB  PROMPT_DESC                         ;
                                          18    AE  9F 00033        PUSHAB  PROMPT_DESC                         ;
                                          04    AC  DD 00036        PUSHL   PROMPT_STRING                       ;
                00000000G   00                    04  FB 00039        CALLS   #4, SYS$FAOL                    ;
                            52                    50  D0 00040        MOVL    R0, RTN_STATUS                  ;
                            09                    52  E8 00043        BLBS    RTN_STATUS, 4$                  ;
                                                  52  DD 00046        PUSHL   RTN_STATUS                      ; 1448
                00000000G   00                    01  FB 00048        CALLS   #1, LIB$SIGNAL                  ;
                            6E                    08  B0 0004F 4$:    MOVW    #8, ANSWER_DESC                 ; 1457
                            03                    6C  91 00052        CMPB    (AP), #3                        ; 1458
                                                  05  1F 00055        BLSSU   5$                              ;
                                          0C    AC  D5 00057        TSTL    12(AP)                          ;
                                                14  12 0005A        BNEQ    6$                              ;
                                                5E  DD 0005C 5$:    PUSHL   SP                              ; 1462
                                          14    AE  9F 0005E        PUSHAB  PROMPT_DESC                     ; 1460
                                          08    AE  9F 00061        PUSHAB  ANSWER_DESC                     ;
                00000000G   00                    03  FB 00064        CALLS   #3, LIB$GET_COMMAND             ; 1462
                            52                    50  D0 0006B        MOVL    R0, RTN_STATUS                  ;
                                                  0F  11 0006E        BRB     7$                              ; 1460
                                                5E  DD 00070 6$:    PUSHL   SP                              ; 1464
                                          :4    AE  9F 00072        PUSHAB  PROMPT_DESC                     ;
                                0C        BC      02  FB 00075        CALLS   #2, @PROMPT_RTN                 ;
                                          52      50  D0 00079        MOVL    R0, RTN_STATUS                  ;
                                          1D      52  E9 0007C        BLBC    RTN_STATUS, 9$                  ;
                0001827A    8F                    52  D1 0007F 7$:    CMPL    RTN_STATUS, #98938              ; 1469
```

```
                                        08  12 00086          BNEQ    8$                                          ;
                                50 00000000G  8F  D0 00088          MOVL    #LIB$_QUIPRO, R0                    ; 1470
                                              04  0008F          RET                                         ;
                                        5E    DD 00090 8$:      PUSHL   SP                                  ; 1473
                             FACE   CF        01  FB 00092          CALLS   #1, LEGAL_ANSWER                    ;
                                52            50  D0 00097          MOVL    R0, RTN_STATUS                      ;
                                              B3  13 0009A          BEQL    4$                                  ;
                                50            52  D0 0009C 9$:      MOVL    RTN_STATUS, R0                      ; 1475
                                              04  0009F          RET                                         ; 1477

; Routine Size:  160 bytes,    Routine Base:  $CODE + 06E7
```

```
;  1492          1478  1 %SBTTL 'LIB$QUAL_FILE_MATCH'
;  1493          1479  1 GLOBAL ROUTINE lib$qual_file_match ( context,
;  1494          1480  1                                      user_fab  : REF $BBLOCK,
;  1495          1481  1                                      file_name : REF $BBLOCK,
;  1496          1482  1                                      prompt_string,
;  1497          1483  1                                      prompt_args,
;  1498          1484  1                                      prompt_rtn ) =
;  1499          1485  1
;  1500          1486  1 !++
;  1501          1487  1 !
;  1502          1488  1 ! FUNCTIONAL DESCRIPTION:
;  1503          1489  1 !
;  1504          1490  1 !       lib$qual_file_match determines if a given file matches the selection
;  1505          1491  1 !       criteria given in the command line.
;  1506          1492  1 !
;  1507          1493  1 ! FORMAL PARAMETERS:
;  1508          1494  1 !
;  1509          1495  1 !       context      :  address of longword containing pointer to the qualifier
;  1510          1496  1 !                       data base.
;  1511          1497  1 !
;  1512          1498  1 !       user_fab     :  address of FAB for file to be evaluated.  This FAB must
;  1513          1499  1 !                       point to a valid NAM block. If the user already has the
;  1514          1500  1 !                       the file open and the file header criteria are to be
;  1515          1501  1 !                       evaluated, the appropriate XAB's must be chained to the
;  1516          1502  1 !                       FAB.  If the file is not open when this routine is
;  1517          1503  1 !                       called, then the XAB chain is not necessary, but may be
;  1518          1504  1 !                       present.  This argument is optional; if it is not given
;  1519          1505  1 !                       the file_name parameter MUST be present.  Both arguments
;  1520          1506  1 !                       may not be present att he same time.
;  1521          1507  1 !
;  1522          1508  1 !       file_name    :  address of a descriptor for the file name of the file
;  1523          1509  1 !                       to be processed.  This parameter may be used instead of
;  1524          1510  1 !                       the user_fab argument.  It is optional.
;  1525          1511  1 !                       NOTE: LIB$QUAL_FILE_MATCH does not do either a $PARSE
;  1526          1512  1 !                       or a $SEARCH with the given file name.
;  1527          1513  1 !
;  1528          1514  1 !       prompt_string:  address of a descriptor for an $FAOL control string to
;  1529          1515  1 !                       be used as a /CONFIRM prompt.  This parameter is
;  1530          1516  1 !                       optional, if /CONFIRM is not being processed.
;  1531          1517  1 !
;  1532          1518  1 !       prompt_args  :  address of argument list for $FAOL call.  This
;  1533          1519  1 !                       parameter is optional.
;  1534          1520  1 !
;  1535          1521  1 !       prompt_rtn   :  address of user's prompt routine, in case
;  1536          1522  1 !                       LIB$GET_COMMAND is inappropriate.  This parameter is
;  1537          1523  1 !                       optional.  The routine is called with the address of the
;  1538          1524  1 !                       descriptor for the expanded prompt string and the
;  1539          1525  1 !                       address of the answer descriptor.
;  1540          1526  1 !
;  1541          1527  1 ! IMPLICIT INPUTS:
;  1542          1528  1 !
;  1543          1529  1 !       None
;  1544          1530  1 !
;  1545          1531  1 ! IMPLICIT OUTPUTS:
;  1546          1532  1 !
;  1547          1533  1 !       None
;  1548          1534  1 !
```

```
;  1549          1535  1 ! ROUTINE VALUE:
;  1550          1536  1 !
;  1551          1537  1 !       SS$_NORMAL    :  the file matches the criteria and may be processed.
;  1552          1538  1 !
;  1553          1539  1 !       LIB$_FILFAIMAT: the file failed the evaluation and should not be
;  1554          1540  1 !                         processed.
;  1555          1541  1 !
;  1556          1542  1 !       LIB$QUIPRO    :  quit processing; the user requested the processing
;  1557          1543  1 !                         cease.
;  1558          1544  1 !
;  1559          1545  1 !       LIB$_INVARG   :  caller passed both the user_fab and file_name
;  1560          1546  1 !                         arguments or neither of them or the context parameter
;  1561          1547  1 !                         was not given.
;  1562          1548  1 !
;  1563          1549  1 !       LIB$_INVXAB   :  invalid XAB chain; a neccessary XAB (XABPRO or XABDAT)
;  1564          1550  1 !                         was missing from the opened file's XAB chain.
;  1565          1551  1 !
;  1566          1552  1 !       Any completion code from $OPEN, $FAOL or the user's prompt routine.
;  1567          1553  1 !
;  1568          1554  1 ! COMPLETION CODES:
;  1569          1555  1 !
;  1570          1556  1 !       None
;  1571          1557  1 !
;  1572          1558  1 ! SIDE EFFECTS:
;  1573          1559  1 !
;  1574          1560  1 !       None
;  1575          1561  1 !
;  1576          1562  1 !--
;  1577          1563  1
;  1578          1564  2 BEGIN
;  1579          1565  2
;  1580          1566  2 BUILTIN
;  1581          1567  2     NULLPARAMETER,
;  1582          1568  2     FP
;  1583          1569  2     ;
;  1584          1570  2
;  1585          1571  2 LOCAL
;  1586          1572  2     rtn_status,                          ! Status returned from external calls
;  1587          1573  2     qualifier     :                      ! address of qualifier data base
;  1588          1574  2         REF $BBLOCK,
;  1589          1575  2     rsa_buffer    :                      ! Resultant name buffer
;  1590          1576  2         VECTOR[ NAM$C_MAXRSS, BYTE ],
;  1591          1577  2     esa_buffer    :                      ! Expanded name buffer
;  1592          1578  2         VECTOR[ NAM$C_MAXRSS, BYTE ],
;  1593          1579  2     lcl_xabdat    :                      ! Local date XAB
;  1594          1580  2         $XABDAT(),
;  1595          1581  2     lcl_xabpro    :                      ! Local protection XAB
;  1596          1582  2         $XABPRO(),
;  1597          1583  2     lcl_nam       :                      ! Local name block
;  1598          1584  2         $NAM(),
;  1599          1585  2     lcl_fab       :                      ! Local FAB
;  1600          1586  2         $FAB(),
;  1601          1587  2     fab           :
;  1602          1588  2         REF $BBLOCK,
;  1603          1589  2     nam           :                      ! NAM block
;  1604          1590  2         REF $BBLOCK,
;  1605          1591  2     xabpro        :                      ! Protection XAB
```

```
;  1606          1592  2          REF $BBLOCK,
;  1607          1593  2      xabdat      :                    ! Date XAB
;  1608          1594  2          REF $BBLOCK,
;  1609          1595  2      xab_holder,                      ! temporary pointer to user's XAB
;  1610          1596  2                                       ! chain (used if file is closed)
;  1611          1597  2      file_date   :                    ! Pointer to file selection date
;  1612          1598  2          REF VECTOR,
;  1613          1599  2      qual_date   :                    ! Pointer to command selection date
;  1614          1600  2          REF VECTOR,
;  1615          1601  2      result_desc :                    ! Descriptor for resultant file name
;  1616          1602  2          VECTOR [ 2 ],
;  1617          1603  2      excl_blk    :                    ! Pointer to the current exclude block
;  1618          1604  2          REF $BBLOCK
;  1619          1605  2      ;
;  1620          1606  2
;  1621          1607  2
;  1622          1608  2  ! Establish the condition handler.
;  1623          1609  2  !
;  1624          1610  2  .FP = LIB$SIG_TO_RET;
;  1625          1611  2
;  1626          1612  2  ! Retrieve the address of the qualifier data base. If the caller didn't give it
;  1627          1613  2  ! to us, then give up.
;  1628          1614  2  !
;  1629          1615  2  IF NULLPARAMETER( 1 )
;  1630          1616  2  THEN
;  1631          1617  3      RETURN( LIB$_INVARG )
;  1632          1618  2  ELSE
;  1633          1619  2      qualifier = ..context;
;  1634          1620  2
;  1635          1621  2
;  1636          1622  2  ! If user previously responded that processing be terminated (^Z or QUIT) to
;  1637          1623  2  ! a /CONFIRM prompt, then return LIB$QUIPRO status.
;  1638          1624  2  !
;  1639          1625  2  IF .qualifier[ quit_processing ]
;  1640          1626  2  THEN
;  1641          1627  2      RETURN( LIB$_QUIPRO );
;  1642          1628  2
;  1643          1629  2
;  1644          1630  2  ! Check the mutually exclusive parameters.
;  1645          1631  2  !
;  1646          1632  2  IF NOT NULLPARAMETER( 2 )
;  1647          1633  2  THEN
;  1648          1634  3      BEGIN
;  1649          1635  3
;  1650          1636  3      IF NOT NULLPARAMETER( 3 )
;  1651          1637  3      THEN
;  1652          1638  3          RETURN( LIB$_INVARG );
;  1653          1639  3
;  1654          1640  3      ! Is the file open?
;  1655          1641  3      !
;  1656          1642  3      IF .user_fab[ FAB$W_IFI ] NEQ 0
;  1657          1643  3      THEN
;  1658          1644  3          qualifier[ file_open ] = true
;  1659          1645  3      ELSE
;  1660          1646  4          BEGIN
;  1661          1647  4          qualifier[ file_open ] = false;
;  1662          1648  4          xab_holder = .user_fab[ FAB$L_XAB ];
```

```
;  1663          1649  3          END;
;  1664          1650  3
;  1665          1651  3      ! Remember that it is the caller's FAB, not our own.
;  1666          1652  3      !
;  1667          1653  3      qualifier[ callers_fab ] = true;
;  1668          1654  3      fab = .user_fab;
;  1669          1655  3      END
;  1670          1656  2  ELSE
;  1671          1657  2      IF NOT NULLPARAMETER( 3 )
;  1672          1658  2      THEN
;  1673          1659  3          BEGIN
;  1674          1660  3
;  1675          1661  3          ! Initialize local RMS control blocks.
;  1676          1662  3          !
;  1677        P 1663  3          $NAM_INIT( NAM = lcl_nam,
;  1678        P 1664  3                  RSA = rsa_buffer,
;  1679        P 1665  3                  RSS = NAM$C_MAXRSS,
;  1680        P 1666  3                  ESA = esa_buffer,
;  1681          1667  3                  ESS = NAM$C_MAXRSS );
;  1682          1668  3
;  1683        P 1669  3          $FAB_INIT( FAB = lcl_fab,
;  1684        P 1670  3                  FNS = .file_name[ DSC$W_LENGTH ],
;  1685        P 1671  3                  FNA = .file_name[ DSC$A_POINTER ],
;  1686          1672  3                  NAM = lcl_nam );
;  1687          1673  3
;  1688          1674  3          qualifier[ file_open ]   = false;
;  1689          1675  3          qualifier[ callers_fab ] = false;
;  1690          1676  3          fab = lcl_fab;
;  1691          1677  3
;  1692          1678  3          ! Parse the file spec to be sure it's okay.
;  1693          1679  3          !
;  1694          1680  3          IF NOT (rtn_status = $PARSE( FAB = .fab ))
;  1695          1681  3          THEN
;  1696          1682  3              RETURN .rtn_status;
;  1697          1683  3
;  1698          1684  3          END
;  1699          1685  2      ELSE
;  1700          1686  2
;  1701          1687  2          ! Neither the file name or the FAB was given, and we need at least
;  1702          1688  2          ! one of them.
;  1703          1689  2          !
;  1704          1690  2          RETURN( LIB$_INVARG );
;  1705          1691  2
;  1706          1692  2
;  1707          1693  2
;  1708          1694  2
;  1709          1695  2  ! Evalulate the file header criteria.
;  1710          1696  2  !
;  1711          1697  2  IF .qualifier[ display_hdr ]
;  1712          1698  2  THEN
;  1713          1699  3      BEGIN
;  1714          1700  3
;  1715          1701  3      ! Find the necessary RMS control blocks.
;  1716          1702  3      !
;  1717          1703  3      find_rms_blocks( .qualifier,
;  1718          1704  3                       .fab,
;  1719          1705  3                       xabdat,
```

```
;1720     1706                                xabpro,
;1721     1707                                lcl_xabdat,
;1722     1708                                lcl_xabpro,
;1723     1709                                .xab_holder );
;1724     1710
;1725     1711        ! If /BEFORE or /SINCE were given on the command line, compare the
;1726     1712        ! appropriate file dates to the ones the user requested.
;1727     1713        !
;1728     1714        IF .qualifier[ before_present] OR .qualifier[ since_present ]
;1729     1715        THEN
;1730     1716
;1731     1717            ! Point to the appropriate date in the file's XABDAT
;1732     1718            !
;1733     1719            SELECTONE true OF
;1734     1720            SET
;1735     1721                [ .qualifier[ created_present ]]  : ! Creation date was requested
;1736     1722                                                    file_date = xabdat[ XAB$Q_CDT ];
;1737     1723
;1738     1724                [ .qualifier[ modified_present ]] : ! Revision date was requested
;1739     1725                                                    file_date = xabdat[ XAB$Q_RDT ];
;1740     1726
;1741     1727                [ .qualifier[ expired_present ]]  : ! Expiration date was requested
;1742     1728                                                    file_date = xabdat[ XAB$Q_EDT ];
;1743     1729
;1744     1730                [ .qualifier[ backup_present ]]   : ! Backup date was requested
;1745     1731                                                    file_date = xabdat[ XAB$Q_BDT ];
;1746     1732            TES;
;1747     1733
;1748     1734
;1749     1735    3       ! /BEFORE was on the command line
;1750     1736    3       !
;1751     1737    3       IF .qualifier[ before_present ]
;1752     1738    3       THEN
;1753     1739    4           BEGIN
;1754     1740    4
;1755     1741    4           ! Point to the date given on the command line.
;1756     1742    4           !
;1757     1743    4           qual_date = qualifier[ before_value ];
;1758     1744    4
;1759     1745    4           ! Compare the file's date to the one on the command line.
;1760     1746    4           !
;1761     1747    4           IF .file_date[ 1 ] GTRU .qual_date[ 1 ] OR
;1762     1748    5              (.file_date[ 1 ] EQLU .qual_date[ 1 ] AND .file_date[ 0 ] GEQU .qual_date[ 0 ])
;1763     1749    4           THEN
;1764     1750    4               RETURN clean_up( .qualifier, LIB$_FILFAIMAT, .fab, .xab_holder );
;1765     1751    4
;1766     1752    3           END;
;1767     1753    3
;1768     1754    3
;1769     1755    3       ! /SINCE was on the command line
;1770     1756    3       !
;1771     1757    3       IF .qualifier[ since_present ]
;1772     1758    3       THEN
;1773     1759    4           BEGIN
;1774     1760    4
;1775     1761    4           ! Point to the date given on the command line.
;1776     1762    4           !
```

```
;  1777          1763  4          qual_date = qualifier[ since_value ];
;  1778          1764  4
;  1779          1765  4          ! Compare the file's date to the one on the command line.
;  1780          1766  4          !
;  1781          1767  4          IF .file_date[ 1 ] LSSU .qual_date[ 1 ] OR
;  1782          1768  5             (.file_date[ 1 ] EQLU .qual_date[ 1 ] AND .file_date[ 0 ] LSSU .qual_date[ 0 ])
;  1783          1769  4          THEN
;  1784          1770  4              RETURN clean_up( .qualifier, LIB$_FILFAIMAT, .fab, .xab_holder );
;  1785          1771  4
;  1786          1772  3          END;
;  1787          1773  3
;  1788          1774  3
;  1789          1775  3      ! /BY_OWNER was given on the command line.
;  1790          1776  3      !
;  1791          1777  3      IF .qualifier[ byowner_present ]
;  1792          1778  3      THEN
;  1793          1779  3
;  1794          1780  3          ! Compare the UIC of the file to the UIC given on the command line.
;  1795          1781  3          !
;  1796          1782  5          IF NOT (( .qualifier[ uic_wldgrp ] OR
;  1797          1783  5                  .xabpro[ XAB$W_GRP ] EQL .qualifier[ uic_group ])
;  1798          1784  4                      AND
;  1799          1785  5                  ( .qualifier[ uic_wldmem ] OR
;  1800          1786  4                  .xabpro[ XAB$W_MBM ] EQL .qualifier[ uic_member ]))
;  1801          1787  3          THEN
;  1802          1788  3              RETURN clean_up( .qualifier, LIB$_FILFAIMAT, .fab, .xab_holder );
;  1803          1789  3
;  1804          1790  2      END;                                    ! File header criteria evaluation
;  1805          1791  2
;  1806          1792  2
;  1807          1793  2 ! Evaluate the file name criteria (/EXCLUDE).
;  1808          1794  2 !
;  1809          1795  2 IF .qualifier[ exclude_present ]
;  1810          1796  2 THEN
;  1811          1797  3     BEGIN
;  1812          1798  3
;  1813          1799  3     ! Get the current file's name.  Use the resultant file name, if it is
;  1814          1800  3     ! present (the file was previously opened).  Otherwise, use the expanded
;  1815          1801  3     ! name string.
;  1816          1802  3     !
;  1817          1803  3     nam = .fab[ FAB$L_NAM ];
;  1818          1804  3     IF .nam[ NAM$B_RS[ ] NEQ 0
;  1819          1805  3     THEN
;  1820          1806  4         BEGIN
;  1821          1807  4         result_desc[ 0 ] = .nam[ NAM$B_RSL ];
;  1822          1808  4         result_desc[ 1 ] = .nam[ NAM$L_RSA ];
;  1823          1809  4         END
;  1824          1810  3     ELSE
;  1825          1811  4         BEGIN
;  1826          1812  4         result_desc[ 0 ] = .nam[ NAM$B_ESL ];
;  1827          1813  4         result_desc[ 1 ] = .nam[ NAM$L_ESA ];
;  1828          1814  3         END;
;  1829          1815  3
;  1830          1816  3
;  1831          1817  3     ! Compare the file name to the file specifications in the exclude list.
;  1832          1818  3     ! If a match is found, then LIB$_FILFAIMAT should be returned, indicating
;  1833          1819  3     ! the file is to be excluded.
```

```
;  1834     1820  3      !
;  1835     1821  3      excl_blk = .qualifier[ exclude_list ];
;  1836     1822  3
;  1837     1823  3      WHILE .excl_blk NEQ 0
;  1838     1824  3      DO
;  1839     1825  4          BEGIN
;  1840     1826  4          IF MATCH( result_desc, excl_blk[ excl_spec_desc ] )
;  1841     1827  4          THEN
;  1842     1828  4              RETURN clean_up( .qualifier, LIB$_FILFAIMAT, .fab, .xab_holder );
;  1843     1829  4
;  1844     1830  4          excl_blk = .excl_blk[ next_excl_blk ];
;  1845     1831  3          END;
;  1846     1832  3
;  1847     1833  2      END;                                    ! File name criteria evaluation
;  1848     1834  2
;  1849     1835  2
;  1850     1836  2  ! Evaluate the user selection criteria (/CONFIRM).
;  1851     1837  2  !
;  1852     1838  2  IF .qualifier[ confirm_present ]
;  1853     1839  2  THEN
;  1854     1840  3      BEGIN
;  1855     1841  3
;  1856     1842  3      ! The prompt string must be present if /CONFIRM is to work.
;  1857     1843  3      !
;  1858     1844  3      IF NULLPARAMETER( 4 )
;  1859     1845  3      THEN RETURN clean_up( .qualifier, LIB$_INVARG, .fab, .xab_holder );
;  1860     1846  3
;  1861     1847  3      ! Prompt for the user's opinion about the current file.  If necessary,
;  1862     1848  3      ! turn off further prompting.  Return the user's answer to the caller.
;  1863     1849  3      !
;  1864     1850  3      rtn_status = lib$confirm_act( .prompt_string,
;  1865     1851  4                                    (IF NULLPARAMETER( 5 )
;  1866     1852  3                                        THEN 0 ELSE .prompt_args ),
;  1867     1853  4                                    (IF NULLPARAMETER( 6 )
;  1868     1854  3                                        THEN 0 ELSE .prompt_rtn ));
;  1869     1855  3
;  1870     1856  3      IF .rtn_status EQL LIB$_QUICONACT
;  1871     1857  3      THEN
;  1872     1858  4          BEGIN
;  1873     1859  4          qualifier[ confirm_present ] = false;
;  1874     1860  4          rtn_status = SS$_NORMAL;
;  1875     1861  3          END;
;  1876     1862  3
;  1877     1863  3
;  1878     1864  3      IF .rtn_status EQL LIB$_NEGANS
;  1879     1865  3      THEN
;  1880     1866  3          rtn_status = LIB$_FILFAIMAT;
;  1881     1867  3
;  1882     1868  3
;  1883     1869  3      IF .rtn_status EQL LIB$_QUIPRO
;  1884     1870  3      THEN
;  1885     1871  3          qualifier[ quit_processing ] = true;
;  1886     1872  3
;  1887     1873  3
;  1888     1874  3      RETURN clean_up( .qualifier, .rtn_status, .fab, .xab_holder);
;  1889     1875  2      END;                                    ! Confirm criteria evaluation
;  1890     1876  2
```

```
;  1891          1877  2
;  1892          1878  2 RETURN clean_up( .qualifier, SS$_NORMAL, .fab, .xab_holder);
;  1893          1879  2
;  1894          1880  1 END;                                    ! End of routine lib$qual_file_match


                        00787              .BLKB   1
                     12 00788 P.ABG:       .BYTE   18
                     2C 00789              .BYTE   44
                   0000 0078A              .WORD   0
               00000000 0078C              .LONG   0
                   0000 00790              .WORD   0
                   0000 00792              .WORD   0
               00000000# 00794             .LONG   0[2]
               00000000# 0079C             .LONG   0[2]
               00000000  007A4             .LONG   0
               00000000  007A8             .LONG   0
               00000000# 007AC             .LONG   0[2]
                     13 007B4 P.ABH:       .BYTE   19
                     58 007B5              .BYTE   88
                   0000 007B6              .WORD   0
               00000000 007B8              .LONG   0
                   FFFF 007BC              .WORD   -1
                     00 007BE              .BYTE   0
                     00 007BF              .BYTE   0
              0000 0000 007C0              .WORD   0, 0
                     00 007C4              .BYTE   0
                     00 007C5              .BYTE   0
                   0000 007C6              .WORD   0
               00000000 007C8              .LONG   0
               00000000 007CC              .LONG   0
                   0000 007D0              .WORD   0
                   0000 007D2              .WORD   0
               00000000 007D4              .LONG   0
               00000000 007D8              .LONG   0
                     02 007DC P.ABI:       .BYTE   2
                     60 007DD              .BYTE   96
                     00 007DE              .BYTE   0
                     00 007DF              .BYTE   0
               00000000 007E0              .LONG   0
                     00 007E4              .BYTE   0
                     00 007E5              .BYTE   0
                     00 007E6              .BYTE   0
                     00 007E7              .BYTE   0
               00000000 007E8              .LONG   0
               00000000 007EC              .LONG   0
                   0000# 007F0             .WORD   0[8]
                   0000# 00800             .WORD   0[3]
                   0000# 00806             .WORD   0[3]
               00000000 0080C              .LONG   0
               00000000 00810              .LONG   0
                     00 00814              .BYTE   0
                     00 00815              .BYTE   0
                     00 00816              .BYTE   0
                     00 00817              .BYTE   0
                     00 00818              .BYTE   0
```

```
                00  00819                .BYTE   0
                00# 0081A                .BYTE   0[2]
          00000000  0081C                .LONG   0
          00000000  00820                .LONG   0
          00000000  00824                .LONG   0
          00000000  00828                .LONG   0
          00000000  0082C                .LONG   0
          00000000  00830                .LONG   0
          00000000# 00834                .LONG   0[2]
                03  0083C P.ABJ:         .BYTE   3
                50  0083D                .BYTE   80
              0000  0083E                .WORD   0
          00000000  00840                .LONG   0
          00000000  00844                .LONG   0
          00000000  00848                .LONG   0
          00000000  0084C                .LONG   0
              0000  00850                .WORD   0
                02  00852                .BYTE   2
                00  00853                .BYTE   0
          00000000  00854                .LONG   0
                00  00858                .BYTE   0
                00  00859                .BYTE   0
                00  0085A                .BYTE   0
                02  0085B                .BYTE   2
          00000000  0085C                .LONG   0
          00000000  00860                .LONG   0
          00000000  00864                .LONG   0
          00000000  00868                .LONG   0
          00000000  0086C                .LONG   0
                00  00870                .BYTE   0
                00  00871                .BYTE   0
              0000  00872                .WORD   0
          00000000  00874                .LONG   0
              0000  00878                .WORD   0
                00  0087A                .BYTE   0
                00  0087B                .BYTE   0
          00000000  0087C                .LONG   0
          00000000  00880                .LONG   0
              0000  00884                .WORD   0
                00  00886                .BYTE   0
                00  00887                .BYTE   0
          00000000  00888                .LONG   0

                                  0FFC 00000    .ENTRY  LIB$QUAL_FILE_MATCH, Save R2,R3,R4,R5,R6,-      ; 1479
                                                        R7,R8,R9,R10,R11
                          5B 00000000G  8F  D0 00002    MOVL    #LIB$_QUIPRO, R11
                          5A      FEEF  CF  9E 00009    MOVAB   P.ABG, R10
                          5E      FCBC  CE  9E 0000E    MOVAB   -836(SP), SP
                0118  CE        6A      2C  28 00013    MOVC3   #44, P.ABG, LCL_XABDAT                  ; 1580
0058  8F        00      2C     AA              28  2C 00019    MOVC5   #40, P.ABH, #0, #88, LCL_XABPRO         ; 1582
                                        00C0  CE     00021
                  60    AE      54     AA      0060  8F  28 00024    MOVC3   #96, P.ABI, LCL_NAM             ; 1584
                  10    AE    00B4     CA      0050  8F  28 0002C    MOVC3   #80, P.ABJ, LCL_FAB             ; 1586
                          6D 00000000G  00  9E 00035    MOVAB   LIB$SIG_TO_RET, (FP)                   ; 1610
                                        6C  95 0003C    TSTB    (AP)                                    ; 1615
```

```
                                    29  13 0003E          BEQL    2$                                      ;
                                04  AC  D5 00040          TSTL    4(AP)                                   ;
                                    2'  13 00043          BEQL    2$                                      ;
                            56  04  BC  D0 00045          MOVL    @CONTEXT, QUALIFIER                     ; 1619
                            58  1C  A6  9E 00049          MOVAB   28(QUALIFIER), R8                       ; 1625
                        04  68      06  E1 0004D          BBC     #6, (R8), 1$                            ;
                            50      5B  D0 00051          MOVL    R11, R0                                 ; 1627
                                        04 00054          RET                                             ;
                            02      6C  91 00055 1$:      CMPB    (AP), #2                                ; 1632
                                    2F  1F 00058          BLSSU   6$                                      ;
                                08  AC  D5 0005A          TSTL    8(AP)                                   ;
                                    2A  13 0005D          BEQL    6$                                      ;
                            03      6C  91 0005F          CMPB    (AP), #3                                ; 1636
                                    08  1F 00062          BLSSU   3$                                      ;
                                0C  AC  D5 00064          TSTL    12(AP)                                  ;
                                    03  13 00067          BEQL    3$                                      ;
                                008D    31 00069 2$:      BRW     7$                                      ;
                            50  08  AC  D0 0006C 3$:      MOVL    USER_FAB, R0                            ; 1642
                                02  A0  B5 00070          TSTW    2(R0)                                   ;
                                    05  13 00073          BEQL    4$                                      ;
                            68      10  88 00075          BISB2   #16, (R8)                               ; 1644
                                    07  11 00078          BRB     5$                                      ;
                            68      10  8A 0007A 4$:      BICB2   #16, (R8)                               ; 1647
                            59  24  A0  D0 0007D          MOVL    36(R0), XAB_HOLDER                      ; 1648
                            68      08  88 00081 5$:      BISB2   #8, (R8)                                ; 1653
                            57      50  D0 00084          MOVL    R0, FAB                                 ; 1654
                                    78  11 00087          BRB     8$                                      ; 1632
                            03      6C  91 00089 6$:      CMPB    (AP), #3                                ; 1657
                                    6B  1F 0008C          BLSSU   7$                                      ;
                                0C  AC  D5 0008E          TSTL    12(AP)                                  ;
                                    66  13 00091          BEQL    7$                                      ;
0060    8F      00          6E          00  2C 00093      MOVC5   #0, (SP), #0, #96, $RMS_PTR             ; 1667
                                60  AE      0009A
                        60  AE      6002    8F  B0 0009C      MOVW    #24578, $RMS_PTR
                        62  AE              01  8E 000A2      MNEGB   #1, $RMS_PTR+2
                        64  AE      FF00    CD  9E 000A6      MOVAB   RSA_BUFFER, $RMS_PTR+4
                        6A  AE              01  8E 000AC      MNEGB   #1, $RMS_PTR+10
                        6C  AE      0144    CE  9E 000B0      MOVAB   ESA_BUFFER, $RMS_PTR+12
0050    8F      00          6E          00  2C 000B6      MOVC5   #0, (SP), #0, #80, $RMS_PTR             ; 1672
                                10  AE      000BD
                        10  AE      5003    8F  B0 000BF      MOVW    #20483, $RMS_PTR
                        26  AE              02  90 000C5      MOVB    #2, $RMS_PTR+22
                        2F  AE              02  90 000C9      MOVB    #2, $RMS_PTR+31
                        38  AE      60      AE  9E 000CD      MOVAB   LCL_NAM, $RMS_PTR+40
                            50      0C      AC  D0 000D2      MOVL    FILE_NAME, R0
                        3C  AE      04      A0  D0 000D6      MOVL    4(R0), $RMS_PTR+44
                        44  AE              60  90 000DB      MOVB    (R0), $RMS_PTR+52
                                    68          18  8A 000DF      BICB2   #24, (R8)                       ; 1675
                                    57      10  AE  9E 000E2      MOVAB   LCL_FAB, FAB                    ; 1676
                                                57  DD 000E6      PUSHL   FAB                             ; 1680
                                00000000G   00          01  FB 000E8      CALLS   #1, SYS$PARSE
                                            54          50  D0 000EF      MOVL    R0, RTN_STATUS
                                            0C          54  E8 000F2      BLBS    RTN_STATUS, 8$
                                            50          54  D0 000F5      MOVL    RTN_STATUS, R0          ; 1682
                                                        04  000F8      RET                             ;
                                        50  00000000G  8F  D0 000F9 7$:      MOVL    #LIB$_INVARG, R0        ; 1690
                                                        04  00100      RET                             ;
```

```
                03              68              05  E0 00101 8$:      BBS     #5, (R8), 9$                    ; 1697
                                          0095  31 00105           BRW     18$
                                            59  DD 00108 9$:       PUSHL   XAB_HOLDER                      ; 1709
                                          00C4  CE 9F 0010A        PUSHAB  LCL_XABPRO                      ; 1703
                                          0120  CE 9F 0010E        PUSHAB  LCL_XABDAT
                                            0C  AE 9F 00112        PUSHAB  XABPRO
                                            14  AE 9F 00115        PUSHAB  XABDAT
                                7E              56  7D 00118        MOVQ    QUALIFIER, -(SP)
                          F906  CF              07  FB 0011B        CALLS   #7, FIND_RMS_BLOCKS
                04              66              02  E0 00120        BBS     #2, (QUALIFIER), 10$            ; 1714
                2A              66              03  E1 00124        BBC     #3, (QUALIFIER), 14$
                07              66              04  E1 00128 10$:    BBC     #4, (QUALIFIER), 11$            ; 1721
                53        04    AE              14  C1 0012C        ADDL3   #20, XABDAT, FILE_DATE          ; 1722
                                                1F  11 00131        BRB     14$
                07              66              05  E1 00133 11$:    BBC     #5, (QUALIFIER), 12$            ; 1724
                53        04    AE              0C  C1 00137        ADDL3   #12, XABDAT, FILE_DATE          ; 1725
                                                14  11 0013C        BRB     14$
                07              66              06  E1 0013E 12$:    BBC     #6, (QUALIFIER), 13$            ; 1727
                53        04    AE              1C  C1 00142        ADDL3   #28, XABDAT, FILE_DATE          ; 1728
                                                09  11 00147        BRB     14$
                                                66  95 00149 13$:    TSTB    (QUALIFIER)                     ; 1730
                                                05  18 0014B        BGEQ    14$
                53        04    AE              24  C1 0014D        ADDL3   #36, XABDAT, FILE_DATE          ; 1731
                12              66              02  E1 00152 14$:    BBC     #2, (QUALIFIER), 15$            ; 1737
                                52        08    A6  9E 00156        MOVAB   8(R6), QUAL_DATE                ; 1743
                          04    A2        04    A3  D1 0015A        CMPL    4(FILE_DATE), 4(QUAL_DATE)      ; 1747
                                                75  1A 0015F        BGTRU   22$
                                                05  12 00161        BNEQ    15$                             ; 1748
                                62              63  D1 00163        CMPL    (FILE_DATE), (QUAL_DATE)
                                                6E  1E 00166        BGEQU   22$
                12              66              03  E1 00168 15$:    BBC     #3, (QUALIFIER), 16$            ; 1757
                                52        10    A6  9E 0016C        MOVAB   16(R6), QUAL_DATE               ; 1763
                          04    A2        04    A3  D1 00170        CMPL    4(FILE_DATE), 4(QUAL_DATE)      ; 1767
                                                5F  1F 00175        BLSSU   22$
                                                05  12 00177        BNEQ    16$                             ; 1768
                                62              63  D1 00179        CMPL    (FILE_DATE), (QUAL_DATE)
                                                58  1F 0017C        BLSSU   22$
                                1B        01    A6  E9 0017E 16$:    BLBC    1(QUALIFIER), 18$               ; 1777
                0A              68              01  E0 00182        BBS     #1, (R8), 17$                   ; 1782
                                50              6E  D0 00186        MOVL    XABPRO, R0                      ; 1783
                          1A    A6        0E    A0  B1 00189        CMPW    14(R0), 26(QUALIFIER)
                                                46  12 0018E        BNEQ    22$
                                0A              68  E8 00190 17$:    BLBS    (R8), 18$                       ; 1785
                                50              6E  D0 00193        MOVL    XABPRO, R0                      ; 1786
                          18    A6        0C    A0  B1 00196        CMPW    12(R0), 24(QUALIFIER)
                                                39  12 0019B        BNEQ    22$
                46              66              01  E1 0019D 18$:    BBC     #1, (QUALIFIER), 24$            ; 1795
                                50        28    A7  D0 001A1        MOVL    40(FAB), NAM                    ; 1803
                                          03    A0  95 001A5        TSTB    3(NAM)                          ; 1804
                                                0C  13 001A8        BEQL    19$
                          08    AE        03    A0  9A 001AA        MOVZBL  3(NAM), RESULT_DESC             ; 1807
                          0C    AE        04    A0  D0 001AF        MOVL    4(NAM), RESULT_DESC+4           ; 1808
                                                0A  11 001B4        BRB     20$                             ; 1804
                          08    AE        0B    A0  9A 001B6 19$:    MOVZBL  11(NAM), RESULT_DESC            ; 1812
                          0C    AE        0C    A0  D0 001BB        MOVL    12(NAM), RESULT_DESC+4          ; 1813
                                52        04    A6  D0 001C0 20$:    MOVL    4(QUALIFIER), EXCL_BLK          ; 1821
                                                21  13 001C4 21$:    BEQL    24$                             ; 1823
```

```
                                    04   A2  9F 001C6          PUSHAB  4(EXCL_BLK)                          ; 1826
                                    0C   AE  9F 001C9          PUSHAB  RESULT_DESC
                          00000000G  00      02 FB 001CC          CALLS   #2, MATCH
                                     0C      50 E9 001D3          BLBC    R0, 23$
                                0280 8F      BB 001D6 22$:        PUSHR   #^M<R7,R9>                      ; 1828
                             00000000G 8F    DD 001DA             PUSHL   #LIB$_FILFAIMAT
                                     77      11 001E0             BRB     36$
                                     52      62 D0 001E2 23$:     MOVL    (EXCL_BLK), EXCL_BLK            ; 1830
                                             11 001E5             BRB     21$                             ; 1823
                                     71      66 E9 001E7 24$:     BLBC    (QUALIFIER), 37$                ; 1838
                                     04      6C 91 001EA          CMPB    (AP), #4                        ; 1844
                                             05 1F 001ED          BLSSU   25$
                                        10   AC D5 001EF          TSTL    16(AP)
                                             0C 12 001F2          BNEQ    26$
                                0280 8F      BB 001F4 25$:        PUSHR   #^M<R7,R9>                      ; 1845
                             00000000G 8F    DD 001F8             PUSHL   #LIB$_INVARG
                                             61 11 001FE          BRB     38$
                                     06      6C 91 00200 26$:     CMPB    (AP), #6                        ; 1853
                                             05 1F 00203          BLSSU   27$
                                        18   AC D5 00205          TSTL    24(AP)
                                             04 12 00208          BNEQ    28$
                                             7E D4 0020A 27$:     CLRL    -(SP)
                                             03 11 0020C          BRB     29$
                                        18   AC DD 0020E 28$:     PUSHL   PROMPT_RTN                      ; 1854
                                     05      6C 91 00211 29$:     CMPB    (AP), #5                        ; 1851
                                             05 1F 00214          BLSSU   30$
                                        14   AC D5 00216          TSTL    20(AP)
                                             04 12 00219          BNEQ    31$
                                             7E D4 0021B 30$:     CLRL    -(SP)
                                             03 11 0021D          BRB     32$
                                        14   AC DD 0021F 31$:     PUSHL   PROMPT_ARGS                     ; 1852
                                        10   AC DD 00222 32$:     PUSHL   PROMPT_STRING                   ; 1850
                                FC31    CF      03 FB 00225          CALLS   #3, LIB$CONFIRM_ACT
                                        54      50 D0 0022A          MOVL    R0, RTN_STATUS
                             00000000G  8F      54 D1 0022D          CMPL    RTN_STATUS, #LIB$_QUICONACT     ; 1856
                                             06 12 00234          BNEQ    33$
                                     66      01 8A 00236          BICB2   #1, (QUALIFIER)                 ; 1859
                                     54      01 D0 00239          MOVL    #1, RTN_STATUS                  ; 1860
                             00000000G  8F      54 D1 0023C 33$:     CMPL    RTN_STATUS, #LIB$_NEGANS        ; 1864
                                             07 12 00243          BNEQ    34$
                                     54 00000000G 8F  D0 00245          MOVL    #LIB$_FILFAIMAT, RTN_STATUS     ; 1866
                                     5B      54 D1 0024C 34$:     CMPL    RTN_STATUS, R11                 ; 1869
                                             04 12 0024F          BNEQ    35$
                                     68   40 8F 88 00251          BISB2   #64, (R8)                       ; 1871
                                        0290 8F BB 00255 35$:     PUSHR   #^M<R4,R7,R9>                   ; 1874
                                             06 11 00259 36$:     BRB     38$
                                        0280 8F BB 0025B 37$:     PUSHR   #^M<R7,R9>                      ; 1878
                                             01 DD 0025F          PUSHL   #1
                                             56 DD 00261 38$:     PUSHL   QUALIFIER
                                F6A4    CF      04 FB 00263          CALLS   #4, CLEAN_UP
                                                04 00268          RET                                     ; 1880

; Routine Size:  617 bytes,    Routine Base:  $CODE + 088C
```

```
;                1881  1 %SBTTL 'LIB$QUAL_FILE_END'
;                1882  1 GLOBAL ROUTINE lib$qual_file_end( context ) =
;                1883  1
;                1884  1 !++
;                1885  1 !
;                1886  1 ! FUNCTIONAL DESCRIPTION:
;                1887  1 !
;                1888  1 !       Clean up any virtual memory which was allocated while processing the
;                1889  1 !       common file qualifiers.
;                1890  1 !
;                1891  1 ! FORMAL PARAMETERS:
;                1892  1 !
;                1893  1 !       context         address of the pointer to the qualifier data base
;                1894  1 !
;                1895  1 ! IMPLICIT INPUTS:
;                1896  1 !
;                1897  1 !       None
;                1898  1 !
;                1899  1 ! IMPLICIT OUTPUTS:
;                1900  1 !
;                1901  1 !       None
;                1902  1 !
;                1903  1 ! ROUTINE VALUE:
;                1904  1 !
;                1905  1 !       SS$_NORMAL      routine completed successfully
;                1906  1 !
;                1907  1 !       Any status return from LIB$FREE_VM
;                1908  1 !
;                1909  1 ! COMPLETION CODES:
;                1910  1 !
;                1911  1 !       None
;                1912  1 !
;                1913  1 ! SIDE EFFECTS:
;                1914  1 !
;                1915  1 !       None
;                1916  1 !
;                1917  1 !--
;                1918  1
;                1919  2 BEGIN
;                1920  2
;                1921  2 LOCAL
;                1922  2     rtn_status,                         ! status returned by external calls
;                1923  2     qualifier : ref $BBLOCK,            ! qualifier data base
;                1924  2     excl_blk  : ref $BBLOCK,            ! pointer to the current exclude block
;                1925  2     next_blk  : ref $BBLOCK,            ! pointer to the next exclude block
;                1926  2     desc_blk  : ref $BBLOCK             ! pointer to the descriptor for the
;                1927  2     ;                                   !   exclude file spec
;                1928  2
;                1929  2
;                1930  2 ! Locate the qualifier data base and the first exclude block.
;                1931  2 !
;                1932  2 qualifier = ..context;
;                1933  2 excl_blk  = .qualifier[ exclude_list ];
;                1934  2
;                1935  2
;                1936  2 ! Free the virtual memory for each exclude block and exclude file spec.
;                1937  2 !
```

```
;  1953          1938  2 UNTIL .excl_blk EQL 0 DO
;  1954          1939  3     BEGIN
;  1955          1940  3
;  1956          1941  3     ! Locate the exclude file descriptor and the next exclude block in the
;  1957          1942  3     ! list.
;  1958          1943  3     !
;  1959          1944  3     desc_blk = excl_blk[ excl_spec_desc ];
;  1960          1945  3     next_blk = .excl_blk[ next_excl_blk ];
;  1961          1946  3
;  1962          1947  3     ! Get rid of the virtual memory.
;  1963          1948  3     !
;  1964          1949  3     IF NOT ( rtn_status = LIB$FREE_VM( %REF(.desc_blk[ DSC$W_LENGTH ]), desc_blk[ DSC$A_POINTER ] ))
;  1965          1950  3     THEN RETURN .rtn_status;
;  1966          1951  3
;  1967          1952  3     IF NOT ( rtn_status = LIB$FREE_VM( %REF(excl_blk_len), excl_blk ))
;  1968          1953  3     THEN RETURN .rtn_status;
;  1969          1954  3
;  1970          1955  3     ! Point to the next exclude block.
;  1971          1956  3     !
;  1972          1957  3     excl_blk = .next_blk;
;  1973          1958  2     END;
;  1974          1959  2
;  1975          1960  2 ! Free up the memory which holds the qualifier data base and zero the context
;  1976          1961  2 ! pointer.
;  1977          1962  2 !
;  1978          1963  3 IF NOT ( rtn_status = LIB$FREE_VM( %REF(qual_blk_len), qualifier ))
;  1979          1964  2 THEN RETURN .rtn_status;
;  1980          1965  2
;  1981          1966  2 .context = 0;
;  1982          1967  2
;  1983          1968  2 RETURN true;
;  1984          1969  1 END;                                   ! End of routine lib$qual_file_end


                                     001C 00000          .ENTRY  LIB$QUAL_FILE_END, Save R2,R3,R4        ; 1882
        54 00000000G   00    9E 00002          MOVAB   LIB$FREE_VM, R4                                 ;
           5E          00    C2 00009          SUBL2   #12, SP                                         ;
  08    AE       04    BC    D0 0000C          MOVL    @CONTEXT, QUALIFIER                             ; 1932
        50       08    AE    D0 00011          MOVL    QUALIFIER, R0                                   ; 1933
  04    AE       04    A0    D0 00015          MOVL    4(R0), EXCL_BLK                                 ;
        51       04    AE    D0 0001A 1$:      MOVL    EXCL_BLK, R1                                    ; 1938
                       2D    13 0001E          BEQL    2$                                              ;
        52       04    A1    9E 00020          MOVAB   4(R1), DESC_BLK                                 ; 1944
        53             61    D0 00024          MOVL    (R1), NEXT_BLK                                  ; 1945
                 04    A2    9F 00027          PUSHAB  4(DESC_BLK)                                     ; 1949
  04    AE             62    3C 0002A          MOVZWL  (DESC_BLK), 4(SP)                               ;
                 04    AE    9F 0002E          PUSHAB  4(SP)                                           ;
        64             02    FB 00031          CALLS   #2, LIB$FREE_VM                                 ;
        2C             50    E9 00034          BLBC    RTN_STATUS, 3$                                  ;
                 04    AE    9F 00037          PUSHAB  EXCL_BLK                                        ; 1952
  04    AE             0C    D0 0003A          MOVL    #12, 4(SP)                                      ;
                 04    AE    9F 0003E          PUSHAB  4(SP)                                           ;
        64             02    FB 00041          CALLS   #2, LIB$FREE_VM                                 ;
        1C             50    E9 00044          BLBC    RTN_STATUS, 3$                                  ;
```

```
                         04   AE          53  D0 00047          MOVL    NEXT_BLK, EXCL_BLK                      ; 1957
                                          CD  11 0004B          BRB     1$                                      ; 1938
                              08   AE      9F 0004D 2$:      PUSHAB  QUALIFIER                               ; 1963
                         04   AE          20  D0 00050          MOVL    #32, 4(SP)                              ;
                              04   AE      9F 00054          PUSHAB  4(SP)                                   ;
                              64          02  FB 00057          CALLS   #2, LIB$FREE_VM                         ;
                              06          50  E9 0005A          BLBC    RTN_STATUS, 3$                          ;
                              04   BC      D4 0005D          CLRL    @CONTEXT                                ; 1966
                              50          01  D0 00060          MOVL    #1, R0                                  ; 1968
                                          04  00063 3$:       RET                                             ; 1969

; Routine Size:  100 bytes,    Routine Base:  $CODE + 0AF5

; 1985          1970  1
; 1986          1971  1 END
; 1987          1972  0 ELUDOM



                                                                .EXTRN  LIB$SIGNAL

;                               PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  _LIB$KEY0$                      0  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;  _LIB$STATE$                    10  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;  $CODE                        2905  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  .  ABS  .                       0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)


;                               Library Statistics
;
;                                       -------- Symbols --------      Pages       Processing
;       File                            Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]TPAMAC.L32;1        42        19        45          14        00:00.0
;  _$255$DUA28:[SYSLIB]STARLET.L32;1     9776       137         1         581        00:01.2



;                                       COMMAND QUALIFIERS

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

; Size:         2296 code + 619 data bytes
; Run Time:       00:50.4
; Elapsed Time:   01:28.7
; Lines/CPU Min:   2347
```

```
; Lexemes/CPU-Min: 27783
; Memory Used:  333 pages
; Compilation Complete
```

LIBACP
LIS

CALLIMAGE
LIS

DISMOUMSG
LIS

LIBCLICAL
LIS

LIBCOMAT
LIS

FILEDFLT
LIS

LIBASSIGN
LIS

CORMSG
LIS

CVTATB
LIS

LIBCQUAL
LIS

INITMSG
LIS

CREMAPSEC
LIS

READOBJ
LIS

SCREEN
LIS

LIBEXECLI
LIS

MOUNTMSG
LIS

PASMSG
LIS

LIBFILPRO
LIS

LIBNETCON
LIS

LIBEXTCON
LIS

LIBMERGE
LIS

LNKMSG
LIS

REPORTIOE
LIS

LIBUNLFIL
LIS

LIBFIDNAM
LIS

MATCHNAME
LIS

OPCMSG
LIS

PLIMSG
LIS