I 3

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

FILEID**CREATE

N 6

```
  CCCCCCCC  RRRRRRRR    EEEEEEEEEE    AAAAAA    TTTTTTTTTT  EEEEEEEEEE
  CCCCCCCC  RRRRRRRR    EEEEEEEEEE    AAAAAA    TTTTTTTTTT  EEEEEEEEEE
CC          RR      RR  EE          AA      AA      TT      EE
CC          RR      RR  EE          AA      AA      TT      EE
CC          RR      RR  EE          AA      AA      TT      EE
CC          RR      RR  EE          AA      AA      TT      EE
CC          RRRRRRRR    EEEEEEEE    AA      AA      TT      EEEEEEEE
CC          RRRRRRRR    EEEEEEEE    AA      AA      TT      EEEEEEEE
CC          RR  RR      EE          AAAAAAAAAA      TT      EE
CC          RR  RR      EE          AAAAAAAAAA      TT      EE
CC          RR    RR    EE          AA      AA      TT      EE                .....
CC          RR    RR    EE          AA      AA      TT      EE                .....
  CCCCCCCC  RR      RR  EEEEEEEEEE  AA      AA      TT      EEEEEEEEEE        .....
  CCCCCCCC  RR      RR  EEEEEEEEEE  AA      AA      TT      EEEEEEEEEE        .....


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 create ( IDENT = 'V04-000',
;     2          0002  0                  ADDRESSING_MODE(EXTERNAL=GENERAL),
;     3          0003  0                  MAIN = create) =
;     4          0004  1 BEGIN
;     5          0005  1
;     6          0006  1 !
;     7          0007  1 !*****************************************************************************
;     8          0008  1 !*                                                                           *
;     9          0009  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    10          0010  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    11          0011  1 !*  ALL RIGHTS RESERVED.                                                     *
;    12          0012  1 !*                                                                           *
;    13          0013  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;    14          0014  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
;    15          0015  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
;    16          0016  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
;    17          0017  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
;    18          0018  1 !*  TRANSFERRED.                                                             *
;    19          0019  1 !*                                                                           *
;    20          0020  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
;    21          0021  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
;    22          0022  1 !*  CORPORATION.                                                             *
;    23          0023  1 !*                                                                           *
;    24          0024  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
;    25          0025  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*                                                                           *
;    28          0028  1 !*****************************************************************************
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 ! FACILITY:  CREATE Command
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !
;    35          0035  1 !       This utility creates files and directories.
;    36          0036  1 !
;    37          0037  1 ! ENVIRONMENT:
;    38          0038  1 !
;    39          0039  1 !       VAX/VMS operating system. unprivileged user mode,
;    40          0040  1 !
;    41          0041  1 ! AUTHOR:  Greg Robert, Nov 1979
;    42          0042  1 !
;    43          0043  1 ! Modified by:
;    44          0044  1 !
;    45          0045  1 !       V03-005 LMP0150         L. Mark Pilant,         12-Sep-1983  13:58
;    46          0046  1 !               Obtain the protection of the newly created file to apply
;    47          0047  1 !               the /PROTECTION qualifier value correctly.
;    48          0048  1 !
;    49          0049  1 !       V03-004 LMP0140         L. Mark Pilant,         19-Aug-1983  9:37
;    50          0050  1 !               Allow handling of alphanumeric UICs
;    51          0051  1 !
;    52          0052  1 !       V03-003 LMP0133         L. Mark Pilant,         4-Aug-1983  15:10
;    53          0053  1 !               Only set the protection if explicitly requested via the
;    54          0054  1 !               /protection qualifier.
;    55          0055  1 !
;    56          0056  1 !       V03-002 SHZ0002         Stephen H. Zalewski     07-Jul-1983
;    57          0057  1 !               Rewrote the way that the /PROTECTION parsing was done.
```

```
;    58          0058  1 !
;    59          0059  1 !       V03-001 SHZ0001         Stephen H. Zalewski     16-May-1983
;    60          0060  1 !               Modified CREATE to use the new CLI.
;    61          0061  1 !
;    62          0062  1 !--
```

```
;    64          0063  1 LIBRARY 'SYS$LIBRARY:STARLET.L32';      ! VAX/VMS common definitions
;    65          0064  1 LIBRARY 'SYS$LIBRARY:TPAMAC.L32';      ! TPARSE macros
;    66          0065  1
;    67          0066  1
;    68          0067  1 MACRO
;    69          0068  1 !
;    70          0069  1 !       Macro to signal a condition to the handler
;    71          0070  1 !
;    72          0071  1
;    73        M 0072  1     write_message(msg) =
;    74        M 0073  1         SIGNAL(msg                          ! Pass the message code
;    75        M 0074  1         %IF %LENGTH GTR 1                   ! and if more than 1 arg
;    76          0075  1             %THEN ,%REMAINING %FI) %;       ! then the rest too
;    77          0076  1
;    78          0077  1
;    79          0078  1 !
;    80          0079  1 !       Define message codes.
;    81          0080  1 !
;    82          0081  1
;    83        P 0082  1 $shr_msgdef(msg,145,global,
;    84        P 0083  1             (openin,error),           ! Unable to open or connect to SYS$INPUT
;    85        P 0084  1             (readerr,error),          ! error while reading SYS$INPUT
;    86        P 0085  1             (closein,error),          ! Unable to close SYS$INPUT
;    87        P 0086  1             (openout,error),          ! Unable to create or connect to user named output file
;    88        P 0087  1             (writeerr,error),         ! error while writing output file
;    89        P 0088  1             (closeout,error),         ! Unable to close output file
;    90        P 0089  1             (created,info),           ! File or directory created
;    91        P 0090  1             (exists,info),            ! File or directory already exists
;    92        P 0091  1             (syntax,severe),          ! Parse failure
;    93        P 0092  1             (dirnotcre,error),        ! Directory not created
;    94        P 0093  1             (badvalue,error),         ! Bad value given for /version or /volume
;    95          0094  1             );
```

```
;   97          0095  1 !
;   98          0096  1 !       EQUATED SYMBOLS
;   99          0097  1 !
;  100          0098  1
;  101          0099  1 LITERAL
;  102          0100  1     TRUE        = 1,                    ! BOOLEAN TRUE
;  103          0101  1     FALSE       = 0;                    ! BOOLEAN FALSE
;  104          0102  1
;  105          0103  1
;  106          0104  1
;  107          0105  1 !
;  108          0106  1 !       Define CLI qualifier bit flag numbers (see qualifier$flags)
;  109          0107  1 !
;  110          0108  1
;  111          0109  1 LITERAL
;  112        P 0110  1     $EQULST (QUAL_,,0,1,
;  113        P 0111  1         (directory,),           ! /DIRECTORY
;  114        P 0112  1         (owner_uic,),           ! /OWNER_UIC
;  115        P 0113  1         (protection,),          ! /PROTECTION
;  116        P 0114  1         (version_limit,),       ! /VERSION_LIMIT
;  117        P 0115  1         (volume,),              ! /VOLUME
;  118          0116  1         (log,));                ! /LOG
;  119          0117  1
```

```
;  121          0118  1 !
;  122          0119  1 ! Table of contents
;  123          0120  1 !
;  124          0121  1
;  125          0122  1 FORWARD ROUTINE
;  126          0123  1     create,                     ! Main create routine
;  127          0124  1     get_output_file,            ! Get next output file
;  128          0125  1     create_dir,                 ! Create directory initialization
;  129          0126  1     create_file,                ! Create file initialization
;  130          0127  1     copy_in_out,                ! Copy SYS$INPUT to output file
;  131          0128  1     log_results,                ! Inform user of results
;  132          0129  1     get_createqls,              ! Get command qualifiers
;  133          0130  1     owner_uic_parse,            ! Parse /OWNER_UIC value
;  134          0131  1     parse_class,                ! Parse protection of one use class
;  135          0132  1     handler,                    ! Condition handler
;  136          0133  1     fab_error,                  ! FAB error handler
;  137          0134  1     rab_error,                  ! RAB error handler
;  138          0135  1     create_error,               ! LIB$CREATE_DIR error handler
;  139          0136  1     vm_error;                   ! LIB$(GET or FREE)_VM err hdlr
;  140          0137  1
;  141          0138  1
;  142          0139  1 EXTERNAL ROUTINE
;  143          0140  1     lib$tparse,                 ! Parses qualifier values
;  144          0141  1     lib$get_vm,                 ! Gets virtual memory
;  145          0142  1     lib$free_vm,                ! Release virtual memory
;  146          0143  1     lib$create_dir,             ! Creates directory entries
;  147          0144  1     lib$cvt_dtb,                ! Convert decimal value to binary
;  148          0145  1     cli$get_value,              ! Get qualifier from CLI
;  149          0146  1     cli$present,                ! Determine if qualifier is present
;  150          0147  1     sys$setdfprot;              ! Read/set default file protection
;  151          0148  1
```

```
;  153          0149  1 !
;  154          0150  1 ! Storage definitions
;  155          0151  1 !
;  156          0152  1
;  157          0153  1 OWN
;  158          0154  1     qualifier$flags     : BITVECTOR[32] ! Qualifier presence bitmap
;  159          0155  1         INITIAL(0),                     ! Initially clear
;  160          0156  1     create$owner_uic    : INITIAL (0),  ! Contains /OWNER_UIC value
;  161          0157  1     create$protection   : VECTOR[2,WORD]
;  162          0158  1                           INITIAL (0),  ! Contains /PROTECTION value
;  163          0159  1     create$version_limit: INITIAL (0),  ! Contains /VERSION_LIMIT value
;  164          0160  1     create$volume       : INITIAL (0),  ! Contains /VOLUME value
;  165          0161  1
;  166          0162  1     tparse_block        : $BBLOCK [tpa$k_length0]       ! TPARSE
;  167          0163  1                           INITIAL (tpa$k_count0,        !  parameter
;  168          0164  1                                                    !  block
;  169          0165  1                                    tpa$m_blanks OR !
;  170          0166  1                                    tpa$m_abbrev),
;  171          0167  1     worst_error: $BBLOCK[4]                         ! Worst error encountered
;  172          0168  1                  INITIAL(ss$_normal);             ! Initially normal status
```

```
;  174          0169  1 !
;  175          0170  1 ! Define RMS blocks
;  176          0171  1 !
;  177          0172  1
;  178          0173  1 OWN
;  179          0174  1     input_nam_result:                    ! Resultant input name
;  180          0175  1         VECTOR [nam$c_maxrss,BYTE],
;  181          0176  1
;  182          0177  1     input_nam_expanded:                  ! Expanded input name
;  183          0178  1         VECTOR [nam$c_maxrss,BYTE],
;  184          0179  1
;  185        P 0180  1     input_nam:  $NAM(                    ! File name block
;  186        P 0181  1         ESA = input_nam_expanded,        ! File name before open
;  187        P 0182  1         ESS = nam$c_maxrss,
;  188        P 0183  1         RSA = input_nam_result,          ! File name after open
;  189          0184  1         RSS = nam$c_maxrss),
;  190          0185  1
;  191        P 0186  1     input_fab:  $FAB(                    ! FAB for input
;  192        P 0187  1         CTX = msg$_openin,               ! Initialize error message
;  193        P 0188  1         NAM = input_nam,                 ! Address of name block
;  194        P 0189  1         FNM = 'SYS$INPUT',               ! File name
;  195          0190  1         FAC = GET),                      ! Open for input
;  196          0191  1
;  197        P 0192  1     input_rab: $RAB(                     ! RAB for input
;  198        P 0193  1         CTX = msg$_readerr,              ! Specify error message
;  199          0194  1         FAB = input_fab),
;  200          0195  1
;  201          0196  1     output_nam_result:                   ! Resultant output name
;  202          0197  1         VECTOR [nam$c_maxrss,BYTE],
;  203          0198  1
;  204          0199  1     output_nam_expanded:                 ! Expanded output name
;  205          0200  1         VECTOR [nam$c_maxrss,BYTE],
;  206          0201  1
;  207        P 0202  1     output_rlf: $NAM(                    ! Related file name block
;  208        P 0203  1         RSA = output_nam_expanded,       ! File name of last parse
;  209          0204  1         RSS = nam$c_maxrss),
;  210          0205  1
;  211        P 0206  1     output_nam: $NAM(                    ! File name block
;  212        P 0207  1         RLF = output_rlf,                ! Related file name block
;  213        P 0208  1         ESA = output_nam_expanded,       ! File name before open
;  214        P 0209  1         ESS = nam$c_maxrss,
;  215        P 0210  1         RSA = output_nam_result,         ! File name after open
;  216          0211  1         RSS = nam$c_maxrss),
;  217          0212  1
;  218          0213  1     output_xabpro: $XABPRO(),            ! XAB for protection and uic
;  219          0214  1
;  220        P 0215  1     output_xaball: $XABALL(              ! XAB for volume specification
;  221        P 0216  1         ALN = LBN,                       ! Allocate by logical block #
;  222        P 0217  1         AOP = HRD,                       ! Report allocation errors
;  223          0218  1         NXT = output_xabpro),            ! Chain to next XAB
;  224          0219  1
;  225        P 0220  1     output_fab: $FAB(                    ! FAB for output
;  226        P 0221  1         CTX = msg$_openout,              ! Initialize error message
;  227        P 0222  1         FOP = SQO,                       ! Sequential access only
;  228        P 0223  1         NAM = output_nam,                ! Address of name block
;  229          0224  1         RAT = CR),                       ! Define file as line oriented
;  230          0225  1
```

```
;  231          P 0226  1     output_rab: $RAB(                       ! RAB for output
;  232          P 0227  1         CTX = msg$_writeerr,                ! Specify error message
;  233            0228  1         FAB = output_fab);
;  234            0229  1
```

```
;  236          0230  1 !
;  237          0231  1 ! Parse UIC string and store binary value.
;  238          0232  1 !
;  239          0233  1
;  240          0234  1 $INIT_STATE (OWNER_UIC_STB, OWNER_UIC_KTB);
;  241          0235  1
;  242        P 0236  1 $STATE  (,
;  243        P 0237  1          ('PARENT'),
;  244        P 0238  1          (TPA$_IDENT,,,,create$owner_uic)
;  245          0239  1          );
;  246          0240  1
;  247        P 0241  1 $STATE  (end_owner_uic,
;  248        P 0242  1          (TPA$_EOS, TPA$_EXIT)
;  249          0243  1          );
```

```
;  251          0244  1 ROUTINE create =                      ! CREATE Main routine
;  252          0245  1
;  253          0246  1 !++
;  254          0247  1 ! Functional description
;  255          0248  1 !
;  256          0249  1 !       This is the main control routine for the create command.
;  257          0250  1 !       It is called from the command language interpreter to
;  258          0251  1 !       create files and directories.
;  259          0252  1 !
;  260          0253  1 ! Calling sequence
;  261          0254  1 !
;  262          0255  1 !       create() from the Command Language Interpreter
;  263          0256  1 !
;  264          0257  1 ! Input parameters
;  265          0258  1 !
;  266          0259  1 !       None
;  267          0260  1 !
;  268          0261  1 ! Output parameters
;  269          0262  1 !
;  270          0263  1 !       None
;  271          0264  1 !
;  272          0265  1 ! Routine value
;  273          0266  1 !
;  274          0267  1 !       Worst error encountered during processing or SS$_NORMAL.
;  275          0268  1 !
;  276          0269  1 !----
;  277          0270  1
;  278          0271  2 BEGIN
;  279          0272  2
;  280          0273  2 LOCAL
;  281          0274  2     status;                            ! Status return
;  282          0275  2
;  283          0276  2
;  284          0277  2 ENABLE handler;                        ! Set condition handler
;  285          0278  2
;  286          0279  2 get_createqls();                       ! Get command qualifiers
;  287          0280  2
;  288          0281  2
;  289          0282  2 !
;  290          0283  2 ! Begin the main loop of the program. Process each file in the input list
;  291          0284  2 !
;  292          0285  2
;  293          0286  2 WHILE get_output_file () DO            ! For each output file
;  294          0287  3     BEGIN
;  295          0288  3     IF .qualifier$flags[qual_directory]    ! If /DIRECTORY specified
;  296          0289  3         THEN status = create_dir ()        !    then create a directory
;  297          0290  3         ELSE status = create_file ();      !    else create a file.
;  298          0291  3
;  299          0292  4     IF (.qualifier$flags[qual_log] AND     ! If logging requested
;  300          0293  4         .status)                           !    and status is ok
;  301          0294  3         THEN log_results (output_fab);     !    then call the logger.
;  302          0295  2     END;                               ! Loop until WHILE is false
;  303          0296  2
;  304          0297  2 RETURN .worst_error;                   ! Exit with worst error encountered
;  305          0298  1 END;
```

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

                                        .PSECT  _LIB$KEY1$,NOWRT,  SHR,  PIC,1

                                00000 ;TPA$KEYSTO
                                00000 U.2:      .BLKB   0
54  4E  45  52  41  50         00000 ;TPA$KEYST
                                00000 U.4:      .ASCII  \PARENT\                        ;
                             FF 00006           .BYTE   -1                              ;
                             FF 00007 ;TPA$KEYFILL
                                00008 U.8:      .BYTE   -1                              ;

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

                                00000 OWNER_UIC_STB::
                                00000           .BLKB   0
                           0100 00000 ;TPA$TYPE
                                00000 U.5:      .WORD   256                             ;
                           45EC 00002 ;TPA$TYPE
                                00002 U.6:      .WORD   17900                           ;
                     00000000* 00004 ;TPA$ADDR
                                00004 U.7:      .LONG   <<CREATE$OWNER_UIC-U.7>-4>      ;
                                00008 END_OWNER_UIC:
                                00008           .BLKB   0
                           15F7 00008 ;TPA$TYPE
                                00008 U.9:      .WORD   5623                            ;
                           FFFF 0000A ;TPA$TARGET
                                0000A U.10:     .WORD   -1                              ;

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

                                00000 OWNER_UIC_KTB::
                                00000           .BLKB   0
                                00000 ;TPA$KEY0
                                00000 U.1:      .BLKB   0
                           0000* 00000 ;TPA$KEY
                                00000 U.3:      .WORD   <U.2-U.1>                       ;

                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

54  55  50  4E  49  24  53  59  53  00000 P.AAA:  .ASCII  \SYS$INPUT\                    ;

                                        .PSECT  $OWN$,NOEXE,2

                         00000000       00000 QUALIFIER$FLAGS:
                                00000           .LONG   0                               ;
                         00000000       00004 CREATE$OWNER_UIC:
                                00004           .LONG   0                               ;
                         00000000       00008 CREATE$PROTECTION:
                                00008           .LONG   0                               ;
                         00000000       0000C CREATE$VERSION_LIMIT:
                                0000C           .LONG   0                               ;
                         00000000       00010 CREATE$VOLUME:
                                00010           .LONG   0                               ;
                 00000003  00000008     00014 TPARSE_BLOCK:
                                00014           .LONG   8, 3                            ;
```

```
                        0001C           .BLKB   28
00000001                00038 WORST_ERROR:
                                .LONG   1                               ;
                        0003C INPUT_NAM_RESULT:
                                .BLKB   255
                        0013B           .BLKB   1
                        0013C INPUT_NAM_EXPANDED:
                                .BLKB   255
                        0023B           .BLKB   1
                     02 0023C INPUT_NAM:
                                .BYTE   2                               ;
                     60 0023D           .BYTE   96                      ;
                     FF 0023E           .BYTE   -1                      ;
                     00 0023F           .BYTE   0                       ;
00000000'            00240              .ADDRESS INPUT_NAM_RESULT       ;
                     00 00244           .BYTE   0                       ;
                     00 00245           .BYTE   0                       ;
                     FF 00246           .BYTE   -1                      ;
                     00 00247           .BYTE   0                       ;
00000000'            00248              .ADDRESS INPUT_NAM_EXPANDED     ;
00000000             0024C              .LONG   0                       ;
                 0000# 00250            .WORD   0[8]                    ;
                 0000# 00260            .WORD   0[3]                    ;
                 0000# 00266            .WORD   0[3]                    ;
00000000             0026C              .LONG   0                       ;
00000000             00270              .LONG   0                       ;
                     00 00274           .BYTE   0                       ;
                     00 00275           .BYTE   0                       ;
                     00 00276           .BYTE   0                       ;
                     00 00277           .BYTE   0                       ;
                     00 00278           .BYTE   0                       ;
                     00 00279           .BYTE   0                       ;
                     00# 0027A          .BYTE   0[2]                    ;
00000000             0027C              .LONG   0                       ;
00000000             00280              .LONG   0                       ;
00000000             00284              .LONG   0                       ;
00000000             00288              .LONG   0                       ;
00000000             0028C              .LONG   0                       ;
00000000             00290              .LONG   0                       ;
00000000#            00294              .LONG   0[2]                    ;
                     03 0029C INPUT_FAB:
                                .BYTE   3                               ;
                     50 0029D           .BYTE   80                      ;
                   0000 0029E           .WORD   0                       ;
00000000             002A0              .LONG   0                       ;
00000000             002A4              .LONG   0                       ;
00000000             002A8              .LONG   0                       ;
00000000             002AC              .LONG   0                       ;
                   0000 002B0           .WORD   0                       ;
                     02 002B2           .BYTE   2                       ;
                     00 002B3           .BYTE   0                       ;
0091109A             002B4              .LONG   9506970                 ;
                     00 002B8           .BYTE   0                       ;
                     00 002B9           .BYTE   0                       ;
                     00 002BA           .BYTE   0                       ;
                     02 002BB           .BYTE   2                       ;
00000000             002BC              .LONG   0                       ;
```

```
00000000  002C0           .LONG   0
00000000' 002C4           .ADDRESS INPUT_NAM
00000000' 002C8           .ADDRESS P.AAA
00000000  002CC           .LONG   0
       09 002D0           .BYTE   9
       00 002D1           .BYTE   0
     0000 002D2           .WORD   0
00000000  002D4           .LONG   0
     0000 002D8           .WORD   0
       00 002DA           .BYTE   0
       00 002DB           .BYTE   0
00000000  002DC           .LONG   0
00000000  002E0           .LONG   0
     0000 002E4           .WORD   0
       00 002E6           .BYTE   0
       00 002E7           .BYTE   0
00000000  002E8           .LONG   0
       01 002EC INPUT_RAB:
                          .BYTE   1
       44 002ED           .BYTE   68
     0000 002EE           .WORD   0
00000000  002F0           .LONG   0
00000000  002F4           .LONG   0
00000000  002F8           .LONG   0
    0000# 002FC           .WORD   0[3]
     0000 00302           .WORD   0
009110B2  00304           .LONG   9506994
     0000 00308           .WORD   0
       00 0030A           .BYTE   0
       00 0030B           .BYTE   0
     0000 0030C           .WORD   0
     0000 0030E           .WORD   0
00000000  00310           .LONG   0
00000000  00314           .LONG   0
00000000  00318           .LONG   0
00000000  0031C           .LONG   0
       00 00320           .BYTE   0
       00 00321           .BYTE   0
       00 00322           .BYTE   0
       00 00323           .BYTE   0
00000000  00324           .LONG   0
00000000' 00328           .ADDRESS INPUT_FAB
00000000  0032C           .LONG   0
          00330 OUTPUT_NAM_RESULT:
                          .BLKB   255
          0042F           .BLKB   1
          00430 OUTPUT_NAM_EXPANDED:
                          .BLKB   255
          0052F           .BLKB   1
       02 00530 OUTPUT_RLF:
                          .BYTE   2
       60 00531           .BYTE   96
       FF 00532           .BYTE   -1
       00 00533           .BYTE   0
00000000' 00534           .ADDRESS OUTPUT_NAM_EXPANDED
       00 00538           .BYTE   0
       00 00539           .BYTE   0
```

```
      00  0053A            .BYTE    0
      00  0053B            .BYTE    0
00000000  0053C            .LONG    0
00000000  00540            .LONG    0
   0000#  00544            .WORD    0[8]
   0000#  00554            .WORD    0[3]
   0000#  0055A            .WORD    0[3]
00000000  00560            .LONG    0
00000000  00564            .LONG    0
      00  00568            .BYTE    0
      00  00569            .BYTE    0
      00  0056A            .BYTE    0
      00  0056B            .BYTE    0
      00  0056C            .BYTE    0
      00  0056D            .BYTE    0
     00#  0056E            .BYTE    0[2]
00000000  00570            .LONG    0
00000000  00574            .LONG    0
00000000  00578            .LONG    0
00000000  0057C            .LONG    0
00000000  00580            .LONG    0
00000000  00584            .LONG    0
00000000# 00588            .LONG    0[2]
      02  00590 OUTPUT_NAM:
                           .BYTE    2
      60  00591            .BYTE    96
      FF  00592            .BYTE    -1
      00  00593            .BYTE    0
00000000' 00594            .ADDRESS OUTPUT_NAM_RESULT
      00  00598            .BYTE    0
      00  00599            .BYTE    0
      FF  0059A            .BYTE    -1
      00  0059B            .BYTE    0
00000000' 0059C            .ADDRESS OUTPUT_NAM_EXPANDED
00000000' 005A0            .ADDRESS OUTPUT_RLF
   0000#  005A4            .WORD    0[8]
   0000#  005B4            .WORD    0[3]
   0000#  005BA            .WORD    0[3]
00000000  005C0            .LONG    0
00000000  005C4            .LONG    0
      00  005C8            .BYTE    0
      00  005C9            .BYTE    0
      00  005CA            .BYTE    0
      00  005CB            .BYTE    0
      00  005CC            .BYTE    0
      00  005CD            .BYTE    0
     00#  005CE            .BYTE    0[2]
00000000  005D0            .LONG    0
00000000  005D4            .LONG    0
00000000  005D8            .LONG    0
00000000  005DC            .LONG    0
00000000  005E0            .LONG    0
00000000  005E4            .LONG    0
00000000# 005E8            .LONG    0[2]
      13  005F0 OUTPUT_XABPRO:
                           .BYTE    19
      58  005F1            .BYTE    88
```

```
                    0000  005F2          .WORD   0
                00000000  005F4          .LONG   0
                    FFFF  005F8          .WORD   -1
                      00  005FA          .BYTE   0
                      00  005FB          .BYTE   0
            0000    0000  005FC          .WORD   0, 0
                      00  00600          .BYTE   0
                      00  00601          .BYTE   0
                    0000  00602          .WORD   0
                00000000  00604          .LONG   0
                00000000  00608          .LONG   0
                    0000  0060C          .WORD   0
                    0000  0060E          .WORD   0
                00000000  00610          .LONG   0
                00000000  00614          .LONG   0
                          00618          .BLKB   48
                      14  00648 OUTPUT_XABALL:
                                         .BYTE   20
                      20  00649          .BYTE   32
                    0000  0064A          .WORD   0
                00000000' 0064C          .ADDRESS OUTPUT_XABPRO
                      01  00650          .BYTE   1
                      02  00651          .BYTE   2
                    0000  00652          .WORD   0
                00000000  00654          .LONG   0
                00000000  00658          .LONG   0
                    0000  0065C          .WORD   0
                      00  0065E          .BYTE   0
                      00  0065F          .BYTE   0
        0000    0000  0000  00660          .WORD   0, 0, 0
                    0000  00666          .WORD   0
                      03  00668 OUTPUT_FAB:
                                         .BYTE   3
                      50  00669          .BYTE   80
                    0000  0066A          .WORD   0
                00000040  0066C          .LONG   o4
                00000000  00670          .LONG   0
                00000000  00674          .LONG   0
                00000000  00678          .LONG   0
                    0000  0067C          .WORD   0
                      02  0067E          .BYTE   2
                      00  0067F          .BYTE   0
                009110A2  00680          .LONG   9506978
                      00  00684          .BYTE   0
                      00  00685          .BYTE   0
                      02  00686          .BYTE   2
                      02  00687          .BYTE   2
                00000000  006P8          .LONG   0
                00000000  0068C          .LONG   0
                00000000' 00690          .ADDRESS OUTPUT_NAM
                00000000  00694          .LONG   0
                00000000  00698          .LONG   0
                      00  0069C          .BYTE   0
                      00  0069D          .BYTE   0
                    0000  0069E          .WORD   0
                00000000  006A0          .LONG   0
                    0000  006A4          .WORD   0
```

```
                00  006A6          .BYTE   0                                                       ;
                00  006A7          .BYTE   0                                                       ;
          00000000  006A8          .LONG   0                                                       ;
          00000000  006AC          .LONG   0                                                       ;
              0000  006B0          .WORD   0                                                       ;
                00  006B2          .BYTE   0                                                       ;
                00  006B3          .BYTE   0                                                       ;
          00000000  006B4          .LONG   0                                                       ;
                01  006B8 OUTPUT_RAB:                                                             ;
                                   .BYTE   1                                                       ;
                44  006B9          .BYTE   68                                                      ;
              0000  006BA          .WORD   0                                                       ;
          00000000  006BC          .LONG   0                                                       ;
          00000000  006C0          .LONG   0                                                       ;
          00000000  006C4          .LONG   0                                                       ;
             0000#  006C8          .WORD   0[3]                                                    ;
              0000  006CE          .WORD   0                                                       ;
          009110D2  006D0          .LONG   9507026                                                 ;
              0000  006D4          .WORD   0                                                       ;
                00  006D6          .BYTE   0                                                       ;
                00  006D7          .BYTE   0                                                       ;
              0000  006D8          .WORD   0                                                       ;
              0000  006DA          .WORD   0                                                       ;
          00000000  006DC          .LONG   0                                                       ;
          00000000  006E0          .LONG   0                                                       ;
          00000000  006E4          .LONG   0                                                       ;
          00000000  006E8          .LONG   0                                                       ;
                00  006EC          .BYTE   0                                                       ;
                00  006ED          .BYTE   0                                                       ;
                00  006EE          .BYTE   0                                                       ;
                00  006EF          .BYTE   0                                                       ;
          00000000  006F0          .LONG   0                                                       ;
          00000000' 006F4          .ADDRESS OUTPUT_FAB                                             ;
          00000000  006F8          .LONG   0                                                       ;

                                 MSG$_OPENIN==        9506970
                                 MSG$_READERR==       9506994
                                 MSG$_CLOSEIN==       9506898
                                 MSG$_OPENOUT==       9506978
                                 MSG$_WRITEERR==      9507026
                                 MSG$_CLOSEOUT==      9506906
                                 MSG$_CREATED==       9506931
                                 MSG$_EXISTS==        9507475
                                 MSG$_SYNTAX==        9507068
                                 MSG$_DIRNOTCRE==     9507522
                                 MSG$_BADVALUE==      9507090
                                          .EXTRN  LIB$TPARSE, LIB$GET_VM
                                          .EXTRN  LIB$FREE_VM, LIB$CREATE_DIR
                                          .EXTRN  LIB$CVT_DTB, CLI$GET_VALUE
                                          .EXTRN  CLI$PRESENT, SYS$SETDFPROT

                                          .PSECT  $CODE$,NOWRT,2

                     0004 00000 CREATE:   .WORD   Save R2                                         ; 0244
                6D       003C  CF   DE 00002          MOVAL   5$, (FP)                           ; 0271
          0000V CF             00   FB 00007          CALLS   #0, GET_CREATEQLS                  ; 0279
          0000V CF             00   FB 0000C 1$:      CALLS   #0, GET_OUTPUT_FILE                ; 0286
```

```
                        28          50  E9 00011          BLBC    R0, 4$                                  ;
                        07   0000'  CF  E9 00014          BLBC    QUALIFIER$FLAGS, 2$                     ; 0288
            0000V  CF               00  FB 00019          CALLS   #0, CREATE_DIR                          ; 0289
                                    05  11 0001E          BRB     3$                                      ;
            0000V  CF               00  FB 00020 2$:      CALLS   #0, CREATE_FILE                         ; 0290
                   52               50  D0 00025 3$:      MOVL    R0, STATUS                              ;
       DE   0000'  CF               05  E1 00028          BBC     #5, QUALIFIER$FLAGS, 1$                 ; 0292
                   DB               52  E9 0002E          BLBC    STATUS, 1$                              ; 0293
                             0000'  CF  9F 00031          PUSHAB  OUTPUT_FAB                              ; 0294
            0000V  CF               01  FB 00035          CALLS   #1, LOG_RESULTS                         ;
                                    D0  11 0003A          BRB     1$                                      ; 0286
                        50   0000'  CF  D0 0003C 4$:      MOVL    WORST_ERROR, R0                         ; 0297
                                        04 00041          RET                                             ; 0298
                                      0000 00042 5$:      .WORD   Save nothing                            ; 0271
                                    7E  D4 00044          CLRL    -(SP)                                   ;
                                    5E  DD 00046          PUSHL   SP                                      ;
                        7E      04  AC  7D 00048          MOVQ    4(AP), -(SP)                            ;
            0000V  CF               03  FB 0004C          CALLS   #3, HANDLER                             ;
                                        04 00051          RET                                             ;

; Routine Size:  82 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  307     0299  1 ROUTINE create_dir =
;  308     0300  1
;  309     0301  1 !----
;  310     0302  1 ! Functional Description
;  311     0303  1 !
;  312     0304  1 !       Parse the file name and invoke the directory creation subroutine.
;  313     0305  1 !
;  314     0306  1 ! Input Parameters
;  315     0307  1 !
;  316     0308  1 !       none
;  317     0309  1 !
;  318     0310  1 ! Output Parameters
;  319     0311  1 !
;  320     0312  1 !       Results of action returned (already signaled).
;  321     0313  1 !
;  322     0314  1 !
;  323     0315  1 !----
;  324     0316  1
;  325     0317  2 BEGIN
;  326     0318  2
;  327     0319  2 LOCAL
;  328     0320  2     p_owner_uic,                        ! Parameters to LIB$CREATE_DIR
;  329     0321  2     p_prot_enable,
;  330     0322  2     p_prot_value,
;  331     0323  2     p_max_versions,
;  332     0324  2     p_rvn,
;  333     0325  2     status,                             ! Catch all status return
;  334     0326  2     desc: VECTOR [2];                   ! Temporary work descriptor
;  335     0327  2
;  336     0328  2
;  337     0329  2 !
;  338     0330  2 ! Parse the file name.  This is done to remove and passwords from
;  339     0331  2 ! the file name in case it is used in an error or logging message.
;  340     0332  2 !
;  341     0333  2
;  342     0334  2 $PARSE (FAB=output_fab);        ! Do RMS PARSE
;  343     0335  2
;  344     0336  2
;  345     0337  2 !
;  346     0338  2 ! RMS parse's fail if the directory does not exists -- so instead
;  347     0339  2 ! of checking returned status look for an expanded name string.
;  348     0340  2 ! If name was so bad that no expansion resulted then inform the user.
;  349     0341  2 !
;  350     0342  2
;  351     0343  2 IF .output_nam [nam$b_esl] EQL 0        ! If no expanded name was obtained
;  352     0344  3 THEN BEGIN
;  353     0345  3         fab_error (output_fab);         !    then report the error
;  354     0346  3         return (false);                 !    and exit with status
;  355     0347  2      END;
;  356     0348  2
;  357     0349  2
;  358     0350  2 !
;  359     0351  2 ! The parse results in a trailing ".;" which must be stripped
;  360     0352  2 ! off before calling LIB$CREATE_DIR.  But if the user supplied a
;  361     0353  2 ! filename, extension, or version then pass it on -- it will trigger
;  362     0354  2 ! an error when the directory create is attempted.
;  363     0355  2 !
```

```
;  364     0356  2
;  365     0357  2 IF .output_nam [nam$b_name] NEQU 0       ! If user supplied a filename
;  366     0358  2 OR .output_nam [nam$b_type] GTRU 1       !    or a type
;  367     0359  2 OR .output_nam [nam$b_ver] GTRU 1        !    or a version
;  368     0360  2 THEN 0                                   ! then just continue
;  369     0361  2 ELSE                                     ! else strip off trailin g".;"
;  370     0362  2     output_nam [nam$b_esl] = .output_nam [nam$b_esl] - 2;
;  371     0363  2
;  372     0364  2
;  373     0365  2 !
;  374     0366  2 ! Create a temporary descriptor of the expanded name.
;  375     0367  2 !
;  376     0368  2
;  377     0369  2 desc[0] =.output_nam[nam$b_esl];         ! Create descriptor of parsed name
;  378     0370  2 desc[1] =.output_nam[nam$l_esa];         !  using length and address from $NAM
;  379     0371  2
;  380     0372  2 !
;  381     0373  2 ! Invoke library create directory routine passing user supplied parameters.
;  382     0374  2 !
;  383     0375  2
;  384     0376  2 p_owner_uic = p_prot_enable = p_prot_value = p_max_versions = p_rvn = 0;
;  385     0377  2
;  386     0378  2 IF .qualifier$flags[qual_owner_uic]
;  387     0379  2 THEN
;  388     0380  2     p_owner_uic = create$owner_uic;
;  389     0381  2
;  390     0382  2 IF .qualifier$flags[qual_protection]
;  391     0383  2 THEN
;  392     0384  3     BEGIN
;  393     0385  3     p_prot_enable = create$protection[1];
;  394     0386  3     p_prot_value = create$protection[0];
;  395     0387  2     END;
;  396     0388  2
;  397     0389  2 IF .qualifier$flags[qual_version_limit]
;  398     0390  2 THEN
;  399     0391  2     p_max_versions = create$version_limit;
;  400     0392  2
;  401     0393  2 IF .qualifier$flags[qual_volume]
;  402     0394  2 THEN
;  403     0395  2     p_rvn = create$volume;
;  404     0396  2
;  405     0397  2 status = lib$create_dir (                ! Call library create/dir routine
;  406     0398  2         desc,                            !  -directory name
;  407     0399  2         .p_owner_uic,                    !  -owner UIC
;  408     0400  2         .p_prot_enable, .p_prot_value,   !  -directory protection
;  409     0401  2         .p_max_versions,                 !  -default version limit
;  410     0402  2         .p_rvn);                         !  -volume
;  411     0403  2
;  412     0404  2 IF NOT .status                           ! Signal an error if necessary.
;  413     0405  2     THEN create_error (desc, .status);
;  414     0406  2
;  415     0407  2 !
;  416     0408  2 ! If directory already existed, print informational message and
;  417     0409  2 ! suppress logging.
;  418     0410  2 !
;  419     0411  2
;  420     0412  2 IF .status EQL SS$_NORMAL                ! If no directories created
```

```
;  421          0413  3 THEN BEGIN
;  422          0414  3      write_message(msg$_exists,1,desc); ! Inform the user
;  423          0415  3      RETURN false;                      ! Suppress logging
;  424          0416  2      END;
;  425          0417  2
;  426          0418  2 RETURN .status;
;  427          0419  1 END;


                                                                  .EXTRN  SYS$PARSE

                                          003C 00000 CREATE_DIR:
                                                                  .WORD   Save R2,R3,R4,R5                              ; 0299
                          55     0000'   CF   9E 00002            MOVAB   QUALIFIER$FLAGS, R5                           ;
                          5E             08   C2 00007            SUBL2   #8, SP                                        ;
                                 0668    C5   9F 0000A            PUSHAB  OUTPUT_FAB                                    ; 0334
              00000000G   00             01   FB 0000E            CALLS   #1, SYS$PARSE                                 ;
                                 059B    C5   95 00015            TSTB    OUTPUT_NAM+11                                 ; 0343
                                         0C   12 00019            BNEQ    1$                                            ;
                                 0668    C5   9F 0001B            PUSHAB  OUTPUT_FAB                                    ; 0345
                  0000V   CF             01   FB 0001F            CALLS   #1, FAB_ERROR                                 ;
                                       008E   31 00024            BRW     9$                                            ; 0346
                                 05CB    C5   95 00027 1$:        TSTB    OUTPUT_NAM+59                                 ; 0357
                                         13   12 0002B            BNEQ    2$                                            ;
                          01     05CC    C5   91 0002D            CMPB    OUTPUT_NAM+60, #1                             ; 0358
                                         0C   1A 00032            BGTRU   2$                                            ;
                          01     05CD    C5   91 00034            CMPB    OUTPUT_NAM+61, #1                             ; 0359
                                         05   1A 00039            BGTRU   2$                                            ;
                  059B    C5             02   82 0003B            SUBB2   #2, OUTPUT_NAM+11                             ; 0362
                          6E     059B    C5   9A 00040 2$:        MOVZBL  OUTPUT_NAM+11, DESC                           ; 0369
                  04      AE     059C    C5   D0 00045            MOVL    OUTPUT_NAM+12, DESC+4                         ; 0370
                                         50   7C 0004B            CLRQ    P_RVN                                         ; 0376
                                         52   7C 0004D            CLRQ    P_PROT_VALUE                                  ;
                                         54   D4 0004F            CLRL    P_OWNER_UIC                                   ;
           04             65             01   E1 00051            BBC     #1, QUALIFIER$FLAGS, 3$                       ; 0378
                          54       04    A5   9E 00055            MOVAB   CREATE$OWNER_UIC, P_OWNER_UIC                 ; 0380
           08             65             02   E1 00059 3$:        BBC     #2, QUALIFIER$FLAGS, 4$                       ; 0382
                          53       0A    A5   9E 0005D            MOVAB   CREATE$PROTECTION+2, P_PROT_ENABLE            ; 0385
                          52       08    A5   9E 00061            MOVAB   CREATE$PROTECTION, P_PROT_VALUE               ; 0386
           04             65             03   E1 00065 4$:        BBC     #3, QUALIFIER$FLAGS, 5$                       ; 0389
                          51       0C    A5   9E 00069            MOVAB   CREATE$VERSION_LIMIT, P_MAX_VERSIONS          ; 0391
           04             65             04   E1 0006D 5$:        BBC     #4, QUALIFIER$FLAGS, 6$                       ; 0393
                          50       10    A5   9E 00071            MOVAB   CREATE$VOLUME, P_RVN                          ; 0395
                                         50   DD 00075 6$:        PUSHL   P_RVN                                         ; 0402
                                         51   DD 00077            PUSHL   P_MAX_VERSIONS                                ; 0401
                                         52   DD 00079            PUSHL   P_PROT_VALUE                                  ; 0400
                                         53   DD 0007B            PUSHL   P_PROT_ENABLE                                 ;
                                         54   DD 0007D            PUSHL   P_OWNER_UIC                                   ; 0399
                                   14    AE   9F 0007F            PUSHAB  DESC                                          ; 0397
              00000000G   00             06   FB 00082            CALLS   #6, LIB$CREATE_DIR                            ;
                          52             50   D0 00089            MOVL    R0, STATUS                                    ;
                          0A             52   E8 0008C            BLBS    STATUS, 7$                                    ; 0404
                                         52   DD 0008F            PUSHL   STATUS                                        ; 0405
                                   04    AE   9F 00091            PUSHAB  DESC                                          ;
                  0000V   CF             02   FB 00094            CALLS   #2, CREATE_ERROR                              ;
                          01             52   D1 00099 7$:        CMPL    STATUS, #1                                    ; 0412
```

```
                                                13  12 0009C          BNEQ    8$                                      ;
                                                5E  DD 0009E          PUSHL   SP                                      ; 0414
                                                01  DD 000A0          PUSHL   #1                                      ;
                                    00911293    8F  DD 000A2          PUSHL   #9507475                                ;
                        00000000G   00          03  FB 000A8          CALLS   #3, LIB$SIGNAL                          ;
                                                04  11 000AF          BRB     9$                                      ; 0415
                                    50          52  D0 000B1 8$:      MOVL    STATUS, R0                              ; 0418
                                                04 000B4              RET                                             ;
                                                50  D4 000B5 9$:      CLRL    R0                                      ; 0419
                                                04 000B7              RET                                             ;

; Routine Size:  184 bytes,    Routine Base:  $CODE$ + 0052
```

```
;  429     0420  1 ROUTINE create_file =
;  430     0421  1
;  431     0422  1 !----
;  432     0423  1 ! Functional Description
;  433     0424  1 !
;  434     0425  1 !       Create the output file and invoke the copy subroutine.
;  435     0426  1 !
;  436     0427  1 ! Input Parameters
;  437     0428  1 !
;  438     0429  1 !       none
;  439     0430  1 !
;  440     0431  1 ! Output Parameters
;  441     0432  1 !
;  442     0433  1 !       Results of the operation are returned (already signaled).
;  443     0434  1 !
;  444     0435  1 !----
;  445     0436  1
;  446     0437  2 BEGIN
;  447     0438  2
;  448     0439  2 LOCAL
;  449     0440  2     default_prot:       WORD,           ! Process default file protection
;  450     0441  2     status;                             ! Catch all status return
;  451     0442  2
;  452     0443  2
;  453     0444  2 output_fab[fab$l_ctx] = msg$_openout;   ! Reinitialize error message
;  454     0445  2
;  455     0446  2 !
;  456     0447  2 ! Load the XAB with any user specified values for file ownership
;  457     0448  2 ! ([group,member]) or protection (System,Group,Owner,World: R W E D),
;  458     0449  2 ! or volume (relative volume number).
;  459     0450  2 !
;  460     0451  2
;  461     0452  2 output_xabpro[xab$l_uic] = .create$owner_uic;   ! Load owner UIC
;  462     0453  2
;  463     0454  2 output_xabpro[xab$w_pro] = -1;                  ! Default protection
;  464     0455  2
;  465     0456  2 output_xaball[xab$w_vol] = .create$volume;      ! Load volume
;  466     0457  2 output_fab[fab$l_xab]    = output_xaball;       ! Link FAB to XAB
;  467     0458  2
;  468     0459  2
;  469     0460  2 status = $CREATE (FAB=output_fab);              ! Create the output file.
;  470     0461  2
;  471     0462  2 !
;  472     0463  2 ! If the $CREATE was successful, call copy_in_out to copy SYS$INPUT
;  473     0464  2 ! to the user specified output file.
;  474     0465  2 !
;  475     0466  2
;  476     0467  2 IF .status                                      ! If create worked
;  477     0468  2     THEN
;  478     0469  3         BEGIN
;  479     0470  3         status = $DISPLAY (FAB=output_fab);
;  480     0471  3         IF NOT .status THEN fab_error (output_fab);
;  481     0472  3         if .qualifier$flags[qual_protection]
;  482     0473  3         THEN output_xabpro[xab$w_pro] =
;  483     0474  3                 .output_xabpro[xab$w_pro] AND NOT .create$protection[1] OR
;  484     0475  3                 .create$protection[0] AND .create$protection[1];
;  485     0476  3         status = copy_in_out ();                !  then call copy
```

```
;  486        0477  3          END
;  487        0478  2      ELSE fab_error(output_fab);                     !  else report an error.
;  488        0479  2
;  489        0480  2
;  490        0481  2 RETURN .status;
;  491        0482  1 END;


                                                            .EXTRN  SYS$CREATE, SYS$DISPLAY

                                    001C 00000 CREATE_FILE:
                                                            .WORD   Save R2,R3,R4                             ; 0420
              54     0000'   CF   9E 00002                  MOVAB   OUTPUT_FAB, R4                            ;
       18     A4  009110A2   8F   D0 00007                  MOVL    #9506978, OUTPUT_FAB+24                   ; 0444
       94     A4      F99C   C4   D0 0000F                  MOVL    CREATE$OWNER_UIC, OUTPUT_XABPRO+12        ; 0452
       90     A4             01   AE 00015                  MNEGW   #1, OUTPUT_XABPRO+8                       ; 0454
       EA     A4      F9A8   C4   B0 00019                  MOVW    CREATE$VOLUME, OUTPUT_XABALL+10           ; 0456
       24     A4        E0   A4   9E 0001F                  MOVAB   OUTPUT_XABALL, OUTPUT_FAB+36              ; 0457
                             54   DD 00024                  PUSHL   R4                                        ; 0460
     00000000G    00         01   FB 00026                  CALLS   #1, SYS$CREATE                            ;
                  52         50   D0 0002D                  MOVL    R0, STATUS                                ;
                  47         52   E9 00030                  BLBC    STATUS, 3$                                ; 0467
                             54   DD 00033                  PUSHL   R4                                        ; 0470
     00000000G    00         01   FB 00035                  CALLS   #1, SYS$DISPLAY                           ;
                  52         50   D0 0003C                  MOVL    R0, STATUS                                ;
                  07         52   E8 0003F                  BLBS    STATUS, 1$                                ; 0471
                             54   DD 00042                  PUSHL   R4                                        ;
          0000V   CF         01   FB 00044                  CALLS   #1, FAB_ERROR                             ;
  21       F998   C4         02   E1 00049 1$:              BBC     #2, QUALIFIER$FLAGS, 2$                   ; 0472
                  51     90  A4   3C 0004F                  MOVZWL  OUTPUT_XABPRO+8, R1                       ; 0474
                  50   F9A2  C4   3C 00053                  MOVZWL  CREATE$PROTECTION+2, R0                   ;
                  51         50   CA 00058                  BICL2   R0, R1                                    ;
                  50   F9A0  C4   3C 0005B                  MOVZWL  CREATE$PROTECTION, R0                     ; 0475
                  53   F9A2  C4   3C 00060                  MOVZWL  CREATE$PROTECTION+2, R3                   ;
                  53         53   D2 00065                  MCOML   R3, R3                                    ;
                  50         53   CA 00068                  BICL2   R3, R0                                    ;
 90     A4        50         51   A9 0006B                  BISW3   R1, R0, OUTPUT_XABPRO+8                   ;
          0000V   CF         00   FB 00070 2$:              CALLS   #0, COPY_IN_OUT                           ; 0476
                  52         50   D0 00075                  MOVL    R0, STATUS                                ;
                             07   11 00078                  BRB     4$                                        ; 0467
                             54   DD 0007A 3$:              PUSHL   R4                                        ; 0478
          0000V   CF         01   FB 0007C                  CALLS   #1, FAB_ERROR                             ;
                  50         52   D0 00081 4$:              MOVL    STATUS, R0                                ; 0481
                                  04 00084                  RET                                               ; 0482

; Routine Size:  133 bytes,    Routine Base:  $CODE$ + 010A


;  492        0483  1
```

```
;  494          0484  1 ROUTINE get_output_file =
;  495          0485  1
;  496          0486  1 !----
;  497          0487  1 ! Functional Description
;  498          0488  1 !
;  499          0489  1 !       Obtain the next file specification from the output file
;  500          0490  1 !       list.
;  501          0491  1 !
;  502          0492  1 ! Input Parameters
;  503          0493  1 !
;  504          0494  1 !       none
;  505          0495  1 !
;  506          0496  1 ! Output Parameters
;  507          0497  1 !
;  508          0498  1 !       TRUE returned if file name was produced
;  509          0499  1 !       FALSE returned if command line exhausted
;  510          0500  1 !
;  511          0501  1 !       output_fab = FAB initialized for current file name
;  512          0502  1 !       output_nam = NAM block associated with FAB
;  513          0503  1 !
;  514          0504  1 !----
;  515          0505  1
;  516          0506  2 BEGIN
;  517          0507  2
;  518          0508  2 LOCAL
;  519          0509  2     output_desc : $BBLOCK[dsc$c_s_bln];
;  520          0510  2
;  521          0511  2 CH$FILL(0,dsc$c_s_bln,output_desc);              ! Make descriptor dynamic
;  522          0512  2 output_desc[dsc$b_class] = dsc$k_class_d;
;  523          0513  2
;  524          0514  2 IF NOT cli$get_value($descriptor('OUTPUT'), output_desc)
;  525          0515  2     THEN RETURN false;
;  526          0516  2
;  527          0517  2
;  528          0518  2 !
;  529          0519  2 !       Initialize the FAB for initial parsing
;  530          0520  2 !
;  531          0521  2
;  532          0522  2 output_rlf[nam$b_rsl] = .output_nam[nam$b_esl];
;  533          0523  2 output_fab[fab$l_fna] = .output_desc[dsc$a_pointer];
;  534          0524  2 output_fab[fab$b_fns] = .output_desc[dsc$w_length];
;  535          0525  2
;  536          0526  2 RETURN true;
;  537          0527  1 END;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

                        54  55  50  54  55  4F  00009 P.AAC:  .ASCII  \OUTPUT\                          ;
                                                0000F         .BLKB   1
                                    00000006    00010 P.AAB:  .LONG   6                               ;
                                    00000000'   00014         .ADDRESS P.AAC                        ;


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

```
                                                003C 00000 GET_OUTPUT_FILE:
                                                                  .WORD    Save R2,R3,R4,R5                     ; 0484
                             5E            08  C2 00002           SUBL2    #8, SP                               ;
     08           0C         6E            00  2C 00005           MOVC5    #0, (SP), #0, #8, OUTPUT_DESC        ; 0511
                                           6E     0000A                                                         ;
                       03    AE            02  90 0000B           MOVB     #2, OUTPUT_DESC+3                    ; 0512
                                           5E  DD 0000F           PUSHL    SP                                   ; 0514
                                    0000'  CF  9F 00011           PUSHAB   P.AAB                                ;
                 00000000G   00            02  FB 00015           CALLS    #2, CLI$GET_VALUE                    ;
                             16            50  E9 0001C           BLBC     R0, 1$                               ;
                   0000'     CF     0000'  CF  90 0001F           MOVB     OUTPUT_NAM+11, OUTPUT_RLF+3          ; 0522
                   0000'     CF        04  AE  D0 00026           MOVL     OUTPUT_DESC+4, OUTPUT_FAB+44         ; 0523
                   0000'     CF            6E  90 0002C           MOVB     OUTPUT_DESC, OUTPUT_FAB+52           ; 0524
                             50            01  D0 00031           MOVL     #1, R0                               ; 0526
                                               04 00034           RET                                           ;
                                           50  D4 00035 1$:       CLRL     R0                                   ; 0527
                                               04 00037           RET                                           ;

; Routine Size:  56 bytes,    Routine Base:  $CODE$ + 018F
```

```
;  539     0528  1 ROUTINE copy_in_out: =
;  540     0529  1
;  541     0530  1 !----
;  542     0531  1 !
;  543     0532  1 ! Functional description
;  544     0533  1 !
;  545     0534  1 !       This routine is called from the main loop to copy from
;  546     0535  1 !       from SYS$INPUT to the created file
;  547     0536  1 !
;  548     0537  1 !
;  549     0538  1 ! Input parameters
;  550     0539  1 !
;  551     0540  1 !       None
;  552     0541  1 !
;  553     0542  1 ! Output parameters
;  554     0543  1 !
;  555     0544  1 !       First error encountered, or success is RETURNed
;  556     0545  1 !
;  557     0546  1 !----
;  558     0547  1
;  559     0548  2 BEGIN
;  560     0549  2
;  561     0550  2 LOCAL
;  562     0551  2     bytes_needed,                       ! Buffer size needed
;  563     0552  2     status;                             ! Status return value
;  564     0553  2
;  565     0554  3 IF NOT (status =
;  566     0555  4 $CONNECT (RAB=output_rab,ERR=rab_error) !Connect to output file
;  567     0556  2 ) THEN return (.status);
;  568     0557  2
;  569     0558  2 input_fab[fab$l_ctx] = msg$_openin;     ! Assign the error message
;  570     0559  2
;  571     0560  3 IF NOT (status =
;  572     0561  4 $OPEN (FAB=input_fab,ERR=fab_error)     ! Open SYS$INPUT
;  573     0562  2 ) THEN return (.status);
;  574     0563  2
;  575     0564  2 !
;  576     0565  2 ! Use the maximum record size (MRS) returned by the $OPEN to
;  577     0566  2 ! request a buffer from virtual memory and store the returned
;  578     0567  2 ! memory address directly in the input RAB.  If MRS = 0 (signifying
;  579     0568  2 ! an undefined record size) then use 512.
;  580     0569  2 !
;  581     0570  2
;  582     0571  2 bytes_needed =                          ! Define the byte count
;  583     0572  3         (IF NOT                         ! (IF valid MRS was returned
;  584     0573  4                 (.input_fab[fab$w_mrs] EQL 0)
;  585     0574  3         THEN .input_fab[fab$w_mrs]      !   then use it
;  586     0575  2         ELSE 512);                      ! Else use the default
```

```
;  588          0576  2
;  589          0577  2 !
;  590          0578  2 ! Now request 'bytes_needed' bytes from VMS.  Have VMS store the
;  591          0579  2 ! address of the granted memory directly in the 'user buffer'
;  592          0580  2 ! address field of the RAB
;  593          0581  2 !
;  594          0582  2
;  595          0583  2 IF NOT
;  596          0584  3         (status = lib$get_vm (          ! Request virtual memory of
;  597          0585  3                 bytes_needed,          !  -this many bytes
;  598          0586  3                 input_rab[rab$l_ubf]))  !  -beginning at this address
;  599          0587  2 THEN
;  600          0588  3                 BEGIN                  ! If allocation fails
;  601          0589  3                 vm_error(.status);     !  signal the user
;  602          0590  3                 return (.status);      ! Return if not fatal
;  603          0591  2                 END;
;  604          0592  2
;  605          0593  2 !
;  606          0594  2 ! Now define the size of the user buffer (usz) equal to
;  607          0595  2 ! bytes_needed
;  608          0596  2 !
;  609          0597  2
;  610          0598  2 input_rab[rab$w_usz] = .bytes_needed;   ! usz <-- bytes_needed
;  611          0599  2
;  612          0600  2 !
;  613          0601  2 ! Now that the necessary buffer space is available CONNECT to the
;  614          0602  2 ! input file
;  615          0603  2 !
;  616          0604  2
;  617          0605  3 IF NOT (status =
;  618          0606  4 $CONNECT (RAB=input_rab,ERR=rab_error)  ! Connect to SYS$INPUT
;  619          0607  2 ) THEN return (.status);
```

```
;  621          0608  2 !
;  622          0609  2 ! This is the main copy loop.  As long as $GET is successful (that
;  623          0610  2 ! is we read a record) issue $PUT's.  Use the buffer address and
;  624          0611  2 ! record size returned by the RMS GET call to fill in the buffer
;  625          0612  2 ! address and size fields of the RAB used for the $PUT.
;  626          0613  2 !
;  627          0614  2
;  628          0615  2 WHILE status =
;  629          0616  3         $GET (RAB=input_rab,ERR=rab_error)           ! Read a record
;  630          0617  3    DO BEGIN
;  631          0618  3       output_rab[rab$l_rbf] = .input_rab[rab$l_rbf]; ! Map input buffer
;  632          0619  3       output_rab[rab$w_rsz] = .input_rab[rab$w_rsz]; ! Map input size
;  633          0620  3       $PUT (RAB=output_rab,ERR=rab_error);           ! Write a record
;  634          0621  2       END;
;  635          0622  2
;  636          0623  3 IF NOT (.status EQL rms$_eof)           ! If any status other than
;  637          0624  2 THEN    return (.status);               !  expected eof, return it
;  638          0625  2
;  639          0626  2 !
;  640          0627  2 ! The file is now copied.  Assign the appropriate error messages
;  641          0628  2 ! (which are passed to the error routines via the 'user context'
;  642          0629  2 ! field (CTX) of the FAB) and $CLOSE both files.
;  643          0630  2 !
;  644          0631  2
;  645          0632  2 input_fab[fab$l_ctx] = msg$_closein;    ! Assign the error message
;  646          0633  2 output_fab[fab$l_ctx] = msg$_closeout;  ! Assign the error message
;  647          0634  2
;  648          0635  3 IF NOT (status =
;  649          0636  4 $CLOSE (FAB=input_fab,ERR=fab_error)    ! Close SYS$INPUT
;  650          0637  2 ) THEN return (.status);
;  651          0638  2
;  652          0639  2 !
;  653          0640  2 ! NOTE: If a file is created with different UIC via the /OWNER qualifier
;  654          0641  2 ! and it is on an ODS-1 structure, then you cannot change the file's
;  655          0642  2 ! attributes if an "owner" protection violation would occur.  In other
;  656          0643  2 ! words, as soon as you "$CREATE" it, you relinquish control to it.
;  657          0644  2 ! Therefore we disable XAB processing since it may call for illegal
;  658          0645  2 ! protection changes or some other illegal activity
;  659          0646  2 !
;  660          0647  2
;  661          0648  2 !output_fab[fab$l_xab] = 0;             ! Delink attribute blocks
;  662          0649  2
;  663          0650  3 IF NOT (status =
;  664          0651  4 $CLOSE (FAB=output_fab,ERR=fab_error)   ! Close output file
;  665          0652  2 ) THEN return (.status);
;  666          0653  2
;  667          0654  2 !
;  668          0655  2 ! Now release the virtual memory acquired above
;  669          0656  2 !
;  670          0657  2
;  671          0658  2 IF NOT
;  672          0659  3         (status = lib$free_vm (          ! Release virtual memory of
;  673          0660  3                 bytes_needed,            !  -this many bytes
;  674          0661  3                 input_rab[rab$l_ubf]))   !  -beginning at this address
;  675          0662  2 THEN
;  676          0663  3                 BEGIN                    ! If deallocation failed
;  677          0664  3                 vm_error(.status);       !  signal the user
```

```
;  678          0665  3                return (.status);       .  return if not fatal
;  679          0666  2                END;
;  680          0667  2
;  681          0668  2 return (true);
;  682          0669  2
;  683          0670  1 END;


                                                                .EXTRN  SYS$CONNECT, SYS$OPEN
                                                                .EXTRN  SYS$GET, SYS$PUT
                                                                .EXTRN  SYS$CLOSE

                                        00FC 00000 COPY_IN_OUT:
                                                                .WORD   Save R2,R3,R4,R5,R6,R7                  ; 0528
                                    57      0000V   CF  9E 00002        MOVAB   FAB_ERROR, R7                   ;
                                    56      0000V   CF  9E 00007        MOVAB   RAB_ERROR, R6                   ;
                                    55 00000000G    00  9E 0000C        MOVAB   SYS$CLOSE, R5                   ;
                                    54 00000000G    00  9E 00013        MOVAB   SYS$CONNECT, R4                 ;
                                    53      0000'   CF  9E 0001A        MOVAB   OUTPUT_RAB, R3                  ;
                                    5E              04  C2 0001F        SUBL2   #4, SP                          ;
                                            0048    8F  BB 00022        PUSHR   #^M<R3,R6>                      ; 0555
                                    64              02  FB 00026        CALLS   #2, SYS$CONNECT                 ;
                                    52              50  D0 00029        MOVL    R0, STATUS                      ;
                                    53              52  E9 0002C        BLBC    STATUS, 3$                      ;
                                FBFC    C3 0091109A 8F  D0 0002F        MOVL    #9506970, INPUT_FAB+24          ; 0558
                                                    5?  DD 00038        PUSHL   R7                              ; 0561
                                            FBE4    C3  9F 0003A        PUSHAB  INPUT_FAB                       ;
                                00000000G   00          02  FB 0003E  CALLS   #2, SYS$OPEN                    ;
                                            52          50  D0 00045    MOVL    R0, STATUS                      ;
                                            37          52  E9 00048    BLBC    STATUS, 3$                      ;
                                            50      FC1A    C3  3C 0004B    MOVZWL  INPUT_FAB+54, R0            ; 0573
                                                            05  12 00050    BNEQ    1$                          ;
                                            50      0200    8F  3C 00052    MOVZWL  #512, R0                    ; 0572
                                            6E              50  D0 00057 1$: MOVL    R0, BYTES_NEEDED            ;
                                                    FC58    C3  9F 0005A    PUSHAB  INPUT_RAB+36                ; 0586
                                                      04    AE  9F 0005E    PUSHAB  BYTES_NEEDED                ; 0584
                                00000000G   00          02  FB 00061    CALLS   #2, LIB$GET_VM                  ;
                                            52          50  D0 00068    MOVL    R0, STATUS                      ;
                                            03          52  E8 0006B    BLBS    STATUS, 2$                      ;
                                                    008B    31  0006E    BRW     6$                              ;
                                    FC54    C3          6E  B0 00071 2$: MOVW    BYTES_NEEDED, INPUT_RAB+32      ; 0598
                                                        56  DD 00076    PUSHL   R6                              ; 0606
                                                    FC34    C3  9F 00078    PUSHAB  INPUT_RAB                   ;
                                            64          02  FB 0007C    CALLS   #2, SYS$CONNECT                 ;
                                            52          50  D0 0007F    MOVL    R0, STATUS                      ;
                                            7E          52  E9 00082 3$: BLBC    STATUS, 7$                      ;
                                                        56  DD 00085 4$: PUSHL   R6                              ; 0616
                                                    FC34    C3  9F 00087    PUSHAB  INPUT_RAB                   ;
                                00000000G   00          02  FB 0008B    CALLS   #2, SYS$GET                     ;
                                            52          50  D0 00092    MOVL    R0, STATUS                      ;
                                            19          52  E9 00095    BLBC    STATUS, 5$                      ;
                                    28      A3      FC5C    C3  D0 00098    MOVL    INPUT_RAB+40, OUTPUT_RAB+40     ; 0618
                                    22      A3      FC56    C3  B0 0009E    MOVW    INPUT_RAB+34, OUTPUT_RAB+34     ; 0619
                                                    0048    8F  BB 000A4    PUSHR   #^M<R3,R6>                      ; 0620
                                00000000G   00          02  FB 000A8    CALLS   #2, SYS$PUT                     ;
                                                        D4  11 000AF    BRB     4$                              ; 0615
```

```
                0001827A  8F              52  D1 000B1 5$:    CMPL    STATUS, #98938                  ; 0623
                                          49  12 000B8        BNEQ    7$                              ;
                    FBFC  C3 00911052     8F  D0 000BA        MOVL    #9506898, INPUT_FAB+24          ; 0632
                      C8  A3 0091105A     8F  D0 000C3        MOVL    #9506906, OUTPUT_FAB+24         ; 0633
                                          57  DD 000CB        PUSHL   R7                              ; 0636
                                    FBE4  C3  9F 000CD        PUSHAB  INPUT_FAB                       ;
                                      65  02  FB 000D1        CALLS   #2, SYS$CLOSE                   ;
                                      52  50  D0 000D4        MOVL    R0, STATUS                      ;
                                      29  52  E9 000D7        BLBC    STATUS, 7$                      ;
                                          57  DD 000DA        PUSHL   R7                              ; 0651
                                      B0  A3  9F 000DC        PUSHAB  OUTPUT_FAB                      ;
                                      65  02  FB 000DF        CALLS   #2, SYS$CLOSE                   ;
                                      52  50  D0 000E2        MOVL    R0, STATUS                      ;
                                      1B  52  E9 000E5        BLBC    STATUS, 7$                      ;
                                    FC58  C3  9F 000E8        PUSHAB  INPUT_RAB+36                    ; 0661
                                      04  AE  9F 000EC        PUSHAB  BYTES_NEEDED                    ; 0659
                0000000G              00  02  FB 000EF        CALLS   #2, LIB$FREE_VM                 ;
                                      52  50  D0 000F6        MOVL    R0, STATUS                      ;
                                      0B  52  E8 000F9        BLBS    STATUS, 8$                      ;
                                          52  DD 000FC 6$:    PUSHL   STATUS                          ; 0664
                    0000V             CF  01  FB 000FE        CALLS   #1, VM_ERROR                    ;
                                      50  52  D0 00103 7$:    MOVL    STATUS, R0                      ; 0665
                                          04 00106            RET                                     ;
                                      50  01  D0 00107 8$:    MOVL    #1, R0                          ; 0668
                                          04 0010A            RET                                     ; 0670

; Routine Size:  267 bytes,    Routine Base:  $CODE$ + 01C7
```

```
0671  1 ROUTINE log_results (fab): =
0672  1
0673  1 !----
0674  1 !
0675  1 ! Functional description
0676  1 !
0677  1 !       This routine is called from the main loop whenever
0678  1 !       logging is requested
0679  1 !
0680  1 ! Input parameters
0681  1 !
0682  1 !       fab = Address of block describing the file
0683  1 !       fab$l_nam = pointer to name block
0684  1 !
0685  1 ! Output parameters
0686  1 !
0687  1 !       First error encountered, or TRUE is RETURNed
0688  1 !
0689  1 !----
0690  1
0691  2 BEGIN
0692  2
0693  2 MAP fab: REF $BBLOCK;                    ! Define fab block format
0694  2 BIND nam = .fab[fab$l_nam]: $BBLOCK;     ! Define name block
0695  2
0696  2 LOCAL desc: VECTOR[2];                   ! Temporary string descriptor
0697  2
0698  2 IF .nam[nam$b_rsl] NEQ 0                 ! If result string nonblank,
0699  3 THEN BEGIN
0700  3     desc[0] = .nam[nam$b_rsl];           ! then display it
0701  3     desc[1] = .nam[nam$l_rsa];
0702  3     END
0703  2 ELSE IF .nam[nam$b_esl] NEQ 0            ! Or if expanded name nonblank
0704  3 THEN BEGIN
0705  3     desc[0] = .nam[nam$b_esl];           ! then display it
0706  3     desc[1] = .nam[nam$l_esa];
0707  3     END
0708  3 ELSE BEGIN
0709  3     desc[0] = .fab[fab$b_fns];           ! Otherwise, use original
0710  3     desc[1] = .fab[fab$l_fna];           ! name string in FAB
0711  2     END;
0712  2
0713  2 write_message(msg$_created,1,DESC);      ! Output an informational msg
0714  2
0715  2 RETURN (true);
0716  2
0717  1 END;
```

```
                            0000 00000 LOG_RESULTS:
                                              .WORD   Save nothing                 ; 0671
5E          08   C2 00002                     SUBL2   #8, SP                       ;
51     04   AC   D0 00005                     MOVL    FAB, R1                      ; 0694
50     28   A1   D0 00009                     MOVL    40(R1), R0                   ;
```

```
                        03   A0  95 0000D       TSTB    3(R0)                          ; 0698
                             0B  13 00010       BEQL    1$                             ;
          6E            03   A0  9A 00012       MOVZBL  3(R0), DESC                    ; 0700
     04   AE            04   A0  D0 00016       MOVL    4(R0), DESC+4                  ; 0701
                             19  11 0001B       BRB     3$                             ; 0698
                        0B   A0  95 0001D 1$:   TSTB    11(R0)                         ; 0703
                             0B  13 00020       BEQL    2$                             ;
          6E            0B   A0  9A 00022       MOVZBL  11(R0), DESC                   ; 0705
     04   AE            0C   A0  D0 00026       MOVL    12(R0), DESC+4                 ; 0706
                             09  11 0002B       BRB     3$                             ; 0703
          6E            34   A1  9A 0002D 2$:   MOVZBL  52(R1), DESC                   ; 0709
     04   AE            2C   A1  D0 00031       MOVL    44(R1), DESC+4                 ; 0710
                             5E  DD 00036 3$:   PUSHL   SP                             ; 0713
                             01  DD 00038       PUSHL   #1                             ;
                  00911073   8F  DD 0003A       PUSHL   #9506931                       ;
000000000G  00               03  FB 00040       CALLS   #3, LIB$SIGNAL                 ;
            50               01  D0 00047       MOVL    #1, R0                         ; 0715
                                 04 0004A       RET                                    ; 0717

; Routine Size:  75 bytes,    Routine Base:  $CODE$ + 02D2
```

```
;  733     0718  1 ROUTINE get_createqls =
;  734     0719  1
;  735     0720  1 !----
;  736     0721  1 !
;  737     0722  1 ! Functional description
;  738     0723  1 !
;  739     0724  1 !       This routine calls CLI to obtain the command line and
;  740     0725  1 !       then all command qualifiers.
;  741     0726  1 !
;  742     0727  1 ! Input parameters
;  743     0728  1 !
;  744     0729  1 !       None
;  745     0730  1 !
;  746     0731  1 ! Output parameters
;  747     0732  1 !
;  748     0733  1 !       qualifier$flags = Bitmap marking which qualifiers are present
;  749     0734  1 !       First error encountered, or TRUE is RETURNed
;  750     0735  1 !
;  751     0736  1 !----
;  752     0737  1
;  753     0738  2 BEGIN
;  754     0739  2
;  755     0740  2 LOCAL
;  756     0741  2     prot_value : WORD INITIAL(0),
;  757     0742  2     prot_mask : WORD INITIAL(0),
;  758     0743  2     status,
;  759     0744  2     value_desc : $BBLOCK[dsc$c_s_bln];
;  760     0745  2
;  761     0746  2
;  762     0747  2 CH$FILL(0,dsc$c_s_bln,value_desc);              ! Make descriptor dynamic.
;  763     0748  2 value_desc[dsc$b_class] = dsc$k_class_d;
;  764     0749  2
;  765     0750  2 qualifier$flags[qual_directory]         = cli$present($descriptor('DIRECTORY'));
;  766     0751  2 qualifier$flags[qual_owner_uic]         = cli$present($descriptor('OWNER_UIC'));
;  767     0752  2 qualifier$flags[qual_protection]        = cli$present($descriptor('PROTECTION'));
;  768     0753  2 qualifier$flags[qual_version_limit]     = cli$present($descriptor('VERSION_LIMIT'));
;  769     0754  2 qualifier$flags[qual_volume]            = cli$present($descriptor('VOLUME'));
;  770     0755  2 qualifier$flags[qual_log]               = cli$present($descriptor('LOG'));
;  771     0756  2
;  772     0757  2 IF .qualifier$flags[qual_owner_uic]
;  773     0758  2 THEN
;  774     0759  3     BEGIN
;  775     0760  3     cli$get_value($descriptor('OWNER_UIC'),value_desc);
;  776     0761  3     IF NOT owner_uic_parse(value_desc)
;  777     0762  3         THEN RETURN false;
;  778     0763  2     END;
;  779     0764  2
;  780     0765  2
;  781     0766  2 IF .qualifier$flags[qual_protection]
;  782     0767  2 THEN
;  783     0768  3     BEGIN
;  784     0769  3     IF cli$present($descriptor('PROTECTION.SYSTEM'))
;  785     0770  3     THEN
;  786     0771  4         BEGIN
;  787     0772  4         prot_mask = .prot_mask OR %X'000F';
;  788     0773  4         IF cli$get_value($descriptor('PROTECTION.SYSTEM'),value_desc)
;  789     0774  4         THEN prot_value = parse_class(value_desc);
```

```
;  790          0775  3          END;
;  791          0776  3      IF cli$present($descriptor('PROTECTION.OWNER'))
;  792          0777  3      THEN
;  793          0778  4          BEGIN
;  794          0779  4          prot_mask = .prot_mask OR %X'00F0';
;  795          0780  4          IF cli$get_value($descriptor('PROTECTION.OWNER'),value_desc)
;  796          0781  4          THEN prot_value = .prot_value OR parse_class(value_desc)^4;
;  797          0782  3          END;
;  798          0783  3      IF cli$present($descriptor('PROTECTION.GROUP'))
;  799          0784  3      THEN
;  800          0785  4          BEGIN
;  801          0786  4          prot_mask = .prot_mask OR %X'0F00';
;  802          0787  4          IF cli$get_value($descriptor('PROTECTION.GROUP'),value_desc)
;  803          0788  4          THEN prot_value = .prot_value OR parse_class(value_desc)^8;
;  804          0789  3          END;
;  805          0790  3      IF cli$present($descriptor('PROTECTION.WORLD'))
;  806          0791  3      THEN
;  807          0792  4          BEGIN
;  808          0793  4          prot_mask = .prot_mask OR %X'F000';
;  809          0794  4          IF cli$get_value($descriptor('PROTECTION.WORLD'),value_desc)
;  810          0795  4          THEN prot_value = .prot_value OR parse_class(value_desc)^12;
;  811          0796  3          END;
;  812          0797  3
;  813          0798  3 !
;  814          0799  3 ! Complement the protection value since at this point, a bit set true
;  815          0800  3 ! indicates that we want to ALLOW access, while the system convention
;  816          0801  3 ! is that a bit set true indicates that we want to DENY access.
;  817          0802  3 !
;  818          0803  3
;  819          0804  3      create$protection[0] = NOT .prot_value;
;  820          0805  3      create$protection[1] = .prot_mask;
;  821          0806  2      END;
;  822          0807  2
;  823          0808  2
;  824          0809  2 IF .qualifier$flags[qual_version_limit]
;  825          0810  2 THEN
;  826          0811  3      BEGIN
;  827          0812  3      cli$get_value($descriptor('VERSION_LIMIT'),value_desc);
;  828          0813  3      status = lib$cvt_dtb(.value_desc[dsc$w_length],
;  829          0814  3                           .value_desc[dsc$a_pointer],
;  830          0815  3                           create$version_limit);
;  831          0816  3      IF NOT .status
;  832          0817  3      THEN
;  833          0818  4          BEGIN
;  834          0819  4          write_message(msg$_badvalue,1,value_desc);
;  835          0820  4          RETURN false;
;  836          0821  3          END;
;  837          0822  2      END;
;  838          0823  2
;  839          0824  2
;  840          0825  2 IF .qualifier$flags[qual_volume]
;  841          0826  2 THEN
;  842          0827  3      BEGIN
;  843          0828  3      cli$get_value($descriptor('VOLUME'),value_desc);
;  844          0829  3      status = lib$cvt_dtb(.value_desc[dsc$w_length],
;  845          0830  3                           .value_desc[dsc$a_pointer],
;  846          0831  3                           create$volume);
```

```
;  847          0832  3     IF NOT .status
;  848          0833  3     THEN
;  849          0834  4         BEGIN
;  850          0835  4         write_message(msg$_badvalue,1,value_desc);
;  851          0836  4         RETURN false;
;  852          0837  3         END;
;  853          0838  2     END;
;  854          0839  2
;  855          0840  2 RETURN true;
;  856          0841  1 END;


                                                                .PSECT  $PLIT$,NOWRT,NOEXE,2

                    59  52  4F  54  43  45  52  49  44  00018 P.AAE:  .ASCII  \DIRECTORY\                ;
                                                        00021        .BLKB   3                          ;
                                                    00000009 00024 P.AAD:  .LONG   9                    ;
                                                    00000000' 00028        .ADDRESS P.AAE               ;
                    43  49  55  5F  52  45  4E  57  4F  0002C P.AAG:  .ASCII  \OWNER_UIC\               ;
                                                        00035        .BLKB   3                          ;
                                                    00000009 00038 P.AAF:  .LONG   9                    ;
                                                    00000000' 0003C        .ADDRESS P.AAG               ;
                4E  4F  49  54  43  45  54  4F  52  50  00040 P.AAI:  .ASCII  \PROTECTION\              ;
                                                        0004A        .BLKB   2                          ;
                                                    0000000A 0004C P.AAH:  .LONG   10                   ;
                                                    00000000' 00050        .ADDRESS P.AAI               ;
        54  49  4D  49  4C  5F  4E  4F  49  53  52  45  56  00054 P.AAK:  .ASCII  \VERSION_LIMIT\         ;
                                                        00061        .BLKB   3                          ;
                                                    0000000D 00064 P.AAJ:  .LONG   13                   ;
                                                    00000000' 00068        .ADDRESS P.AAK               ;
                                        45  4D  55  4C  4F  56  0006C P.AAM:  .ASCII  \VOLUME\           ;
                                                        00072        .BLKB   2                          ;
                                                    00000006 00074 P.AAL:  .LONG   6                    ;
                                                    00000000' 00078        .ADDRESS P.AAM               ;
                                                47  4F  4C  0007C P.AAO:  .ASCII  \LOG\                 ;
                                                        0007F        .BLKB   1                          ;
                                                    00000003 00080 P.AAN:  .LONG   3                    ;
                                                    00000000' 00084        .ADDRESS P.AAO               ;
                        43  49  55  5F  52  45  4E  57  4F  00088 P.AAQ:  .ASCII  \OWNER_UIC\           ;
                                                        00091        .BLKB   3                          ;
                                                    00000009 00094 P.AAP:  .LONG   9                    ;
                                                    00000000' 00098        .ADDRESS P.AAQ               ;
54  53  59  53  2E  4E  4F  49  54  43  45  54  4F  52  50  0009C P.AAS:  .ASCII  \PROTECTION.SYSTEM\     ;
                                                    4D  45  000AB                                       ;
                                                        000AD        .BLKB   3                          ;
                                                    00000011 000B0 P.AAR:  .LONG   17                   ;
                                                    00000000' 000B4        .ADDRESS P.AAS               ;
54  53  59  53  2E  4E  4F  49  54  43  45  54  4F  52  50  000B8 P.AAU:  .ASCII  \PROTECTION.SYSTEM\     ;
                                                    4D  45  000C7                                       ;
                                                        000C9        .BLKB   3                          ;
                                                    00000011 000CC P.AAT:  .LONG   17                   ;
                                                    00000000' 000D0        .ADDRESS P.AAU               ;
45  4E  57  4F  2E  4E  4F  49  54  43  45  54  4F  52  50  000D4 P.AAW:  .ASCII  \PROTECTION.OWNER\     ;
                                                        52  000E3                                       ;
                                                    00000010 000E4 P.AAV:  .LONG   16                   ;
                                                    00000000' 000E8        .ADDRESS P.AAW               ;
```

```
45 4E 57 4F 2E 4E 4F 49 54 43 45 54 4F 52 50  000EC P.AAY:  .ASCII  \PROTECTION.OWNER\                                  ;
                                          52  000FB                                                                      ;
                                    00000010  000FC P.AAX:  .LONG   16                                                   ;
                                   00000000'  00100         .ADDRESS P.AAY                                               ;
55 4F 52 47 2E 4E 4F 49 54 43 45 54 4F 52 50  00104 P.ABA:  .ASCII  \PROTECTION.GROUP\                                  ;
                                          50  00113                                                                      ;
                                    00000010  00114 P.AAZ:  .LONG   16                                                   ;
                                   00000000'  00118         .ADDRESS P.ABA                                               ;
55 4F 52 47 2E 4E 4F 49 54 43 45 54 4F 52 50  0011C P.ABC:  .ASCII  \PROTECTION.GROUP\                                  ;
                                          50  0012B                                                                      ;
                                    00000010  0012C P.ABB:  .LONG   16                                                   ;
                                   00000000'  00130         .ADDRESS P.ABC                                               ;
4C 52 4F 57 2E 4E 4F 49 54 43 45 54 4F 52 50  00134 P.ABE:  .ASCII  \PROTECTION.WORLD\                                  ;
                                          44  00143                                                                      ;
                                    00000010  00144 P.ABD:  .LONG   16                                                   ;
                                   00000000'  00148         .ADDRESS P.ABE                                               ;
4C 52 4F 57 2E 4E 4F 49 54 43 45 54 4F 52 50  0014C P.ABG:  .ASCII  \PROTECTION.WORLD\                                  ;
                                          44  0015B                                                                      ;
                                    00000010  0015C P.ABF:  .LONG   16                                                   ;
                                   00000000'  00160         .ADDRESS P.ABG                                               ;
      54 49 4D 49 4C 5F 4E 4F 49 53 52 45 56  00164 P.ABI:  .ASCII  \VERSION_LIMIT\                                     ;
                                              00171         .BLKB   3
                                    0000000D  00174 P.ABH:  .LONG   13                                                   ;
                                   00000000'  00178         .ADDRESS P.ABI                                               ;
                           45 4D 55 4C 4F 56  0017C P.ABK:  .ASCII  \VOLUME\                                            ;
                                              00182         .BLKB   2
                                    00000006  00184 P.ABJ:  .LONG   6                                                    ;
                                   00000000'  00188         .ADDRESS P.ABK                                               ;


                                                            .PSECT  $CODE$,NOWRT,2

                                        0FFC  00000 GET_CREATEQLS:
                                                            .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11                ; 0718
                    5B 00000000G   00    9E  00002          MOVAB   CLI$GET_VALUE, R11                                   ;
                    5A     0000'   CF    9E  00009          MOVAB   QUALIFIER$FLAGS, R10                                 ;
                    59     0000'   CF    9E  0000E          MOVAB   P.AAD, R9                                            ;
                    58 00000000G   00    9E  00013          MOVAB   CLI$PRESENT, R8                                      ;
                    5E             08    C2  0001A          SUBL2   #8, SP                                               ;
                                   57    B4  0001D          CLRW    PROT_VALUE                                           ; 0738
                                   56    B4  0001F          CLRW    PROT_MASK                                            ;
   08          00          6E      00    2C  00021          MOVC5   #0, (SP), #0, #8, VALUE_DESC                         ; 0747
                                   6E        00026
                   03      AE      02    90  00027          MOVB    #2, VALUE_DESC+3                                     ; 0748
                                   59    DD  0002B          PUSHL   R9                                                   ; 0750
                           68      01    FB  0002D          CALLS   #1, CLI$PRESENT                                      ;
   6A          01          00      50    F0  00030          INSV    R0, #0, #1, QUALIFIER$FLAGS                          ;
                               14  A9    9F  00035          PUSHAB  P.AAF                                                ; 0751
                           68      01    FB  00038          CALLS   #1, CLI$PRESENT                                      ;
   6A          01          01      50    F0  0003B          INSV    R0, #1, #1, QUALIFIER$FLAGS                          ;
                               28  A9    9F  00040          PUSHAB  P.AAH                                                ; 0752
                           68      01    FB  00043          CALLS   #1, CLI$PRESENT                                      ;
   6A          01          02      50    F0  00046          INSV    R0, #2, #1, QUALIFIER$FLAGS                          ;
                               40  A9    9F  0004B          PUSHAB  P.AAJ                                                ; 0753
                           68      01    FB  0004E          CALLS   #1, CLI$PRESENT                                      ;
   6A          01          03      50    F0  00051          INSV    R0, #3, #1, QUALIFIER$FLAGS                          ;
```

```
                            50  A9  9F 00056         PUSHAB  P.AAL                                   ; 0754
                       6B       01  FB 00059         CALLS   #1, CLI$PRESENT                         ;
     6A          01    04       50  F0 0005C         INSV    R0, #4, #1, QUALIFIER$FLAGS             ;
                            5C  A9  9F 00061         PUSHAB  P.AAN                                   ; 0755
                       6B       01  FB 00064         CALLS   #1, CLI$PRESENT                         ;
     6A          01    05       50  F0 00067         INSV    R0, #5, #1, QUALIFIER$FLAGS             ;
                 15    6A       01  E1 0006C         BBC     #1, QUALIFIER$FLAGS, 1$                 ; 0757
                                5E  DD 00070         PUSHL   SP                                      ; 0760
                            70  A9  9F 00072         PUSHAB  P.AAP                                   ;
                       6B       02  FB 00075         CALLS   #2, CLI$GET_VALUE                       ;
                                5E  DD 00078         PUSHL   SP                                      ; 0761
                 0000V CF       01  FB 0007A         CALLS   #1, OWNER_UIC_PARSE                     ;
                       03       50  E8 0007F         BLBS    R0, 1$                                  ;
                          010A  31  00082            BRW     11$                                     ;
                 03    6A       02  E0 00085 1$:     BBS     #2, QUALIFIER$FLAGS, 2$                 ; 0766
                          00A4  31  00089            BRW     7$                                      ;
                          008C  C9  9F 0008C 2$:     PUSHAB  P.AAR                                   ; 0769
                       6B       01  FB 00090         CALLS   #1, CLI$PRESENT                         ;
                       19       50  E9 00093         BLBC    R0, 3$                                  ;
                       56       0F  88 00096         BISB2   #15, PROT_MASK                          ; 0772
                                5E  DD 00099         PUSHL   SP                                      ; 0773
                          00A8  C9  9F 0009B         PUSHAB  P.AAT                                   ;
                       6B       02  FB 0009F         CALLS   #2, CLI$GET_VALUE                       ;
                       0A       50  E9 000A2         BLBC    R0, 3$                                  ;
                                5E  DD 000A5         PUSHL   SP                                      ; 0774
                 0000V CF       01  FB 000A7         CALLS   #1, PARSE_CLASS                         ;
                       57       50  B0 000AC         MOVW    R0, PROT_VALUE                          ;
                          00C0  C9  9F 000AF 3$:     PUSHAB  P.AAV                                   ; 0776
                       6B       01  FB 000B3         CALLS   #1, CLI$PRESENT                         ;
                       1D       50  E9 000B6         BLBC    R0, 4$                                  ;
                       56   F0  8F  88 000B9         BISB2   #240, PROT_MASK                         ; 0779
                                5E  DD 000BD         PUSHL   SP                                      ; 0780
                          00D8  C9  9F 000BF         PUSHAB  P.AAX                                   ;
                       6B       02  FB 000C3         CALLS   #2, CLI$GET_VALUE                       ;
                       0D       50  E9 000C6         BLBC    R0, 4$                                  ;
                                5E  DD 000C9         PUSHL   SP                                      ; 0781
                 0000V CF       01  FB 000CB         CALLS   #1, PARSE_CLASS                         ;
                       50       10  C4 000D0         MULL2   #16, R0                                 ;
                       57       50  A8 000D3         BISW2   R0, PROT_VALUE                          ;
                          00F0  C9  9F 000D6 4$:     PUSHAB  P.AAZ                                   ; 0783
                       6B       01  FB 000DA         CALLS   #1, CLI$PRESENT                         ;
                       1F       50  E9 000DD         BLBC    R0, 5$                                  ;
                       56 0F00  8F  A8 000E0         BISW2   #3840, PROT_MASK                        ; 0786
                                5E  DD 000E5         PUSHL   SP                                      ; 0787
                          0108  C9  9F 000E7         PUSHAB  P.ABB                                   ;
                       6B       02  FB 000EB         CALLS   #2, CLI$GET_VALUE                       ;
                       0E       50  E9 000EE         BLBC    R0, 5$                                  ;
                                5E  DD 000F1         PUSHL   SP                                      ; 0788
                 0000V CF       01  FB 000F3         CALLS   #1, PARSE_CLASS                         ;
                 50    50       08  78 000F8         ASHL    #8, R0, R0                              ;
                       57       50  A8 000FC         BISW2   R0, PROT_VALUE                          ;
                          0120  C9  9F 000FF 5$:     PUSHAB  P.ABD                                   ; 0790
                       6B       01  FB 00103         CALLS   #1, CLI$PRESENT                         ;
                       1F       50  E9 00106         BLBC    R0, 6$                                  ;
                       56 F000  8F  A8 00109         BISW2   #61440, PROT_MASK                       ; 0793
                                5E  DD 0010E         PUSHL   SP                                      ; 0794
                          0138  C9  9F 00110         PUSHAB  P.ABF                                   ;
```

```
                6B              02  FB 00114          CALLS   #2, CLI$GET_VALUE                                    ;
                0E              50  E9 00117          BLBC    R0, 6$                                               ;
                                5E  DD 0011A          PUSHL   SP                                                   ; 0795
          0000V CF              01  FB 0011C          CALLS   #1, PARSE_CLASS
     50         50              0C  78 00121          ASHL    #12, R0, R0
                57              50  A8 00125          BISW2   R0, PROT_VALUE
          08    AA              57  B2 00128 6$:      MCOMW   PROT_VALUE, CREATE$PROTECTION                        ; 0804
          0A    AA              56  B0 0012C          MOVW    PROT_MASK, CREATE$PROTECTION+2                       ; 0805
     20         6A              03  E1 00130 7$:      BBC     #3, QUALIFIER$FLAGS, 8$                              ; 0809
                                5E  DD 00134          PUSHL   SP                                                   ; 0812
                      0150      C9  9F 00136          PUSHAB  P.ABH
                6B              02  FB 0013A          CALLS   #2, CLI$GET_VALUE
                      0C        AA  9F 0013D          PUSHAB  CREATE$VERSION_LIMIT                                 ; 0813
                      08        AE  DD 00140          PUSHL   VALUE_DESC+4                                         ; 0814
                7E    08        AE  3C 00143          MOVZWL  VALUE_DESC, -(SP)                                    ; 0813
     00000000G  00              03  FB 00147          CALLS   #3, LIB$CVT_DTB
                52              50  D0 0014E          MOVL    R0, STATUS
                24              52  E9 00151          BLBC    STATUS, 9$                                           ; 0816
     33         6A              04  E1 00154 8$:      BBC     #4, QUALIFIER$FLAGS, 10$                             ; 0825
                                5E  DD 00158          PUSHL   SP                                                   ; 0828
                      0160      C9  9F 0015A          PUSHAB  P.ABJ
                6B              02  FB 0015E          CALLS   #2, CLI$GET_VALUE
                      10        AA  9F 00161          PUSHAB  CREATE$VOLUME                                        ; 0829
                      08        AE  DD 00164          PUSHL   VALUE_DESC+4                                         ; 0830
                7E    08        AE  3C 00167          MOVZWL  VALUE_DESC, -(SP)                                    ; 0829
     00000000G  00              03  FB 0016B          CALLS   #3, LIB$CVT_DTB
                52              50  D0 00172          MOVL    R0, STATUS
                13              52  E8 00175          BLBS    STATUS, 10$                                          ; 0832
                                5E  DD 00178 9$:      PUSHL   SP                                                   ; 0835
                                01  DD 0017A          PUSHL   #1
                00911112        8F  DD 0017C          PUSHL   #9507090
     00000000G  00              03  FB 00182          CALLS   #3, LIB$SIGNAL
                                04  11 00189          BRB     11$                                                  ; 0836
                50              01  D0 0018B 10$:     MOVL    #1, R0                                               ; 0840
                                04  0018E             RET
                                50  D4 0018F 11$:     CLRL    R0                                                   ; 0841
                                04  00191             RET                                                          ;

; Routine Size:  402 bytes,    Routine Base:  $CODE$ + 031D
```

```
;  858          0842  1 ROUTINE owner_uic_parse (qual_desc_block): =
;  859          0843  1
;  860          0844  1 !----
;  861          0845  1 !
;  862          0846  1 ! Functional description
;  863          0847  1 !
;  864          0848  1 !       This routine is called from CLI to parse the value
;  865          0849  1 !       associated with the /OWNER_UIC qualifier.
;  866          0850  1 !
;  867          0851  1 ! Input parameters
;  868          0852  1 !
;  869          0853  1 !       qual_desc_block = Address of CLI request block
;  870          0854  1 !
;  871          0855  1 ! Output parameters
;  872          0856  1 !
;  873          0857  1 !       create$_owner_uic = Value if /OWNER_UIC specified
;  874          0858  1 !       First error encountered, or TRUE is RETURNed
;  875          0859  1 !
;  876          0860  1 !----
;  877          0861  1
;  878          0862  2 BEGIN
;  879          0863  2
;  880          0864  2 LOCAL
;  881          0865  2     status;
;  882          0866  2
;  883          0867  2 MAP qual_desc_block: REF $BBLOCK;       ! Define block format
;  884          0868  2
;  885          0869  2 BIND
;  886          0870  2     uicvec = create$owner_uic : VECTOR [2,WORD];
;  887          0871  2
;  888          0872  2 !
;  889          0873  2 ! Move the descriptor returned by CLI into the TPARSE parameter block
;  890          0874  2 !
;  891          0875  2
;  892          0876  2 tparse_block[tpa$l_stringcnt] = .qual_desc_block[dsc$w_length];
;  893          0877  2 tparse_block[tpa$l_stringptr] = .qual_desc_block[dsc$a_pointer];
;  894          0878  2
;  895          0879  2 !
;  896          0880  2 ! Now call TPARSE using the appropriate STATE and KEYWORD tables.
;  897          0881  2 ! If successful TPARSE will load storage as follows:
;  898          0882  2 !
;  899          0883  2 !       owner_uic <-- converted UIC
;  900          0884  2 !
;  901          0885  2
;  902          0886  3 IF NOT (status = lib$tparse     (               ! Call TPARSE with
;  903          0887  3                         tparse_block,          !  -parameter block
;  904          0888  3                         owner_uic_stb,         !  -state table
;  905          0889  3                         owner_uic_ktb))        !  -keyword table
;  906          0890  3
;  907          0891  3     THEN BEGIN                                 ! If TPARSE fails
;  908        P 0892  3         write_message (                       ! Inform the user
;  909        P 0893  3                         msg$_syntax,         !  -of SYNTAX error
;  910        P 0894  3                         1,                   !  -one FAO arguement
;  911        P 0895  3                         .qual_desc_block,    !  -UIC string
;  912          0896  3                         .status);            !  -plus original error
;  913          0897  3         return (.status);                    ! Only if non-fatal
;  914          0898  2         END;
```

```
;  915          0899  2
;  916          0900  2
;  917          0901  2 return (true);
;  918          0902  2
;  919          0903  1 END;


                                                          UICVEC=          CREATE$OWNER_UIC


                                            000C 00000 OWNER_UIC_PARSE:
                                                                 .WORD    Save R2,R3                            ; 0842
                         52          04   AC   D0 00002          MOVL     QUAL_DESC_BLOCK, R2                   ; 0876
               0000'     CF               62   3C 00006          MOVZWL   (R2), TPARSE_BLOCK+8                  ;
               0000'     CF          04   A2   D0 0000B          MOVL     4(R2), TPARSE_BLOCK+12                ; 0877
                                  0000'   CF   9F 00011          PUSHAB   OWNER_UIC_KTB                         ; 0886
                                  0000'   CF   9F 00015          PUSHAB   OWNER_UIC_STB                         ;
                                  0000'   CF   9F 00019          PUSHAB   TPARSE_BLOCK                          ;
           00000000G     00               03   FB 0001D          CALLS    #3, LIB$TPARSE                        ;
                         53               50   D0 00024          MOVL     R0, STATUS                            ;
                         15               53   E8 00027          BLBS     STATUS, 1$                            ;
                                          0C   BB 0002A          PUSHR    #^M<R2,R3>                            ; 0896
                                          01   DD 0002C          PUSHL    #1                                    ;
                              009110FC    8F   DD 0002E          PUSHL    #9507068                              ;
           00000000G     00               04   FB 00034          CALLS    #4, LIB$SIGNAL                        ;
                         50               53   D0 0003B          MOVL     STATUS, R0                            ; 0897
                                               04 0003E          RET                                            ;
                         50               01   D0 0003F 1$:      MOVL     #1, R0                                ; 0901
                                               04 00042          RET                                            ; 0903

; Routine Size:  67 bytes,    Routine Base:  $CODE$ + 04AF
```

```
1 ROUTINE parse_class (desc) =
1
1 !---
1 !
1 ! This routine parses one class of user (e.g. SYSTEM, OWNER, GROUP, WORLD)
1 ! to see what protection is allowed.  The value returned in the low 4 bits
1 ! is the protection code, with the bits set to reflect that access is
1 ! requested.  Note that this is exactly the opposite of what the system wants.
1 !
1 ! Inputs:
1 !
1 !       DESC -- a descriptor pointing to the ASCII representation of the
1 !               protection desired
1 !
1 !---
1
2 BEGIN
2
2 MAP desc : REF $BBLOCK;
2
2 LOCAL
2     pointer,                          ! Pointer to string
2     result : INITIAL(0);              ! Resultant protection (default is no accesss)
2
2 !
2 ! Scan for the occurrence of each keyletter, and, if it is there, set the
2 ! appropriate bit.
2 !
2 pointer = .desc[dsc$a_pointer];
2 INCR index FROM 1 to .desc[dsc$w_length] DO
3     BEGIN
3     LOCAL char : BYTE;
3     char = CH$RCHAR_A(pointer);
3     IF .char EQL'R'
3     THEN result = .result OR %X'1'
3     ELSE IF .char EQL  'W'
3     THEN result = .result OR %X'2'
3     ELSE IF .char EQL 'E'
3          OR .char EQL 'P'
3     THEN result = .result OR %X'4'
3     ELSE IF .char EQL 'D'
3          OR .char EQL 'L'
3     THEN result = .result OR %X'8'
3     ELSE SIGNAL_STOP (msg$_syntax, 1, .desc);
2     END;
2
2 RETURN .result;
1 END;
```

```
                        007C 00000 PARSE_CLASS:
                                   .WORD   Save R2,R3,R4,R5,R6          ; 0904
      52      04    AC  D0 00002   MOVL    DESC, R2                     ; 0932
      56      04    A2  D0 00006   MOVL    4(R2), POINTER               ;
```

```
                    55              62  3C 0000A          MOVZWL  (R2), R5                        ; 0933
                                    53  7C 0000D          CLRQ    INDEX
                                    4C  11 0000F          BRB     8$
                    50              86  90 00011 1$:      MOVB    (POINTER)+, CHAR                ; 0936
                52  8F              50  91 00014          CMPB    CHAR, #82                       ; 0937
                                    05  12 00018          BNEQ    2$
                    54              01  88 0001A          BISB2   #1, RESULT                      ; 0938
                                    3E  11 0001D          BRB     8$
                57  8F              50  91 0001F 2$:      CMPB    CHAR, #87                       ; 0939
                                    05  12 00023          BNEQ    3$
                    54              02  88 00025          BISB2   #2, RESULT                      ; 0940
                                    33  11 00028          BRB     8$
                45  8F              50  91 0002A 3$:      CMPB    CHAR, #69                       ; 0941
                                    06  13 0002E          BEQL    4$
                50  8F              50  91 00030          CMPB    CHAR, #80                       ; 0942
                                    05  12 00034          BNEQ    5$
                    54              04  88 00036 4$:      BISB2   #4, RESULT                      ; 0943
                                    22  11 00039          BRB     8$
                44  8F              50  91 0003B 5$:      CMPB    CHAR, #68                       ; 0944
                                    06  13 0003F          BEQL    6$
                4C  8F              50  91 00041          CMPB    CHAR, #76                       ; 0945
                                    05  12 00045          BNEQ    7$
                    54              08  88 00047 6$:      BISB2   #8, RESULT                      ; 0946
                                    11  11 0004A          BRB     8$
                                    52  DD 0004C 7$:      PUSHL   R2                              ; 0947
                                    01  DD 0004E          PUSHL   #1
                        009110FC    8F  DD 00050          PUSHL   #9507068
            00000000G               00  03 FB 00056      CALLS   #3, LIB$STOP
        B0                          53  55 F3 0005D 8$:   AOBLEQ  R5, INDEX, 1$                  ; 0933
                                    50  54 D0 00061       MOVL    RESULT, R0                      ; 0950
                                        04 00064          RET                                     ; 0951

; Routine Size:  101 bytes,    Routine Base:  $CODE$ + 04F2

;  969          0952  1
```

```
1 ROUTINE handler (signal_args, mechanism_args) =
1
1 !---
1 !
1 !       This condition handler gets control on any signalled
1 !       condition in order to save the highest severity error
1 !       to be returned by exit from the image.
1 !
1 ! Inputs:
1 !
1 !       signal_args = Address of signal argument list
1 !       mechanism_args = Address of mechanism argument list
1 !
1 ! Outputs:
1 !
1 !       worst_error is updated with highest severity error.
1 !
1 !---
1
2 BEGIN
2
2 MAP
2     signal_args: REF $BBLOCK,                  ! Adr of signal arg list
2     mechanism_args: REF $BBLOCK;                      ! Adr of mech. arg list
2
2 LOCAL
2     code: $BBLOCK [LONG];                      ! Condition code (longword)
2
2 code = .signal_args [chf$l_sig_name];          ! Get condition code
2
2 IF .code [sts$v_severity] GTR .worst_error [sts$v_severity]
2 THEN
2     worst_error = .code OR sts$m_inhib_msg;    ! Set new worst error
2
2 ss$_resignal                                   ! Continue signalling
2
1 END;



                                        0000 00000 HANDLER:.WORD   Save nothing                        ; 0953
                     50        04    AC   D0 00002         MOVL    SIGNAL_ARGS, R0                     ; 0981
                     50        04    A0   D0 00006         MOVL    4(R0), CODE                         ;
      51    0000'  CF        03            00   EF 0000A         EXTZV   #0, #3, WORST_ERROR, R1             ; 0983
      51            50        03            00   ED 00011         CMPZV   #0, #3, CODE, R1                    ;
                                     0A   15 00016         BLEQ    1$                                  ;
            0000'  CF        50 10000000    8F   C9 00018         BISL3   #268435456, CODE, WORST_ERROR       ; 0985
                     50      0918    8F   3C 00022 1$:     MOVZWL  #2328, R0                           ; 0989
                                          04 00027         RET                                         ;

; Routine Size:  40 bytes,    Routine Base:  $CODE$ + 0557
```

```
;  1009          0990  1 ROUTINE fab_error (fab): =
;  1010          0991  1
;  1011          0992  1 !----
;  1012          0993  1 !
;  1013          0994  1 ! Functional description
;  1014          0995  1 !
;  1015          0996  1 !       This routine is called from RMS whenever an error
;  1016          0997  1 !       occurs during an RMS file function call.
;  1017          0998  1 !
;  1018          0999  1 ! Input parameters
;  1019          1000  1 !
;  1020          1001  1 !       fab = Address of block used in the RMS call.
;  1021          1002  1 !       fab$l_nam = pointer to name block
;  1022          1003  1 !       fab$l_ctx = error message to be used
;  1023          1004  1 !
;  1024          1005  1 ! Output parameters
;  1025          1006  1 !
;  1026          1007  1 !       Expanded error messages to user
;  1027          1008  1 !       FAB status is RETURNed
;  1028          1009  1 !
;  1029          1010  1 !----
;  1030          1011  1
;  1031          1012  2 BEGIN
;  1032          1013  2
;  1033          1014  2 MAP fab: REF $BBLOCK;                   ! Define fab block format
;  1034          1015  2 BIND nam = .fab[fab$l_nam]: $BBLOCK;    ! Define name block
;  1035          1016  2
;  1036          1017  2 LOCAL desc: VECTOR[2];                  ! Temporary string descriptor
;  1037          1018  2
;  1038          1019  2 IF .nam[nam$b_rsl] NEQ 0                ! If result string nonblank,
;  1039          1020  3 THEN BEGIN
;  1040          1021  3     desc[0] = .nam[nam$b_rsl];          ! then display it
;  1041          1022  3     desc[1] = .nam[nam$l_rsa];
;  1042          1023  3     END
;  1043          1024  2 ELSE IF .nam[nam$b_esl] NEQ 0           ! Or if expanded name nonblank
;  1044          1025  3 THEN BEGIN
;  1045          1026  3     desc[0] = .nam[nam$b_esl];          ! then display it
;  1046          1027  3     desc[1] = .nam[nam$l_esa];
;  1047          1028  3     END
;  1048          1029  3 ELSE BEGIN
;  1049          1030  3     desc[0] = .fab[fab$b_fns];          ! Otherwise, use original
;  1050          1031  3     desc[1] = .fab[fab$l_fna];          ! name string in FAB
;  1051          1032  2     END;
;  1052          1033  2
;  1053        P 1034  2 write_message(.fab[fab$l_ctx],1,DESC,   ! Output an error message
;  1054        P 1035  2                 .fab[fab$l_sts],      ! with fab error code
;  1055          1036  2                 .fab[fab$l_stv]);     ! and secondary code
;  1056          1037  2
;  1057          1038  2 RETURN (.fab[fab$l_sts]);
;  1058          1039  2
;  1059          1040  1 END;


                                0004  00000 FAB_ERROR:
```

```
                                                   .WORD   Save R2                        ; 0990
                5E           08  C2 00002          SUBL2   #8, SP                         ;
                52      04   AC  D0 00005          MOVL    FAB, R2                        ; 1015
                50      28   A2  D0 00009          MOVL    40(R2), R0                     ;
                        03   A0  95 0000D          TSTB    3(R0)                          ; 1019
                             0B  13 00010          BEQL    1$                             ;
                6E      03   A0  9A 00012          MOVZBL  3(R0), DESC                    ; 1021
          04    AE      04   A0  D0 00016          MOVL    4(R0), DESC+4                  ; 1022
                             19  11 0001B          BRB     3$                             ; 1019
                        0B   A0  95 0001D 1$:      TSTB    11(R0)                         ; 1024
                             0B  13 00020          BEQL    2$                             ;
                6E      0B   A0  9A 00022          MOVZBL  11(R0), DESC                   ; 1026
          04    AE      0C   A0  D0 00026          MOVL    12(R0), DESC+4                 ; 1027
                             09  11 0002B          BRB     3$                             ; 1024
                6E      34   A2  9A 0002D 2$:      MOVZBL  52(R2), DESC                   ; 1030
          04    AE      2C   A2  D0 00031          MOVL    44(R2), DESC+4                 ; 1031
                7E      08   A2  7D 00036 3$:      MOVQ    8(R2), -(SP)                   ; 1036
                        08   AE  9F 0003A          PUSHAB  DESC                           ;
                             01  DD 0003D          PUSHL   #1                             ;
                        18   A2  DD 0003F          PUSHL   24(R2)                         ;
   00000000G    00           05  FB 00042          CALLS   #5, LIB$SIGNAL                 ;
                50      08   A2  D0 00049          MOVL    8(R2), R0                      ; 1038
                                 04 0004D          RET                                    ; 1040

; Routine Size:  78 bytes,    Routine Base:  $CODE$ + 057F
```

```
;1061     1041  1 ROUTINE rab_error (rab): =
;1062     1042  1
;1063     1043  1 !----
;1064     1044  1 !
;1065     1045  1 ! Functional description
;1066     1046  1 !
;1067     1047  1 !       This routine is called from RMS whenever an error
;1068     1048  1 !       occurs during an RMS record function call.
;1069     1049  1 !
;1070     1050  1 ! Input parameters
;1071     1051  1 !
;1072     1052  1 !       rab = Address of block used in the RMS call.
;1073     1053  1 !       rab$l_ctx = error message to be used
;1074     1054  1 !       rab$l_fab = pointer to fab block
;1075     1055  1 !       fab$l_nam = pointer to name block
;1076     1056  1 !
;1077     1057  1 ! Output parameters
;1078     1058  1 !
;1079     1059  1 !       Expanded error message to user
;1080     1060  1 !       RAB status is RETURNed
;1081     1061  1 !
;1082     1062  1 !----
;1083     1063  1
;1084     1064  2 BEGIN
;1085     1065  2
;1086     1066  2 MAP rab: REF $BBLOCK;                   ! Define RMS block format
;1087     1067  2 BIND
;1088     1068  2 fab = .rab[rab$l_fab] : $BBLOCK,        ! Define fab
;1089     1069  2 nam = .fab[fab$l_nam] : $BBLOCK;        ! Define nam
;1090     1070  2
;1091     1071  2 LOCAL desc: VECTOR[2];                  ! Temporary string descriptor
;1092     1072  2
;1093     1073  4 IF ((.rab [rab$l_sts] EQL rms$_eof)     ! If error is end of file
;1094     1074  3    AND                                  ! and
;1095     1075  3    (.rab[rab$l_ctx] EQL msg$_readerr))  ! this was a read call
;1096     1076  2 THEN                                    ! then
;1097     1077  2     RETURN (.rab[rab$l_sts]);          ! don't bother to report it
;1098     1078  2
;1099     1079  2 IF .nam[nam$b_rsl] NEQ 0                ! IF result string nonblank,
;1100     1080  3 THEN BEGIN
;1101     1081  3     desc[0] = .nam[nam$b_rsl];         ! then display it
;1102     1082  3     desc[1] = .nam[nam$l_rsa];
;1103     1083  3     END
;1104     1084  2 ELSE IF .nam[nam$b_esl] NEQ 0           ! Or if expanded name nonblank
;1105     1085  3 THEN BEGIN
;1106     1086  3     desc[0] = .nam[nam$b_esl];         ! then display it
;1107     1087  3     desc[1] = .nam[nam$l_esa];
;1108     1088  3     END
;1109     1089  3 ELSE BEGIN
;1110     1090  3     desc[0] = .fab[fab$b_fns];         ! Otherwise, use original
;1111     1091  3     desc[1] = .fab[fab$l_fna];         ! name string in FAB
;1112     1092  2     END;
;1113     1093  2
;1114   P 1094  2 write_message(.rab[rab$l_ctx],1,DESC,  ! Output an error message
;1115   P 1095  2                .rab[rab$l_sts],       ! with RMS error code
;1116     1096  2                .rab[rab$l_stv]);      ! and secondary code
;1117     1097  2
```

```
; 1118          1098  2 RETURN (.rab[rab$l_sts]);
; 1119          1099  2
; 1120          1100  1 END;


                                                     0004 00C00 RAB_ERROR:
                                                                        .WORD   Save R2                                 ; 1041
                        5E            08  C2 00002                      SUBL2   #8, SP                                  ;
                        52        04  AC  D0 00005                      MOVL    RAB, R2                                 ; 1068
                        51        3C  A2  D0 00009                      MOVL    60(R2), R1                              ;
                        50        28  A1  D0 0000D                      MOVL    40(R1), R0                              ; 1069
        0001827A        8F        08  A2  D1 00011                      CMPL    8(R2), #98938                           ; 1073
                                      0A  12 00019                      BNEQ    1$                                      ;
        009110B2        8F        18  A2  D1 0001B                      CMPL    24(R2), #9506994                        ; 1075
                                      3C  13 00023                      BEQL    5$                                      ;
                                  03  A0  95 00025  1$:                 TSTB    3(R0)                                   ; 1079
                                      0B  13 00028                      BEQL    2$                                      ;
                        6E        03  A0  9A 0002A                      MOVZBL  3(R0), DESC                             ; 1081
              04        AE        04  A0  D0 0002E                      MOVL    4(R0), DESC+4                           ; 1082
                                      19  11 00033                      BRB     4$                                      ; 1079
                                  0B  A0  95 00035  2$:                 TSTB    11(R0)                                  ; 1084
                                      0B  13 00038                      BEQL    3$                                      ;
                        6E        0B  A0  9A 0003A                      MOVZBL  11(R0), DESC                            ; 1086
              04        AE        0C  A0  D0 0003E                      MOVL    12(R0), DESC+4                          ; 1087
                                      09  11 00043                      BRB     4$                                      ; 1084
                        6E        34  A1  9A 00045  3$:                 MOVZBL  52(R1), DESC                            ; 1090
              04        AE        2C  A1  D0 00049                      MOVL    44(R1), DESC+4                          ; 1091
                        7E        08  A2  7D 0004E  4$:                 MOVQ    8(R2), -(SP)                            ; 1096
                                  08  AE  9F 00052                      PUSHAB  DESC                                    ;
                                      01  DD 00055                      PUSHL   #1                                      ;
                                  18  A2  DD 00057                      PUSHL   24(R2)                                  ;
        00000000G       00            05  FB 0005A                      CALLS   #5, LIB$SIGNAL                          ;
                        50        08  A2  D0 00061  5$:                 MOVL    8(R2), R0                               ; 1098
                                          04 00065                      RET                                             ; 1100

; Routine Size:  102 bytes,    Routine Base:  $CODE$ + 05CD
```

```
; 1122          1101  1 ROUTINE create_error (string_desc,status): =
; 1123          1102  1
; 1124          1103  1 !----
; 1125          1104  1 !
; 1126          1105  1 ! Functional description
; 1127          1106  1 !
; 1128          1107  1 !       This routine is called when an error is returned
; 1129          1108  1 !       by LIB$CREATE_DIR.
; 1130          1109  1 !
; 1131          1110  1 ! Input parameters
; 1132          1111  1 !
; 1133          1112  1 !       string_desc = address of a string descriptor for the name
; 1134          1113  1 !         of the directory being processed
; 1135          1114  1 !       status = error status passed by caller
; 1136          1115  1 !
; 1137          1116  1 ! Output parameters
; 1138          1117  1 !
; 1139          1118  1 !       Expanded error message to user
; 1140          1119  1 !       Input status is RETURNed
; 1141          1120  1 !
; 1142          1121  1 !----
; 1143          1122  1
; 1144          1123  2 BEGIN
; 1145          1124  2
; 1146          1125  2 MAP string_desc: REF $BBLOCK;
; 1147          1126  2
; 1148          1127  2 write_message(msg$_dirnotcre,1,.string_desc,.status);
; 1149          1128  2
; 1150          1129  2 RETURN (.status);
; 1151          1130  2
; 1152          1131  1 END;


                                    0000 00000 CREATE_ERROR:
                                                      .WORD   Save nothing                  ; 1101
                  7E         04  AC   7D 00002        MOVQ    STRING_DESC, -(SP)            ; 1127
                                      01   DD 00006   PUSHL   #1                            ;
                     009112C2     8F  DD 00008        PUSHL   #9507522                      ;
         00000000G   00           04  FB 0000E        CALLS   #4, LIB$SIGNAL                ;
                     50      08  AC   D0 00015        MOVL    STATUS, R0                    ; 1129
                                      04 00019        RET                                   ; 1131

; Routine Size:  26 bytes,    Routine Base:  $CODE$ + 0633
```

```
;  1154          1132  1 ROUTINE vm_error (status): =
;  1155          1133  1
;  1156          1134  1 !----
;  1157          1135  1 !
;  1158          1136  1 ! Functional description
;  1159          1137  1 !
;  1160          1138  1 !       This routine is called when an error is returned
;  1161          1139  1 !       by LIB$GET_VM or LIB$FREE_VM
;  1162          1140  1 !
;  1163          1141  1 ! Input parameters
;  1164          1142  1 !
;  1165          1143  1 !       status = error status passed by caller
;  1166          1144  1 !
;  1167          1145  1 ! Output parameters
;  1168          1146  1 !
;  1169          1147  1 !       Expanded error message to user
;  1170          1148  1 !       Input status is RETURNed
;  1171          1149  1 !
;  1172          1150  1 !----
;  1173          1151  1
;  1174          1152  2 BEGIN
;  1175          1153  2
;  1176          1154  2 write_message(.status);
;  1177          1155  2
;  1178          1156  2 RETURN (.status);
;  1179          1157  2
;  1180          1158  1 END;


                                0000 00000 VM_ERROR:
                                           .WORD   Save nothing                    ;  1132
                           04   AC  DD 00002       PUSHL   STATUS                  ;  1154
                00000000G  00       01  FB 00005  CALLS   #1, LIB$SIGNAL          ;
                           50   04  AC  D0 0000C  MOVL    STATUS, R0              ;  1156
                                    04 00010       RET                             ;  1158

; Routine Size:  17 bytes,    Routine Base:  $CODE$ + 064D
```

```
; 1182          1159  1 END
; 1183          1160  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL, LIB$STOP

;                               PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;   $OWN$                         1788  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $PLIT$                         396  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   _LIB$KEY0$                       2  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;   _LIB$STATE$                     12  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;   _LIB$KEY1$                       8  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(1)
;   $CODE$                        1630  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,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]STARLET.L32;1     9776       119         1         581        00:01.0
;   _$255$DUA28:[SYSLIB]TPAMAC.L32;1        42        22        52          14        00:00.1




;                                   COMMAND QUALIFIERS

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

; Size:         1630 code + 2206 data bytes
; Run Time:       00:36.3
; Elapsed Time:   01:54.8
; Lines/CPU Min:   1917
; Lexemes/CPU-Min: 32065
; Memory Used:  225 pages
; Compilation Complete
```

JBCPRSDEF
REQ

CNVCLIATB
LIS

INFO
LIS

TYPE
REQ

CHRSUB
LIS

CNVCLINUM
LIS

SHODEVDEF
REQ

CLIMAC
MAR

CNVCLIFRM
LIS

DIGRAMS
LIS

CALCMAX
LIS

JBCCMDPRS
LIS

CLIUTLMAC
MAR

CVTTIME
LIS

SHOWDEF
REQ

CREATE
LIS