```
DDDDDDDDDDD         CCCCCCCCCCC  LLL
DDDDDDDDDDDD        CCCCCCCCCCCC LLL
DDDDDDDDDDDD        CCCCCCCCCCCC LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDDDDDDDDDDD        CCCCCCCCCCCC LLLLLLLLLLLLLLL
DDDDDDDDDDDD        CCCCCCCCCCCC LLLLLLLLLLLLLLL
DDDDDDDDDDDD        CCCCCCCCCCCC LLLLLLLLLLLLLLL
```

FILEID**FILECMDS

N 3

```
FFFFFFFFFF    IIIIII    LL          EEEEEEEEEE    CCCCCCCC  MM      MM  DDDDDDDD     SSSSSSSS
FFFFFFFFFF    IIIIII    LL          EEEEEEEEEE    CCCCCCCC  MM      MM  DDDDDDDD     SSSSSSSS
FF              II      LL          EE          CC          MMMM  MMMM  DD      DD   SS
FF              II      LL          EE          CC          MMMM  MMMM  DD      DD   SS
FF              II      LL          EE          CC          MM  MM  MM  DD      DD   SS
FF              II      LL          EE          CC          MM  MM  MM  DD      DD   SS
FFFFFFFF        II      LL          EEEEEEEE    CC          MM      MM  DD      DD     SSSSSS
FFFFFFFF        II      LL          EEEEEEEE    CC          MM      MM  DD      DD     SSSSSS
FF              II      LL          EE          CC          MM      MM  DD      DD           SS
FF              II      LL          EE          CC          MM      MM  DD      DD           SS
FF              II      LL          EE          CC          MM      MM  DD      DD           SS      ....
FF              II      LL          EE          CC          MM      MM  DD      DD           SS      ....
FF            IIIIII    LLLLLLLLLL  EEEEEEEEEE    CCCCCCCC  MM      MM  DDDDDDDD     SSSSSSSS      ....
FF            IIIIII    LLLLLLLLLL  EEEEEEEEEE    CCCCCCCC  MM      MM  DDDDDDDD     SSSSSSSS      ....

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

Table of contents

```
0000     1          .TITLE  FILECMDS - FILE I/O COMMAND EXECUTION
0000     2          .IDENT  'V04-000'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY        *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19 ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28 ; FILE I/O DCLS COMMAND EXECUTION
0000    29 ;
0000    30 ;       CLOSE FILE
0000    31 ;       OPEN FILE
0000    32 ;       READ FILE
0000    33 ;       WRITE FILE
0000    34 ;
0000    35 ; D. N. CUTLER 12-FEB-78
0000    36 ;
0000    37 ; MODIFIED BY:
0000    38 ;
0000    39 ;       V03-017 HWS0104         Harold Schultz          21-Aug-1984
0000    40 ;               Fix accvio in "invalid symbol" error path in READ
0000    41 ;               command.
0000    42 ;
0000    43 ;       V03-016 HWS0067         Harold Schultz          24-May-1984
0000    44 ;               Enable WRITE/SYMBOL to also handle quoted expressions
0000    45 ;               in addition to just symbols.
0000    46 ;
0000    47 ;       V03-015 HWS0063         Harold Schultz          27-Apr-1984
0000    48 ;               Initialize TIME_OUT flag in READ.
0000    49 ;
0000    50 ;       V03-014 HWS0050         Harold Schultz          03-Apr-1984
0000    51 ;               Add READ/TIME_OUT=n.
0000    52 ;               Add CLOSE/LOG.
0000    53 ;
0000    54 ;       V03-013 PCG0013         Peter George            14-Mar-1984
0000    55 ;               Get #11 right.  Allow writes to level-0 SYS$OUTPUT.
0000    56 ;               Fix PSL$C_USER access mode setting. Save status in
0000    57 ;               WRITE command when expanding symbols.
```

```
0000     58 ;
0000     59 ;       V03-012 HWS0023         Harold Schultz          08-Mar-1984
0000     60 ;               Set PSL$C_USER in FAB$V_FILE_MODE prior to opening a file.
0000     61 ;               Add /SYMBOL to WRITE command to allow output of symbols as
0000     62 ;               well as expressions.
0000     63 ;
0000     64 ;       V03-011 PCG0012         Peter George            25-Sep-1983
0000     65 ;               Allow writes to level-0 SYS$OUTPUT.
0000     66 ;
0000     67 ;       V03-010 PCG0011         Peter George            07-Sep-1983
0000     68 ;               Use new logical name services to create and delete
0000     69 ;               file logical names. Do not set PRN RAT attribute when
0000     70 ;               opening a file for read-only access.
0000     71 ;
0000     72 ;       V03-009 PCG0010         Peter George            15-Jul-1983
0000     73 ;               Modify READ symbol validation algorithm.
0000     74 ;
0000     75 ;       V03-008 PCG0009         Peter George            15-Jun-1983
0000     76 ;               Allow expressions in WRITE command.
0000     77 ;
0000     78 ;       V03-007 PCG0008         Peter George            01-Mar-1983
0000     79 ;               Call DCL$GETNVAL.
0000     80 ;               Understand that a terminal XAB exists.
0000     81 ;
0000     82 ;       V03-006 PCG0007         Peter George            14-Feb-1983
0000     83 ;               Fix OPEN/APPEND flag bit bug.
0000     84 ;
0000     85 ;       V03-005 PCG0006         Peter George            11-Jan-1983
0000     86 ;               Remove all references to ERRIFI and ERRRAB.
0000     87 ;               Fix READ/EOF bug.
0000     88 ;
0000     89 ;       V03-004 PCG0005         Peter George            07-Jan-1983
0000     90 ;               Fix WRITE/UPDATE accvio.
0000     91 ;               Explicitly reset READ/MATCH mode.
0000     92 ;
0000     93 ;       V03-003 PCG0004         Peter George            29-Dec-1982
0000     94 ;               Allow + to delimit WRITE command parameters.
0000     95 ;               Use DCL$SAVE_STATUS instead of DCL$SET_STATUS.
0000     96 ;               Add READ/MATCH.
0000     97 ;
0000     98 ;       V03-002 PCG0003         Peter George            03-Dec-1982
0000     99 ;               Allocate write buffer on the stack.
0000    100 ;
0000    101 ;       V03-001 PCG0002         Peter George            23-Nov-1982
0000    102 ;               Use WRK_G_BUFFER instead of WRK_G_RECORD.
0000    103 ;---
```

```
                        0000    105 ;
                        0000    106 ; MACRO LIBRARY CALLS
                        0000    107 ;
                        0000    108
                        0000    109         PRCDEF                          ;DEFINE PROCESS WORK AREA
                        0000    110         WRKDEF                          ;DEFINE COMMAND WORK AREA
                        0000    111         PTRDEF                          ;DEFINE RESULT PARSE DESCRIPTOR FORMAT
                        0000    112         IDFDEF                          ;DEFINE INDIRECT FRAME OFFSETS
                        0000    113         SYMDEF                          ;DEFINE TYPES OF SYMBOLS
                        0000    114         $CLIMSGDEF                      ;DEFINE ERROR/STATUS CODES
                        0000    115         $FABDEF                         ;DEFINE FAB OFFSETS
                        0000    116         $LNMDEF                         ;DEFINE LNM OFFSETS
                        0000    117         $NAMDEF                         ;DEFINE NAME BLOCK OFFSETS
                        000C    118         $PSLDEF                         ;DEFINE PROCESSOR STATUS FIELDS
                        000C    119         $RABDEF                         ;DEFINE RAB OFFSETS
                        000C    120         $STSDEF                         ;DEFINE STATUS LONG WORD VALUES
                        0000    121
                        0000    122 ;
                        0000    123 ; PROCESS PERMANENT FILE PREFIX
                        0000    124 ;
                        0000    125
                0000001B 0000   126 ESCAPE=27
                        0000    127
                00000000 0000   128         .PSECT  DCL$ZCODE,BYTE,RD,NOWRT
                        0000    129 ;
                        0000    130 ; DATA INPUT PROMPT
                        0000    131 ;
09 3A 61 74 61 44 00'   0000    132 DATAP:  .ASCIC  'Data:  '
                     06 0000
                        0007    133
                        0007    134 ;
                        0007    135 ; DEFAULT FILE NAME STRING
                        0007    136 ;
                  54 41 44 2E 00' 0007  137 DFSPEC: .ASCIC  '.DAT'
                     04 0007
                        000C    138 ;
                        000C    139 ; KEYWORDS FOR OPEN/SHARE
                        000C    140 ;
                     44 41 45 52 000C   141 READ:   .ASCII  'READ'
                     54 49 52 57 0010   142 WRITE:  .ASCII  'WRIT'
                        0014    143
                        0014    144 LNM$PROCESS:
53 53 45 43 4F 52 50 24 4D 4E 4C 00' 0014  145         .ASCIC  'LNM$PROCESS'
                     0B 0014
```

```
                        0020   147          .SBTTL  CLOSE FILE
                        0020   148 ;+
                        0020   149 ; DCL$CLOSE - CLOSE FILE
                        0020   150 ;
                        0020   151 ; THIS ROUTINE IS CALLED AS AN INTERNAL COMMAND TO EXECUTE THE CLOSE FILE DCLS
                        0020   152 ; COMMAND.
                        0020   153 ;
                        0020   154 ; INPUTS:
                        0020   155 ;
                        0020   156 ;       R8 = ADDRESS OF SCRATCH BUFFER DESCRIPTOR.
                        0020   157 ;       R9 = ADDRESS OF SCRATCH STACK.
                        0020   158 ;       R10 = BASE ADDRESS OF COMMAND WORK AREA.
                        0020   159 ;       R11 = BASE ADDRESS OF PROCESS WORK AREA.
                        0020   160 ;
                        0020   161 ; OUTPUTS:
                        0020   162 ;
                        0020   163 ;       THE SPECIFIED FILE IS CLOSED.
                        0020   164 ;-
                        0020   165 DCL$CLOSE::                                  ;CLOSE FILE
                        0020   166          CLRBIT  WRK_V_CLOSERR,WRK_W_FLAGS2(R10) ;ASSUME /LOG
                        0024   167 ;
                        0024   168 ; ***** WARNING, GETID "RETURNS" VIA CO-ROUTINE CALL, NOTHING MAY BE ON THE STACK
                        0024   169 ;
                 063C   30      0024   170          BSBW    GETID                   ;GET FILE IDENTIFICATION
               32 50    E8      0027   171          BLBS    R0,20$                  ;IF LBS FILE ID FOUND
                        002A   172 ;
                        002A   173 ; HAVE AN UNDEFINED FILE STATUS ERROR. CHECK FOR /NOLOG IN COMMAND LINE
                        002A   174 ;
      BA AA     F9B6 CA   9E    002A   175          MOVAB   WRK_G_RESULT(R10),WRK_L_RSLNXT(R10)  ;RELOAD TOKEN POINTER
         7E       50   D0    0030   176          MOVL    R0,-(SP)                ;SAVE STATUS
                FFCA'  30      0033   177 10$:     BSBW    DCL$GETDVAL             ;GET NEXT TOKEN IN COMMAND LINE
            55    04   D1      0036   178          CMPL    #PTR_K_ENDLINE,R5       ;END OF THE LINE?
                  1C   13      0039   179          BEQL    12$                     ;YES, GOTO EXIT
            55    00   D1      003B   180          CMPL    #PTR_K_COMDQUAL,R5      ;COMMAND QUALIFIER?
                  F3   12      003E   181          BNEQ    10$                     ;NO, CHECK NEXT TOKEN
                FFBD'  30      0040   182          BSBW    DCL$GETNVAL             ;GET QUALIFIER NUMBER
         51   00'8F   91      0043   183          CMPB    #CLI$K_CLOS_LOG,R1      ;IS IT /LOG?
                  EA   12      0047   184          BNEQ    10$                     ;NO, CHECK NEXT TOKEN
                       0049   185
                       0049   186          CLRBIT  WRK_V_CLOSERR,WRK_W_FLAGS2(R10) ;ASSUME /LOG
      E2 53     00   E1      004D   187          BBC     #PTR_V_NEGATE-PTR_V_FLAGS,R3,10$  ;SKIP REST IF /LOG
                       0051   188          SETBIT  WRK_V_CLOSERR,WRK_W_FLAGS2(R10) ;INDICATE /NOLOG
                  DC   11      0055   189          BRB     10$                     ;CHECK NEXT TOKEN
                       0057   190
            50    8E   D0      0057   191 12$:     MOVL    (SP)+,R0                ;RESTORE STATUS
                  48   11      005A   192          BRB     30$
                       005C   193
                05CE   30      005C   194 20$:     BSBW    CHECKPPF                ;CHECK FOR PROCESS PERMANENT FILE
                  43   13      005F   195          BEQL    30$                     ;IF EQL, YES
                       0061   196 ;
                       0061   197 ; DELETE LOGICAL NAME.
                       0061   198 ;
                  02   DD      0061   199          PUSHL   #PSL$C_SUPER            ;PUSH ACCESS MODE
         7E       53   7D      0063   200          MOVQ    R3,-(SP)                ;BUILD LOGICAL NAME DECRIPTOR
         51     AB AF   9E      0066   201          MOVAB   LNM$PROCESS,R1          ;BUILD TABLE NAME DESCRIPTOR
            50    81   9A      006A   202          MOVZBL  (R1)+,R0                ;
            7E    50   7D      006D   203          MOVQ    R0,-(SP)                ;
```

```
                50    5E    D0  0070    204          MOVL    SP,R0                           ;SAVE ADDRESS OF DESCRIPTORS
                                0073    205          $DELLNM_S  TABNAM=(R0),-                ;DELETE THE LOGICAL NAME
                                0073    206                     LOGNAM=8(R0),-           ;
                                0073    207                     ACMODE=16(R0)            ;
                5E    14    C0  0082    208          ADDL    #4*5,SP                         ;CLEAN STACK
                                0085    209
                67    79    D0  0085    210          MOVL    -(R9),(R7)                      ;REMOVE FILE DESCRIPTOR FROM LIST
             58    1C AB    D0  0088    211          MOVL    PRC_L_INDFAB(R11),R8            ;GET ADDRESS OF INDIRECT FAB
          02 A8    1C A9    B0  008C    212          MOVW    RAB$L_CTX+4(R9),FAB$W_IFI(R8) ;INSERT INTERNAL FILE INDEX
                50    59    D0  0091    213          MOVL    R9,R0                           ;SET ADDRESS OF BLOCK TO DEALLOCATE
             51    48 8F    9A  0094    214          MOVZBL  #<RAB$C_BLN+4+7>&^C<7>,R1 ;SET LENGTH OF BLOCK TO DEALLOCATE
                   FF65'     30  0098    215          BSBW    DCL$DEADYNMEM                   ;DEALLOCATE FILE DESCRIPTOR BLOCK
                                009B    216          $CLOSE  FAB=(R8)                        ;CLOSE FILE
                          05    00A4    217 30$:     RSB                                     ;
```

```
                     00A5   219          .SBTTL  OPEN FILE
                     00A5   220 ;+
                     00A5   221 ; DCL$OPEN - OPEN FILE
                     00A5   222 ;
                     00A5   223 ; THIS ROUTINE IS CALLED AS AN INTERNAL COMMAND TO EXECUTE THE OPEN FILE DCLS
                     00A5   224 ; COMMAND.
                     00A5   225 ;
                     00A5   226 ; INPUTS:
                     00A5   227 ;
                     00A5   228 ;       R8 = ADDRESS OF SCRATCH BUFFER DESCRIPTOR.
                     00A5   229 ;       R9 = ADDRESS OF SCRATCH STACK.
                     00A5   230 ;       R10 = BASE ADDRESS OF COMMAND WORK AREA.
                     00A5   231 ;       R11 = BASE ADDRESS OF PROCESS WORK AREA.
                     00A5   232 ;
                     00A5   233 ; OUTPUTS:
                     00A5   234 ;
                     00A5   235 ;       THE SPECIFIED FILE IS OPENED AND A FILE DESCRIPTOR IS CONSTRUCTED.
                     00A5   236 ;-
                     00A5   237
                     00A5   238 DCL$OPEN::                                  ;OPEN FILE
 58   1C AB    D0   00A5   239          MOVL    PRC_L_INDFAB(R11),R8       ;GET ADDRESS OF INDIRECT FAB
      02 A8    B4   00A9   240          CLRW    FAB$W_IFI(R8)              ;CLEAR INTERNAL FILE INDEX
      36 A8    B4   00AC   241          CLRW    FAB$W_MRS(R8)              ;USED FOR FLAGS TO "OPEN_CREATE"
      10 A8    D4   00AF   242          CLRL    FAB$L_ALQ(R8)              ;CLEAR ALLOCATION QUANTITY
      14 A8    B4   00B2   243          CLRW    FAB$W_DEQ(R8)              ;CLEAR DEFAULT FILE EXTENSION QUANTITY
      16 A8    94   00B5   244          CLRB    FAB$B_FAC(R8)              ;CLEAR FILE ACCESS TYPE
      17 A8    94   00B8   245          CLRB    FAB$B_SHR(R8)              ;CLEAR FILE SHARING
   04   03    F0   00BB   246          INSV    #PSL$C_USER,#FAB$V_FILE_MODE,-  ;SET FILE ACCESS PROTECTION MODE
   4A A8    02   00BE   247                  #FAB$S_FILE_MODE,FAB$B_ACMODES(R8)      ;
04 A8  00040000 8F   D0   00C1   248          MOVL    #FAB$M_PPF,FAB$L_FOP(R8) ;SET FILE OPEN OPTIONS
   35 A8    FF3A CF   90   00C9   249          MOVB    DFSPEC,FAB$B_DNS(R8)     ;SET SIZE OF DEFAULT NAME STRING
   30 A8    FF35 CF   9E   00CF   250          MOVAB   DFSPEC+1,FAB$L_DNA(R8)   ;SET ADDRESS OF DEFAULT NAME STRING
      1D A8    00   90   00D5   251          MOVB    #FAB$C_SEQ,FAB$B_ORG(R8);SET FILE ORGANIZATION ASSUMING CREATE
         1E A8    94   00D9   252          CLRB    FAB$B_RAT(R8)            ;CLEAR RECORD ATTRIBUTE TYPE
      1F A8    03   90   00DC   253          MOVB    #FAB$C_VFC,FAB$B_RFM(R8) ;SET RECORD FORMAT
            FF1D'  30   00E0   254 20$:      BSBW    DCL$GETDVAL              ;GET NEXT DESCRIPTOR VALUES
         55   04   D1   00E3   255          CMPL    #PTR_K_ENDLINE,R5        ;END OF LINE?
              03   12   00E6   256          BNEQ    21$                      ;BRANCH IF NOT
            0089   31   00E8   257          BRW     30$                      ;BRANCH IF DONE WITH SCAN
         55   00   D1   00EB   258 21$:      CMPL    #PTR_K_COMDQUAL,R5       ;COMMAND QUALIFIER?
              F0   12   00EE   259          BNEQ    20$                      ;IF NEQ NO
            FF0D'  30   00F0   260          BSBW    DCL$GETNVAL              ;GET QUALIFIER NUMBER
      51   00'8F   91   00F3   261          CMPB    #CLI$K_OPEN_READ,R1      ;READ QUALIFER?
              1E   13   00F7   262          BEQL    10$                      ;IF EQL YES
      51   00'8F   91   00F9   263          CMPB    #CLI$K_OPEN_WRIT,R1      ;WRITE QUALIFIER?
              30   13   00FD   264          BEQL    25$                      ;BRANCH IF YES
      51   00'8F   91   00FF   265          CMPB    #CLI$K_OPEN_APPE,R1      ;APPEND QUALIFIER?
              1C   13   0103   266          BEQL    26$                      ;BRANCH IF YES
      51   00'8F   91   0105   267          CMPB    #CLI$K_OPEN_SHAR,R1      ;SHARE QUALIFIER?
              32   13   0109   268          BEQL    28$                      ;BRANCH IF YES
      51   00'8F   91   010B   269          CMPB    #CLI$K_OPEN_ERRO,R1      ;SUPRESS ERROR PRINTOUT?
              CF   12   010F   270          BNEQ    20$                      ;IF NEQ NO
      36 A8    02   A8   0111   271          BISW    #2,FAB$W_MRS(R8)         ;YES, SET PARAM TO "OPEN_CREATE"
              C9   11   0115   272          BRB     20$
      16 A8    02   88   0117   273 10$:     BISB    #FAB$M_GET,FAB$B_FAC(R8) ;MARK /READ EXPLICITLY SPECIFIED
      36 A8    01   AA   011B   274          BICW    #1,FAB$W_MRS(R8)         ;USE $OPEN IN "OPEN_CREATE"
              BF   11   011F   275 11$:     BRB     20$
```

```
                36 A8    08    A8  0121   276 26$:    BISW    #8,FAB$W_MRS(R8)          ;SET EOF ROP BIT ON CONNECT (/APPEND)
                16 A8    0D    88  0125   277         BISB    #FAB$M_PUT!FAB$M_DEL!FAB$M_UPD,FAB$B_FAC(R8) ;MARK WRITABLE
                1E A8    04    90  0129   278         MOVB    #FAB$M_PRN,FAB$B_RAT(R8) ;SET RECORD ATTRIBUTE TYPE
                         B1    11  012D   279         BRB     20$
                16 A8    0D    88  012F   280 25$:    BISB    #FAB$M_PUT!FAB$M_DEL!FAB$M_UPD,FAB$B_FAC(R8) ;MARK /WRITE EXPLICITLY
                1E A8    04    90  0133   281         MOVB    #FAB$M_PRN,FAB$B_RAT(R8) ;SET RECORD ATTRIBUTE TYPE
                36 A8    01    A8  0137   282         BISW    #1,FAB$W_MRS(R8)          ;USE $CREATE IN "OPEN_CREATE"
                         A3    11  013B   283         BRB     20$                       ;
                17 A8    0F    90  013D   284 28$:    MOVB    #FAB$M_GET!FAB$M_PUT!FAB$M_DEL!FAB$M_UPD,FAB$B_SHR(R8) ;ASSUME /SHAR
                   02    54    91  0141   285         CMPB    R4,#PTR_K_COLON           ;DOES A VALUE FOLLOW /SHARE?
                         9A    12  0144   286         BNEQ    20$                       ;IF NOT, GO WITH DEFAULT
                      FEB7'    30  0146   287         BSBW    DCL$GETDVAL               ;GET REQUIRED VALUE OF QUALIFIER
                      04    51  D1  0149   288         CMPL    R1,#4                     ;MORE THAN 4 CHARACTERS?
                         03    15  014C   289         BLEQ    29$                       ;BRANCH IF NOT
                      51    04  D0  014E   290         MOVL    #4,R1                     ;TRUNCATE TO 4 CHARACTER KEYWORD
                      56    51  7D  0151   291 29$:    MOVQ    R1,R6                     ;SAVE DESCRIPTOR
                17 A8    02    90  0154   292         MOVB    #FAB$M_GET,FAB$B_SHR(R8) ;ALLOW OTHERS TO READ FILE
FEAE CF         67    56    29  0158   293         CMPC3   R6,(R7),READ              ;/SHARE=READ?
                         80    13  015E   294         BEQL    20$                       ;BRANCH IF YES
                17 A8    0F    90  0160   295         MOVB    #FAB$M_GET!FAB$M_PUT!FAB$M_DEL!FAB$M_UPD,FAB$B_SHR(R8) ;ALLOW OTHERS
FEA6 CF         67    56    29  0164   296         CMPC3   R6,(R7),WRITE             ;/SHARE=WRITE?
                         B3    13  016A   297         BEQL    11$                       ;BRANCH IF YES
                                   016C   298         STATUS  IVKEYW                    ;INVALID KEYWORD
                               05  0173   299         RSB
                                   0174   300
             04 16 A8    01    E1  0174   301 30$:    BBC     #FAB$V_GET,FAB$B_FAC(R8),40$ ;IF /READ EXPLICITLY PRESENT,
                36 A8    01    AA  0179   302         BICW    #1,FAB$W_MRS(R8)          ;USE $OPEN EVEN IF /WRITE SPECIFIED TOO
                   16 A8    95  017D   303 40$:    TSTB    FAB$B_FAC(R8)             ;WERE EITHER /READ OR /WRITE SPECIFIED?
                         04    12  0180   304         BNEQ    45$                       ;BRANCH IF AT LEAST ONE WAS
                16 A8    02    88  0182   305         BISB    #FAB$M_GET,FAB$B_FAC(R8) ;ASSUME /READ IF NOTHING SPECIFIED
                                   0186   306 ;
                                   0186   307 ; ***** WARNING, GETID "RETURNS" VIA CO-ROUTINE CALL, NOTHING MAY BE ON THE STACK
                                   0186   308 ;
                      04DA    30  0186   309 45$:    BSBW    GETID                     ;GET FILE ID
                   64 50    E8  0189   310         BLBS    R0,60$                    ;IF LBS FILE ALREADY DEFINED
                56    53    7D  018C   311         MOVQ    R3,R6                     ;SAVE LOGICAL NAME PARAMETERS
             51    48 8F    9A  018F   312         MOVZBL  #RAB$C_BLN+4,R1           ;GET SIZE OF BLOCK REQUIRED
                   FE6A'    30  0193   313         BSBW    DCL$ALLDYNMEM             ;ALLOCATE FILE DESCRIPTOR BLOCK
                   50 50    E9  0196   314         BLBC    R0,50$                    ;IF LBC ALLOCATION FAILURE
                59    52    D0  0199   315         MOVL    R2,R9                     ;SAVE ADDRESS OF ALLOCATED BLOCK
                   FE61'    30  019C   316         BSBW    DCL$GETDVAL               ;GET FILE DESCRIPTOR VALUES
             34 A8    51    90  019F   317         MOVB    R1,FAB$B_FNS(R8)          ;SET SIZE OF FILE NAME
             2C A8    52    D0  01A3   318         MOVL    R2,FAB$L_FNA(R8)          ;SET ADDRESS OF FILE NAME
             52    36 A8    3C  01A7   319         MOVZWL  FAB$W_MRS(R8),R2          ;GET FLAGS
                   36 A8    B4  01AB   320         CLRW    FAB$W_MRS(R8)             ;NO MAXIMUM RECORD SIZE
             68 AB    04    A8  01AE   321         BISW    #PRC_M_DISABL,PRC_W_FLAGS(R11) ;DISABLE CONTROL Y/C AST'S
                50    58    D0  01B2   322         MOVL    R8,R0                     ;FAB ADDRESS TO OPEN/CREATE
                51    52    D0  01B5   323         MOVL    R2,R1                     ;PASS OPEN/CREATE FLAGS (LOW 2 BITS)
                      03D8    30  01B8   324         BSBW    DCL$OPEN_CREATE           ;OPEN/CREATE THE FILE
                   59 50    E9  01BB   325         BLBC    R0,90$                    ;BRANCH IF ERROR
50    000388D2 8F    DO    01BE   326         MOVL    #CLI$_INVRFM,R0           ;ASSUME RECORD FORMAT IS UNDEFINED
             1F A8    00    91  01C5   327         CMPB    #FAB$C_UDF,FAB$B_RFM(R8);RECORD FORMAT UNDEFINED?
                         3E    13  01C9   328         BEQL    80$                       ;YES, THEN CLOSE FILE AND SET STATUS
                   23 52    E8  01CB   329         BLBS    R2,70$                    ;BRANCH IF CREATE
                                   01CE   330
                                   01CE   331 ;
                                   01CE   332 ; OPEN FILE FOR READ ACCESS
```

```
                                01CE    333 ;
                             52   DD  01CE    334          PUSHL   R2                              ;SAVE FLAGS
                 04 A9   08 BB   0044 8F   28  01D0    335          MOVC    #RAB$C_BLN,@PRC_L_INPRAB(R11),4(R9) ;COPY INPUT RAB
                             52 8ED0  01D8    336          POPL    R2                              ;RESTORE FLAGS
                          08 A9   D4  01DB    337          CLRL    RAB$L_ROP+4(R9)                 ;CLEAR ROP
                       17 52      03   E1  01DE    338          BBC     #3,R2,75$                       ;/APPEND?
                                       01E2    339          SETBIT  RAB$V_EOF,RAB$L_ROP+4(R9) ;SET EOF BIT FOR CONNECT
                                10     11  01E7    340          BRB     75$                             ;PERFORM THE CONNECT
                                       01E9    341
                                       01E9    342 ;
                                       01E9    343 ; SYMBOL TABLE OVERFLOW
                                       01E9    344 ;
                                       01E9    345
                                       01E9    346 50$:     STATUS  SYMOVF                          ;SET SYMBOL TABLE OVERFLOW
                                05     01F0    347 60$:     RSB                                     ;
                                       01F1    348
                                       01F1    349 ;
                                       01F1    350 ; OPEN FILE FOR WRITE ACCESS
                                       01F1    351 ;
                                       01F1    352
                 04 A9   0C BB   0044 8F   28  01F1    353 70$:     MOVC    #RAB$C_BLN,@PRC_L_OUTRAB(R11),4(R9) ;COPY OUTPUT RAB
                          06 A9   B4  01F9    354 75$:     CLRW    RAB$W_ISI+4(R9)                 ;CLEAR INTERNAL STREAM INDEX
                                       01FC    355          $CONNECT RAB=4(R9)                      ;CONNECT RECORD STREAM
                                1E 50  E8  0206    356          BLBS    R0,100$                         ;IF LBS SUCCESSFUL CONNECT
                                       0209    357
                                       0209    358 ;
                                       0209    359 ; CONNECT OR LOGICAL NAME CREATION FAILURE
                                       0209    360 ;
                                       020C    361
                                50     DD  02      362 80$:     PUSHL   R0                              ;SAVE STATUS
                                       02      363          $CLOSE  FAB=(R8)                        ;CLOSE FILE
                                50 8ED0  021      364          POPL    R0                              ;RETREIVE STATUS
                                       0217    365
                                       0217    366 ;
                                       0217    367 ; OPEN FAILURE
                                       0217    368 ;
                                       0217    369
                                50     DD  0217    370 90$:     PUSHL   R0                              ;SAVE STATUS
                             50 59     D0  0219    371          MOVL    R9,R0                           ;SET ADDRESS OF BLOCK TO DEALLOCATE
                          51   48 8F     9A  021C    372          MOVZBL  #<RAB$C_BLN+4+7>&^C<7>,R1 ;SET SIZE OF BLOCK TO RELEASE
                               FDDD'     30  0220    373          BSBW    DCL$DEADYNMEM                   ;DEALLOCATE FILE DESCRIPTOR BLOCK
                                50 8ED0  0223    374          POPL    R0                              ;RESTORE STATUS
                                05     0226    375          RSB                                     ;
                                       0227    376
                                       0227    377 ;
                                       0227    378 ; SET IMPLIED CARRIAGE CONTROL IN ISI VALUE
                                       0227    379 ;
                                       0227    380
                             06 02     F0  0227    381 100$:    INSV    #FAB$M_CR,#RAB$V_PPF_RAT,- ;SET IMPLIED CARRIAGE CONTROL
                          06 A9   08         022A    382                  #RAB$S_PPF_RAT,RAB$W_ISI+4(R9) ;
                                       022D    383
                                       022D    384 ;
                                       022D    385 ; FILE SUCCESSFULLY OPEN - CREATE LOGICAL NAME AND MERGE INTO
                                       022D    386 ; FILE DESCRIPTOR LIST
                                       022D    387 ;
                          51   28 A8     D0  022D    388          MOVL    FAB$L_NAM(R8),R1        ;GET ADDRESS OF NAME BLOCK
                             5E   10     C2  0231    389          SUBL    #16,SP                  ;ALLOCATE SPACE TO STORE DEVICE NAME
```

```
                50   14 A1   9A  0234   390          MOVZBL  NAM$T_DVI(R1),R0          ;GET LENGTH OF DEVICE NAME
                  7E   50   7D  0238   391          MOVQ    R0,-(SP)                 ;SAVE VOLATILE REGISTERS
          08 AE   15 A1   50   28  023B   392          MOVC    R0,NAM$T_DVI+1(R1),8(SP) ;COPY DEVICE IDENTIFICATION
                    63   3A   90  0241   393          MOVB    #^A/:/,(R3)              ;APPEND A :
                    50   8E   7D  0244   394          MOVQ    (SP)+,R0                 ;RESTORE VOLATILE REGISTERS
                         1B   DD  0247   395          PUSHL   #ESCAPE                  ;BUILD INTERNAL FILE INDEX STRING
              02 AE    02 A8   B0  0249   396          MOVW    FAB$W_IFI(R8),2(SP)      ;
                                   024E   397
                         02   DD  024E   398          PUSHL   #LNM$M_CONFINE           ;SET CONFINE NAME ATTRIBUTE
                  00000200   8F   DD  0250   399          PUSHL   #LNM$M_TERMINAL          ;SET TERMINAL TRANSLATION ATTRIBUTE
                         0C   E1  0256   400          BBC     #NAM$V_CNCL_DEV,-        ;IS DEVICE CONCEALED?
                     . 34 A1        0258   401                  NAM$L_FNB(R7),110$       ;
                                   025B   402          SETBIT  LNM$V_CONCEALED,(SP)     ;SET CONCEALED TRANSLATION ATTRIBUTE
                         02   DD  025F   403 110$:    PUSHL   #PSL$C_SUPER             ;SET ACCESS MODE
                                   0261   404
                         7E   7C  0261   405          CLRQ    -(SP)                    ;TERMINATE THE ITEM LIST
                      14 AE   9F  0263   406          PUSHAB  20(SP)                   ;BUILD EQUIVALENCE NAME ITEM
                      04 A0   9F  0266   407          PUSHAB  4(R0)                    ;SET LENGTH
                02 AE    02   B0  0269   408          MOVW    #LNM$_STRING,2(SP)       ;SET ITEM CODE
                         7E   D4  026D   409          CLRL    -(SP)                    ;BUILD TRANSLATION ATTRIBUTES ITEM
                      18 AE   9F  026F   410          PUSHAB  24(SP)                   ;SET ATTRIBUTES ADDRESS
                00030004   8F   DD  0272   411          PUSHL   #LNM$_ATTRIBUTES@16+4    ;SET ATTRIBUTES CODE AND LENGTH
                                   0278   412
              51    FD98   CF   9E  0278   413          MOVAB   LNM$PROCESS,R1           ;BUILD TABLE NAME DESCRIPTOR
                    50   81   9A  027D   414          MOVZBL  (R1)+,R0                 ;
                    7E   50   7D  0280   415          MOVQ    R0,-(SP)                 ;
                    7E   56   7D  0283   416          MOVQ    R6,-(SP)                 ;BUILD LOGICAL NAME DESCRIPTOR
                                   0286   417
                    51   5E   D0  0286   418          MOVL    SP,R1                    ;SAVE ADDRESS OF LOGICAL NAME DESCRIPTOR
                                   0289   419          $CRELNM_S  LOGNAM=(R1),-         ;CREATE THE LOGICAL NAME
                                   0289   420                     TABNAM=8(R1),-        ;
                                   0289   421                     ITMLST=16(R1),-       ;
                                   0289   422                     ACMODE=44(R1),-       ;
                                   0289   423                     ATTR=52(R1)           ;
5E      0000004C   8F   C0  029E   424          ADDL    #20+<14*4>,SP            ;RESTORE THE STACK
                  03 50   E8  02A5   425          BLBS    R0,120$                  ;IF LBC CREATION FAILURE
                    FF5E   31  02A8   426          BRW     80$                      ;
                                   02AB   427
    1C A9    02 A8   3C  02AB   428 120$:    MOVZWL  FAB$W_IFI(R8),RAB$L_CTX+4(R9) ;RAB$L_CTX = FAB$W_IFI
    1E A9    1D A8   90  02B0   429          MOVB    FAB$B_ORG(R8),RAB$L_CTX+2+4(R9) ;RAB$L_CTX+2 = FAB$B_ORG
       69    70 AB   D0  02B5   430          MOVL    PRC_L_PPFLIST(R11),(R9)  ;INSERT NEW FILE DESCRIPTOR INTO LIST
       70 AB    59   D0  02B9   431          MOVL    R9,PRC_L_PPFLIST(R11)    ;
                     05  02BD   432          RSB                              ;
```

```
                     02BE    434          .SBTTL  READ FILE
                     02BE    435 ;+
                     02BE    436 ; DCL$READ - READ FILE
                     02BE    437 ;
                     02BE    438 ; THIS ROUTINE IS CALLED AS AN INTERNAL COMMAND TO EXECUTE THE READ FILE DCLS
                     02BE    439 ; COMMAND.
                     02BE    440 ;
                     02BE    441 ; INPUTS:
                     02BE    442 ;
                     02BE    443 ;       R8 = ADDRESS OF SCRATCH BUFFER DESCRIPTOR.
                     02BE    444 ;       R9 = ADDRESS OF SCRATCH STACK.
                     02BE    445 ;       R10 = BASE ADDRESS OF COMMAND WORK AREA.
                     02BE    446 ;       R11 = BASE ADDRESS OF PROCESS WORK AREA.
                     02BE    447 ;
                     02BE    448 ; OUTPUTS:
                     02BE    449 ;
                     02BE    450 ;       A RECORD IS READ FROM THE SPECIFIED FILE AND STORED IN THE SPECIFIED
                     02BE    451 ;       STRING VARIABLE.
                     02BE    452 ;-
                     02BE    453
           00000800  02BE    454 MAXRECSIZE      = 2048                  ;MAXIMUM LENGTH OF INPUT RECORD
                     02BE    455
                     02BE    456 DCL$READ::                              ;READ FILE
                     02BE    457 ;
                     02BE    458 ; ***** WARNING, GETID "RETURNS" VIA CO-ROUTINE CALL, NOTHING MAY BE ON THE STACK
                     02BE    459 ;
              03A2   30 02BE    460          BSBW    GETID               ;GET FILE IDENTIFICATION
            01 50    E8 02C1    461          BLBS    R0,5$               ;BRANCH IF FILE SPECIFICATION IS OK
                     05 02C4    462          RSB                         ;EXIT WITH ERROR
              FD38'  30 02C5    463 5$:      BSBW    DCL$GETDVAL         ;GET DESCRIPTOR OF SYMBOL NAME
           53   51   7D 02C8    464          MOVQ    R1,R3               ;SAVE SYMBOL NAME FOR POSSIBLE ERROR
                     02CB    465          STATUS  SYMLNG              ;ASSUME SYMBOL IS TOO LONG
000000FF   8F   51   D1 02D2    466          CMPL    R1,#255             ;SYMBOL NAME TOO LARGE?
                09   1A 02D9    467          BGTRU   10$                 ;IF GTRU YES
           56   51   7D 02DB    468          MOVQ    R1,R6               ;SAVE SYMBOL NAME DESCRIPTOR
              FD1F'  30 02DE    469          BSBW    DCL$CHKALPHA        ;CHECK IF FIRST CHARACTER IS LEGAL
            03 50    E8 02E1    470          BLBS    R0,20$              ;IF LBS THEN YES
              029C   31 02E4    471 10$:     BRW     DSPXIT              ;
                     02E7    472
                     02E7    473 ;
                     02E7    474 ; OBTAIN VALUES OF /KEY, /MATCH, /INDEX, /PROMPT, /TIME_OUT
                     02E7    475 ; AND PRESENCE OF /DELETE, /[NO]LOCK
                     02E7    476 ;
                     02E7    477 20$:     ASSUME  RAB$C_SEQ EQ 0
            04 B8    B4 02E7    478          CLRW    @4(R8)              ;ASSUME /NOTIME_OUT
            1E A9    94 02EA    479          CLRB    RAB$B_RAC(R9)       ;ASSUME /KEY ABSENT (SEQUENTIAL READ)
            34 A9    94 02ED    480          CLRB    RAB$B_PSZ(R9)       ;NO PROMPT STRING SPECIFIED YET
                     02F0    481          SETBIT  RAB$V_PMT,RAB$L_ROP(R9) ;ASSUME PROMPTING ENABLED, THOUGH
                     02F5    482          CLRBIT  RAB$V_NLK,RAB$L_ROP(R9) ;ASSUME /LOCK
                     02FA    483          CLRBIT  RAB$V_RRL,RAB$L_ROP(R9) ;ASSUME RECORD LOCK CHECKING ON READ
                     02FE    484          CLRBIT  RAB$V_KGE,RAB$L_ROP(R9) ;ASSUME /MATCH=EQ
                     0303    485          CLRBIT  RAB$V_KGT,RAB$L_ROP(R9) ;
                     0308    486          CLRBIT  RAB$V_ETO,RAB$L_ROP(R9) ;DO NOT USE THE TERMINAL XAB
                7E   D4 030D    487          CLRL    -(SP)               ;ASSUME /DELETE ABSENT (NO DELETION)
 BA AA      F9B6 CA   9E 030F    488          MOVAB   WRK_G_RESULT(R10),WRK_L_RSLNXT(R10) ;RELOAD TOKEN POINTER
              FCE8'  30 0315    489 30$:     BSBW    DCL$GETDVAL         ;GET NEXT DESCRIPTOR VALUES
           55   04   D1 0318    490          CMPL    #PTR_K_ENDLINE,R5   ;END OF LINE?
```

```
                03   12  031B   491          BNEQ    31$                     ;SKIP IF MORE TO PARSE
              00F8   31  031D   492          BRW     60$                     ;DONE WITH SCAN
           55   00   D1  0320   493 31$:     CMPL    #PTR_K_COMDQUAL,R5      ;COMMAND QUALIFIER?
                F0   12  0323   494          BNEQ    30$                     ;IF NEQ NO
              FCD8'  30  0325   495          BSBW    DCL$GETNVAL             ;GET QUALIFIER NUMBER
     51    00'8F   91  0328   496          CMPB    #CLI$K_READ_KEY,R1      ;/KEY QUALIFIER?
                3B   13  032C   497          BEQL    35$                     ;BRANCH IF YES
     51    00'8F   91  032E   498          CMPB    #CLI$K_READ_MATC,R1     ;/MATCH QUALIFIER?
                46   13  0332   499          BEQL    33$                     ;BRANCH IF YES
     51    00'8F   91  0334   500          CMPB    #CLI$K_READ_INDE,R1     ;/INDEX QUALIFER?
                6F   13  0338   501          BEQL    40$                     ;BRANCH IF YES
     51    00'8F   91  033A   502          CMPB    #CLI$K_READ_PROM,R1     ;/PROMPT QUALIFIER?
                54   13  033E   503          BEQL    42$                     ;BRANCH IF YES
     51    00'8F   91  0340   504          CMPB    #CLI$K_READ_LOCK,R1     ;/LOCK QUALIFIER?
                14   13  0344   505          BEQL    45$                     ;BRANCH IF YES
     51    00'8F   91  0346   506          CMPB    #CLI$K_READ_TIME,R1     ;/TIME_OUT QUALIFIER?
                0B   13  034A   507          BEQL    44$                     ;BRANCH IF YES
     51    00'8F   91  034C   508          CMPB    #CLI$K_READ_DELE,R1     ;/DELETE QUALIFIER?
                C3   12  0350   509          BNEQ    30$                     ;BRANCH IF NOT
        6E   01   D0  0352   510          MOVL    #1,(SP)                 ;MARK $DELETE TO BE DONE AFTER $GET
                BE   11  0355   511          BRB     30$
              0085   31  0357   512 44$:     BRW     50$                     ;PROCESS /TIME_OUT QUALIFIER
                     035A   513
     B7 53    00   E1  035A   514 45$:     BBC     #PTR_V_NEGATE-PTR_V_FLAGS,R3,30$ ;BRANCH IF /LOCK SPECIFIED (DEFAULT
                     035E   515          SETBIT  RAB$V_NLK,RAB$L_ROP(R9) ;INHIBIT AUTOMATIC LOCKING ON GET
                     0363   516          SETBIT  RAB$V_RRL,RAB$L_ROP(R9) ;INHIBIT RECORD LOCK CHECKING AS WELL
                AC   11  0367   517          BRB     30$
                     0369   518
              FC94'  30  0369   519 35$:     BSBW    DCL$GETDVAL             ;GET REQUIRED VALUE ON /KEY
     34 A9      51   90  036C   520          MOVB    R1,RAB$B_KSZ(R9)        ;ASSUME KEY IS A STRING
     30 A9      52   D0  0370   521          MOVL    R2,RAB$L_KBF(R9)
     1E A9      01   90  0374   522          MOVB    #RAB$C_KEY,RAB$B_RAC(R9) ;TELL RMS TO READ BY KEY
                9B   11  0378   523          BRB     30$
                     037A   524
              FC83'  30  037A   525 33$:     BSBW    DCL$GETDVAL             ;GET REQUIRED VALUE ON /MATCH
   62    4547 8F   B1  037D   526          CMPW    #^A/GE/,(R2)            ;IS IT GE?
                91   1F  0382   527          BLSSU   30$                     ;NO, EQ (DEFAULT)
                07   1A  0384   528          BGTRU   34$                     ;NO, GT
                     0386   529          SETBIT  RAB$V_KGE,RAB$L_ROP(R9) ;SET GE MATCH OPTION
                88   11  038B   530          BRB     30$                     ;
                     038D   531 34$:     SETBIT  RAB$V_KGT,RAB$L_ROP(R9) ;SET GT MATCH OPTION
                81   11  0392   532          BRB     30$
                     0394   533
              FC69'  30  0394   534 42$:     BSBW    DCL$GETDVAL             ;GET REQUIRED VALUE ON /PROMPT
     30 A9      52   D0  0397   535          MOVL    R2,RAB$L_PBF(R9)        ;SET PROMPT ADDRESS
     34 A9      51   90  039B   536          MOVB    R1,RAB$B_PSZ(R9)        ;SET PROMPT SIZE
                05   12  039F   537          BNEQ    43$                     ;BRANCH IF NON-NULL
                     03A1   538          CLRBIT  RAB$V_PMT,RAB$L_ROP(R9) ;MARK NULL PROMPT BY TURNING IT OFF
              FF6C   31  03A6   539 43$:     BRW     30$
                     03A9   540
              FC54'  30  03A9   541 40$:     BSBW    DCL$GETDVAL             ;GET REQUIRED VALUE ON /INDEX
        54    51   7D  03AC   542          MOVQ    R1,R4                   ;SAVE DESCRIPTOR OVER CALL
        52    51   7D  03AF   543          MOVQ    R1,R2                   ;SET DESCRIPTOR OF STRING TO CONVERT
        51    01   9A  03B2   544          MOVZBL  #PRC_K_DEC,R1           ;SET RADIX FOR CONVERSION
              FC48'  30  03B5   545          BSBW    DCL$CNVASCBIN           ;CONVERT DECIMAL ASCII TO BINARY
                07   12  03B8   546          BNEQ    32$                     ;IF ERROR, REPORT INVALID VALUE
     35 A9      51   90  03BA   547          MOVB    R1,RAB$B_KRF(R9)        ;STORE ISAM INDEX NUMBER
```

```
                FF54   31   03BE    548          BRW     30$
                       03C1    549 32$:     STATUS  IVCHAR                  ;INVALID CHARACTER
             53   54   7D   03C8    550 49$:     MOVQ    R4,R3               ;SET DESCRIPTOR OF ERROR TEXT
             5E   04   C0   03CB    551          ADDL    #4,SP               ;POP /DELETE FLAG OFF STACK
                       03CE    552 ;
                       03CE    553 ; RESTORE RAB TO ORIGINAL STATE
                       03CE    554 ;
                       03CE    555          SETBIT  RAB$V_ETO,RAB$L_ROP(R9) ;USE THE TERMINAL XAB IN THE FUTURE
                       03D3    556          CLRBIT  RAB$V_NLK,RAB$L_ROP(R9) ;TURN OFF RECORD LOCKING
                       03D8    557          CLRBIT  RAB$V_RRL,RAB$L_ROP(R9) ;    AND RECORD LOCK CHECKING
                01A4   31   03DC    558          BRW     DSPXIT              ;REPORT ERROR AND EXIT
                       03DF    559
              04 B8    B4   03DF    560 50$:     CLRW    @4(R8)              ;ASSUME /NOTIME_OUT
          26 53   00    E0   03E2    561          BBS     #PTR_V_NEGATE-PTR_V_FLAGS,R3,52$  ;SKIP IF /NOTIME_OUT
                FC17'  30   03E6    562          BSBW    DCL$GETDVAL         ;GET TIMEOUT VALUE
             54   51   7D   03E9    563          MOVQ    R1,R4               ;SAVE DESCRIPTOR FOR POSSIBLE ERROR
             52   51   7D   03EC    564          MOVQ    R1,R2               ;SET DESCRIPTOR OF STRING TO CONVERT
             51   01   D0   03EF    565          MOVL    #PRC_K_DEC,R1       ;DO ASCII DECIMAL TO BINARY CONVERSION
                FC0B'  30   03F2    566          BSBW    DCL$CNVNOEDIT       ;CONVERT THE STRING
                  51   D5   03F5    567          TSTL    R1                  ;WAS VALUE NEGATIVE?
                  16   19   03F7    568          BLSS    55$                 ;YES, REPORT ERROR
       000000FF 8F    51   D1   03F9    569          CMPL    R1,#255             ;VALUE > 255?
                  0D   14   0400    570          BGTR    55$                 ;YES, REPORT ERROR
           04 B8    51   90   0402    571          MOVB    R1,@4(R8)           ;SAVE TIMEOUT COUNT
    04 B8    4000 8F    A8   0406    572          BISW    #^X4000,@4(R8)      ;INDICATE /TIME_OUTPUT PRESENT
                FF06   31   040C    573 52$:     BRW     30$                 ;CHECK NEXT TOKEN
                       040F    574
                       040F    575 55$:     STATUS  INVRANGE                ;SET INVALID RANGE ERROR STATUS
                  B0   11   0416    576          BRB     49$                 ;REPORT ERROR
                       0418    577
                       0418    578
                       0418    579 ;
                       0418    580 ; READ RECORD
                       0418    581 ;
                       0418    582
                       0418    583 60$:     CLRBIT  RAB$V_TMO,RAB$L_ROP(R9) ;ASSUME /NOTIME_OUT
    04 B8    4000 8F    B3   041D    584          BITW    #^X4000,@4(R8)      ;WAS A TIMEOUT VALUE SPECIFIED
                  0A   13   0423    585          BEQL    62$                 ;NO, SKIP TIMEOUT SETUP
                       0425    586          SETBIT  RAB$V_TMO,RAB$L_ROP(R9) ;SET TIMEOUT VALUE PRESENT FLAG
      1F A9    04 B8    90   042A    587          MOVB    @4(R8),RAB$B_TMO(R9)    ;SET TIMEOUT VALUE
                       042F    588
         5E    F800 CE    9E   042F    589 62$:     MOVAB   -MAXRECSIZE(SP),SP      ;ALLOCATE INPUT BUFFER ON STACK
               58   5E    D0   0434    590          MOVL    SP,R8               ;GET ADDRESS OF INPUT BUFFER
    20 A9    0800 8F    B0   0437    591          MOVW    #MAXRECSIZE,RAB$W_USZ(R9) ;SET SIZE OF RECORD BUFFER
       24 A9    58   D0   043D    592          MOVL    R8,RAB$L_UBF(R9)        ;SET ADDRESS OF RECORD BUFFER
                       0441    593          ;&&& This is a bug.  RAB$L_CTX is set to FAB$L_DEV for special PPF
                       0441    594          ;&&& files like SYS$COMMAND, etc.
          20    1A A9    91   0441    595          CMPB    RAB$L_CTX+2(R9),#FAB$C_IDX ;ORG = ISAM?
                  11   13   0445    596          BEQL    65$                 ;IF SO, SKIP SETTING OF PROMPT STRING
               34 A9    95   0447    597          TSTB    RAB$B_PSZ(R9)       ;PROMPT STRING EXPLICITLY SPECIFIED?
                  0C   12   044A    598          BNEQ    65$                 ;IF NOT, SET THE FOLLOWING DEFAULT
    34 A9    FBB0 CF    90   044C    599          MOVB    DATAP,RAB$B_PSZ(R9)     ;SET SIZE OF PROMPT STRING
    30 A9    FBAB CF    9E   0452    600          MOVAB   DATAP+1,RAB$L_PBF(R9)   ;SET ADDRESS OF PROMPT STRING
       68 AB    04   A8   0458    601 65$:     BISW    #PRC_M_DISABL,PRC_W_FLAGS(R11) ;DISABLE CONTROL Y/C AST'S
                       045C    602          $GET    RAB=(R9)                ;READ NEXT RECORD FROM FILE
                       0465    603          CLRBIT  RAB$V_TMO,RAB$L_ROP(R9) ;TURN TIMEOUT PROCESSING OFF
                  2C 50   E9   046A    604          BLBC    R0,80$              ;IF LBC I/O ERROR
```

```
                51    22 A9    3C   046D    605          MOVZWL  RAB$W_RSZ(R9),R1        ;GET SIZE OF RECORD READ
                   52    58    D0   0471    606          MOVL    R8,R2                   ;SET ADDRESS OF RECORD READ
                   53    56    7D   0474    607          MOVQ    R6,R3                   ;SET DESCRIPTOR OF SYMBOL NAME
                55    38 AB    9E   0477    608          MOVAB   PRC_Q_LOCAL(R11),R5     ;SET ADDRESS OF SYMBOL TABLE LISTHEAD
                   50    00    D0   047B    609          MOVL    #SYM_K_STRING,R0        ;SET TYPE TO STRING
                      FB7F'    30   047E    610          BSBW    DCL$ALLOCSYM            ;ALLOCATE AND INSERT SYMBOL IN TABLE
                     15 50     E9   0481    611          BLBC    R0,80$                  ;IF LBC ALLOCATION FAILURE
                               0484    612          STATUS  NORMAL                  ;SET NORMAL COMPLETION STATUS
                09 0800 CE     E9   048B    613          BLBC    MAXRECSIZE(SP),80$      ;BRANCH IF /DELETE NOT SPECIFIED
                               0490    614          $DELETE RAB=(R9)                ;DELETE CURRENT RECORD JUST READ
             5E   0804 CE     9E   0499    615 80$:     MOVAB   MAXRECSIZE+4(SP),SP     ;DEALLOCATE INPUT BUFFER
                               049E    616 ;
                               049E    617 ; RESTORE RAB TO ORIGINAL STATE - SO THAT IN CASE WE ARE DEALING WITH A
                               049E    618 ; STANDARD PPF RAB, SUCH AS SYS$INPUT, NORMAL USE OF THE RAB FOR PROCEDURE
                               049E    619 ; INPUT IS NOT MESSED UP.
                               049E    620 ;
                               049E    621          SETBIT  RAB$V_ETO,RAB$L_ROP(R9) ;USE THE TERMINAL XAB IN THE FUTURE
                               04A3    622          CLRBIT  RAB$V_NLK,RAB$L_ROP(R9) ;TURN OFF RECORD LOCKING
                               04A8    623          CLRBIT  RAB$V_RRL,RAB$L_ROP(R9) ;   AND RECORD LOCK CHECKING
                            05 04AC    624          RSB                             ;
```

```
                        04AD   626          .SBTTL  WRITE FILE
                        04AD   627 ;+
                        04AD   628 ; DCL$WRITE - WRITE FILE
                        04AD   629 ;
                        04AD   630 ; THIS ROUTINE IS CALLED AS AN INTERNAL COMMAND TO EXECUTE THE WRITE FILE DCLS
                        04AD   631 ; COMMAND.
                        04AD   632 ;
                        04AD   633 ; INPUTS:
                        04AD   634 ;
                        04AD   635 ;       R8 = ADDRESS OF SCRATCH BUFFER DESCRIPTOR.
                        04AD   636 ;       R9 = ADDRESS OF SCRATCH STACK.
                        04AD   637 ;       R10 = BASE ADDRESS OF COMMAND WORK AREA.
                        04AD   638 ;       R11 = BASE ADDRESS OF PROCESS WORK AREA.
                        04AD   639 ;
                        04AD   640 ; OUTPUTS:
                        04AD   641 ;
                        04AD   642 ;       THE SPECIFIED LIST OF STRING VARIABLES IS COLLECTED TOGETHER INTO A
                        04AD   643 ;       SINGLE RECORD AND WRITTEN TO THE SPECIFIED FILE.
                        04AD   644 ;-
                        04AD   645
                        04AD   646 DCL$WRITE::                              ;WRITE FILE
                        04AD   647 ;
                        04AD   648 ; ***** WARNING, GETID "RETURNS" VIA CO-ROUTINE CALL, NOTHING MAY BE ON THE STACK
                        04AD   649 ;
                01B3    30 04AD   650          BSBW    GETID                ;GET FILE IDENTIFICATION
              01 50     E8 04B0   651          BLBS    R0,1$                ;BRANCH IF OK
                        05 04B3   652          RSB                          ;RETURN WITH ERROR
                        04B4   653 ;
                        04B4   654 ;       CHECK FOR PRESENCE OF /SYMBOL OR /UPDATE
                        04B4   655 ;
                   56   D4 04B4   656 1$:    CLRL    R6                   ;ASSUME /UPDATE ABSENT (NO $UPDATE)
          BA AA F9B6 CA   9E 04B6   657          MOVAB   WRK_G_RESULT(R10),WRK_L_RSLNXT(R10) ;RELOAD TOKEN POINTER
                FB41'     30 04BC   658 2$:    BSBW    DCL$GETDVAL          ;GET NEXT DESCRIPTOR VALUES
                55   03   D1 04BF   659          CMPL    #PTR_K_PARAMETR,R5   ;END OF LINE?
                     1E   13 04C2   660          BEQL    5$                   ;BRANCH IF DONE WITH SCAN
                55   00   D1 04C4   661          CMPL    #PTR_K_COMDQUAL,R5   ;COMMAND QUALIFIER?
                     F3   12 04C7   662          BNEQ    2$                   ;IF NEQ NO
                FB34'     30 04C9   663          BSBW    DCL$GETNVAL          ;GET QUALIFIER NUMBER
             51   00'8F   91 04CC   664          CMPB    #CLI$K_WRIT_UPDA,R1  ;/UPDATE QUALIFIER?
                     05   12 04D0   665          BNEQ    3$                   ;BRANCH IF NOT
                56   02   88 04D2   666          BISB    #2,R6                ;MARK $UPDATE TO BE DONE, NOT $PUT
                     E5   11 04D5   667          BRB     2$
             51   00'8F   91 04D7   668 3$:    CMPB    #CLI$K_WRIT_SYMB,R1  ;/SYMBOL QUALIFIER?
                     DF   12 04DB   669          BNEQ    2$                   ;IF NEQ, NO
                56   01   88 04DD   670          BISB    #1,R6                ;MARK /SYMBOL SEEN FLAG
                     DA   11 04E0   671          BRB     2$
                        04E2   672 ;
                        04E2   673 ;       GET SYMBOLS OR EXPRESSIONS AND CONCATENATE THEM IN THE OUTPUT BUFFER
                        04E2   674 ;
                7E   56   D0 04E2   675 5$:    MOVL    R6,-(SP)             ;STORE /UPDATE AND /SYMBOL FLAG
                58   5E   D0 04E5   676          MOVL    SP,R8                ;SET ENDING ADDRESS OF OUTPUT BUFFER
             5E F800 CE   9E 04E8   677          MOVAB   -MAXRECSIZE(SP),SP   ;ALLOCATE THE OUTPUT BUFFER ON THE STACK
               28 A9 5E   D0 04ED   678          MOVL    SP,RAB$L_RBF(R9)     ;SET ADDRESS OF OUTPUT BUFFER
                57   5E   D0 04F1   679          MOVL    SP,R7                ;
                FB09'     30 04F4   680 10$:    BSBW    DCL$GETDVAL          ;GET DESCRIPTOR VALUES
                56   54   D0 04F7   681          MOVL    R4,R6                ;SAVE TERMINATOR CLASS NUMBER
                        04FA   682          STATUS  BUFOVF                   ;ASSUME OUTPUT BUFFER WILL OVERFLOW
```

```
                     0501    683          MOVQ    R1,R3                          ;SAVE VALUE DESCRIPT. FOR POSSIB. ERROR
   25 68    E9       0504    684          BLBC    (R8),30$                       ;IF CLEAR, VALUE IS AN EXPRESSION
                     0507    685
   56 04    D1       0507    686          CMPL    #PTR_K_PLUS,R6                 ;IS TERMINATOR A '+'
      0B    12       050A    687          BNEQ    20$                            ;BRANCH IF NOT
      53    D6       050C    688          INCL    R3                             ;INCLUDE '+' IN ERROR MESSAGE
                     050E    689          STATUS  NOCCAT                         ;SET CONCAT. NOT ALLOWED ERROR MSG.
      67    11       0515    690          BRB     90$                            ;REPORT ERROR
                     0517    691
   62 22    91       0517    692 20$:     CMPB    #^A/''/,(R2)                   ;IS THIS A QUOTED STRING?
      05    12       051A    693          BNEQ    25$                            ;NO, IT MUST BE A SYMBOL
    FAE1'   30       051C    694          BSBW    DCL$COMPSTRING                 ;YES, COMPRESS STRING BEFORE COPYING
      0B    11       051F    695          BRB     30$                            ;COPY STRING
                     0521    696
      19    BB       0521    697 25$:     PUSHR   #^M<R0,R3,R4>                  ;SAVE SYMBOL DESCRIPTOR AND STATUS
    FADA'   30       0523    698          BSBW    DCL$SYM_STRING                 ;EVALUATE SYMBOL
      19    BA       0526    699          POPR    #^M<R0,R3,R4>                  ;RESTORE SYMBOL DESCRIPTOR AND STATUS
      51    D5       0528    700          TSTL    R1                             ;SYMBOL DEFINED?
      4B    13       052A    701          BEQL    60$                            ;NO, REPORT SYMBOL NOT FOUND
                     052C    702
      51    D7       052C    703 30$:     DECL    R1                             ;ANY MORE CHARACTERS TO MOVE?
      0A    19       052E    704          BLSS    40$                            ;IF LSS NO
   58 57    D1       0530    705          CMPL    R7,R8                          ;ANY ROOM IN OUTPUT BUFFER?
      49    13       0533    706          BEQL    90$                            ;IF EQL NO
   87 82    90       0535    707          MOVB    (R2)+,(R7)+                    ;MOVE CHARACTER TO OUTPUT BUFFER
      F2    11       0538    708          BRB     30$                            ;
   56 05    D1       053A    709 40$:     CMPL    #PTR_K_COMMA,R6                ;ANY MORE PARAMETERS TO COLLECT?
      B5    13       053D    710          BEQL    10$                            ;IF EQL YES
57    28 A9 C2       053F    711          SUBL    RAB$L_RBF(R9),R7               ;CALCULATE LENGTH OF OUTPUT RECORD
22 A9    57 B0       0543    712          MOVW    R7,RAB$W_RSZ(R9)               ;SET LENGTH OF OUTPUT RECORD
                     0547    713
68 AB    04 A8       0547    714          BISW    #PRC_M_DISABL,PRC_W_FLAGS(R11) ;DISABLE CONTROL Y/C AST'S
 0F 68   01 E1       054B    715          BBC     #1,(R8),78$                    ;BRANCH IF NOT /UPDATE
                     054F    716 ;
                     054F    717 ; UPDATE RECORD
                     054F    718 ;
                     054F    719          $UPDATE RAB=(R9)                       ;UPDATE RECORD
5E    0804 CE  9E    0558    720          MOVAB   MAXRECSIZE+4(SP),SP            ;RESTORE THE STACK PTR
                05   055D    721          RSB
                     055E    722 ;
                     055E    723 ; OUTPUT RECORD
                     055E    724 ;
                     055E    725 78$:     ;&&& This is a bug.  RAB$L_CTX is set to FAB$L_DEV for special PPF
                     055E    726          ;&&& files like SYS$COMMAND, etc.
  20   1A A9  91     055E    727          CMPB    RAB$L_CTX+2(R9),#FAB$C_IDX     ;ORG = ISAM?
         04  12     0562    728          BNEQ    45$                            ;BRANCH IF NOT
  1E A9  01  90     0564    729          MOVB    #RAB$C_KEY,RAB$B_RAC(R9)       ;IF ISAM, ALWAYS INSERT IN KEY ORDER
                     0568    730 45$:     $PUT    RAB=(R9)                       ;OUTPUT RECORD
5E    0804 CE  9E    0571    731          MOVAB   MAXRECSIZE+4(SP),SP            ;RESTORE THE STACK PTR
                05   0576    732          RSB                                    ;
                     0577    733
                     0577    734 ;
                     0577    735 ; UNDEFINED SYMBOL
                     0577    736 ;
                     0577    737
                     0577    738 60$:     STATUS  UNDSYM                         ;SET UNDEFINED SYMBOL STATUS
                     057E    739
```

```
                5E    0804 CE    9E  057E    740 90$:    MOVAB   MAXRECSIZE+4(SP),SP     ;RESTORE THE STACK PTR
                                     0583    741
                                     0583    742 ;
                                     0583    743 ; CALCULATE ADDRESS OF DISPLAY SEGMENT AND CLEAR COMMAND IN EXECUTION
                                     0583    744 ;
                                     0583    745
                F48A CA    54    D0  0583    746 DSPXIT: MOVL    R4,WRK_L_MARKPTR(R10)   ;SET ADDRESS OF DISPLAY SEGMENT
              F486 CA    54    53    C1  0588    747         ADDL3   R3,R4,WRK_L_EXPANDPTR(R10) ;CALCULATE ENDING ADDRESS OF DISPLAY SEGM
                                     058E    748         CLRBIT  WRK_V_COMMAND,WRK_W_FLAGS(R10) ;DISPLAY ERROR TEXT WITH MESSAGE
                                  05 0592    749         RSB                             ;
```

```
                     0593   751          .SBTTL  DCL$OPEN_CREATE - OPEN/CREATE FILE
                     0593   752 ;+
                     0593   753 ; DCL$OPEN_CREATE - OPEN/CREATE FILE
                     0593   754 ;
                     0593   755 ; THESE ROUTINES OPEN AN EXISTING FILE OR CREATE A NEW OUTPUT FILE
                     0593   756 ; GIVEN A FAB WHICH IS ALREADY SET UP TO CALL THE $OPEN OR $CREATE
                     0593   757 ; RMS SERVICES DIRECTLY.  IN ADDITION TO PERFORMING THE SPECIFIED
                     0593   758 ; RMS SERVICE THIS CODE PRODUCES GOOD ERROR MESSAGES.
                     0593   759 ;
                     0593   760 ; INPUTS:
                     0593   761 ;
                     0593   762 ;       R0 = FAB ADDRESS
                     0593   763 ;       R1 = FLAGS
                     0593   764 ;            BIT 0 = 0 FOR OPEN
                     0593   765 ;                  = 1 FOR CREATE
                     0593   766 ;            BIT 1 = 0 FOR NORMAL ERROR REPORTS IF ERROR
                     0593   767 ;                  = 1 IF NOT SUPPOSED TO ISSUE ERROR REPORT
                     0593   768 ;            BIT 2 = 0 FOR NORMAL RESULT FILE SPEC PROCESSING
                     0593   769 ;                    1 IF RESULT STRING BUFFER IS ALREADY SPECIFIED
                     0593   770 ;            BIT 3 = USED LOCALLY BY DCL$OPEN
                     0593   771 ;
                     0593   772 ;
                     0593   773 ; IMPLICIT INPUTS:
                     0593   774 ;
                     0593   775 ;       FAB$L_FNA AND FAB$B_FNS DESCRIBE THE FILE NAME STRING
                     0593   776 ;       FAB$L_NAM POINTS TO A NAME BLOCK
                     0593   777 ;       THE NAME BLOCK IN TURN HAS NO RESULT STRING OR EXPANDED STRING
                     0593   778 ;       ADDRESS SET UP
                     0593   779 ;
                     0593   780 ; OUTPUTS:
                     0593   781 ;
                     0593   782 ;       R0 = STATUS - IF ERROR, INIHIBIT MSG BIT IS SET
                     0593   783 ;       R1 ALTERED
                     0593   784 ;       ALL OTHER REGISTERS PRESERVED
                     0593   785 ;
                     0593   786 ;-
                     0593   787 DCL$OPEN_CREATE::
            1C   BB  0593   788          PUSHR   #^M<R2,R3,R4>                   ;SAVE SOME REGISTERS
        52  50   7D  0595   789          MOVQ    R0,R2                           ;R2=FAB ADR
                     0598   790                                                  ;R3=OPEN/CREATE, ERR REPORT FLAGS
      54  28 A2   D0  0598   791          MOVL    FAB$L_NAM(R2),R4                ;NAME BLOCK ADDRESS
      17 53   02   E0  059C   792          BBS     #2,R3,10$                       ;BRANCH IF ESA/RSA ALREADY SPECIFIED
   5E   FF00 CE   DE  05A0   793          MOVAL   -<<NAM$C_MAXRSS+3>&^C<3>>(SP),SP ;RESERVE NAME STRING BUFFER
      04 A4   5E   D0  05A5   794          MOVL    SP,NAM$L_RSA(R4)                ;SET RESULT NAME STRING ADDRESS
      0C A4   5E   D0  05A9   795          MOVL    SP,NAM$L_ESA(R4)                ;SET EXPANDED NAME STRING ADDRESS
                     05AD   796          ASSUME  NAM$B_RSS+1 EQ NAM$B_RSL
02 A4    FF 8F   9B  05AD   797          MOVZBW  #NAM$C_MAXRSS,NAM$B_RSS(R4) ;RESULT STRING BUFFER SIZE
                     05B2   798                                                  ;ZERO RESULT STRING LENGTH
                     05B2   799          ASSUME  NAM$B_ESS+1 EQ NAM$B_ESL
0A A4    FF 8F   9B  05B2   800          MOVZBW  #NAM$C_MAXRSS,NAM$B_ESS(R4) ;EXPANDED STRING BUFFER SIZE
                     05B7   801                                                  ;ZERO EXPANDED STRING LENGTH
         0B 53   E8  05B7   802 10$:     BLBS    R3,20$                          ;BRANCH IF CREATE FUNCTION
                     05BA   803          $OPEN   (R2)                            ;OPEN THE FILE
            09   11  05C3   804          BRB     30$
                     05C5   805 20$:     $CREATE (R2)                            ;CREATE THE FILE
         4A 50   E8  05CE   806 30$:     BLBS    R0,80$                          ;BRANCH IF SUCCESSFUL
                     05D1   807 ;
```

```
                                05D1    808 ; ERROR OPENING OR CREATING THE FILE
                                05D1    809 ;
                          04 A4   DD    05D1    810         PUSHL   NAM$L_RSA(R4)            ;ADDRESS OF RESULT NAME STRING
                       7E 03 A4   9A    05D4    811         MOVZBL  NAM$B_RSL(R4),-(SP)      ;RESULT STRING SIZE
                             0F   12    05D8    812         BNEQ    50$                      ;BRANCH IF THERE WAS A RESULT STRING
                       6E 0B A4   9A    05DA    813         MOVZBL  NAM$B_ESL(R4),(SP)       ;EXPANDED NAME STRING SIZE
                             09   12    05DE    814         BNEQ    50$                      ;BRANCH IF THERE WAS AN EXPANDED NAME STRING
                    04 AE 2C A2   D0    05E0    815         MOVL    FAB$L_FNA(R2),4(SP)      ;USE ORIGINAL FILE NAME, SET ADR
                       6E 34 A2   9A    05E5    816         MOVZBL  FAB$B_FNS(R2),(SP)       ;SIZE OF ORIGINAL NAME
                                05E9    817 ;
                                05E9    818 ; NOW BUILD THE PUTMSG PARAMETER LIST
                                05E9    819 ;
                          0C A2   DD    05E9    820 50$:    PUSHL   FAB$L_STV(R2)            ;STV FOR THE RMS ERROR CODE
                             50   DD    05EC    821         PUSHL   R0                       ;RMS ERROR CODE
                          08 AE   7F    05EE    822         PUSHAQ  8(SP)                    ;ADDRESS OF FILE NAME STRING DESCRIPTOR
                             01   DD    05F1    823         PUSHL   #1                       ;FAO ARGUMENT COUNT
                       7E    03   B0    05F3    824         MOVW    #<CLI$_NORMAL & STS$M_FAC_NO>@-16,-(SP) ;PUT IN CLI FACILITY CODE
                    0002'8F          B0    05F6    825         MOVW    #<<SHR$_OPENIN & ^C<STS$M_SEVERITY>> ! -
                             7E            05FA    826                 <STS$K_ERROR @ STS$V_SEVERITY>>,-(SP) ;ASSUME FUNCTION IS OPEN
                          05 53   E9    05FB    827         BLBC    R3,60$                   ;BRANCH IF IT IS OPEN
                    0002'8F          B0    05FE    828         MOVW    #<<SHR$_OPENOUT & ^C<STS$M_SEVERITY>> ! -
                             6E            0602    829                 <STS$K_ERROR @ STS$V_SEVERITY>>,(SP) ;IT WAS A CREATE
                             05   DD    0603    830 60$:    PUSHL   #5                       ;NUMBER OF PARAMETERS TO PUTMSG
                       06 53      01   E0    0605    831         BBS     #1,R3,70$                ;BRANCH IF NOT REPORTING ERROR
                          50      5E   D0    0609    832         MOVL    SP,R0                    ;ADDRESS OF PUTMSG PARAMETER LIST
                                F9F1'  30    060C    833         BSBW    DCL$PUTMSG               ;DO THE PUTMSG CALL
50    10 AE    10000000 8F          C9    060F    834 70$:    BISL3   #STS$M_INHIB_MSG,16(SP),R0 ;RECOVER THE OPEN ERROR CODE (STS)
                                0618    835                             ;AND SET THE INHIBIT MESSAGE FLAG
                             5E   20   C0    0618    836         ADDL    #<6*4+8>,SP              ;CLEAN OFF THE PUTMSG ARG LIST
                                061B    837                             ;AND THE FILE NAME DESCRIPTOR
                       0B 53      02   E0    061B    838 80$:    BBS     #2,R3,90$                ;BRANCH IF ESA/RSA ALREADY SPECIFIED
                    5E    0100 CE      DE    061F    839         MOVAL   <<NAM$C_MAXRSS+3>&^C<3>>(SP),SP ;NAME STRING SIZE TO RESERVE
                                0624    840         ASSUME  NAM$B_RSS+1 EQ NAM$B_RSL
                          02 A4   B4    0624    841         CLRW    NAM$B_RSS(R4)            ;CLEAN UP THE NAME BLOCK
                                0627    842         ASSUME  NAM$B_ESS+1 EQ NAM$B_ESL
                          0A A4   B4    0627    843         CLRW    NAM$B_ESS(R4)            ;NAME STRINGS ARE NOT VALID
                             1C   BA    062A    844 90$:    POPR    #^M<R2,R3,R4>            ;RESTORE SAVED REGISTERS
                                  05    062C    845         RSB
```

```
                     062D   847          .SBTTL  LOCAL SUBROUTINES
                     062D   848 ;+
                     062D   849 ; CHECKPPF - CHECK FOR PROCESS PERMANENT FILE
                     062D   850 ;
                     062D   851 ; THIS ROUTINE IS CALLED TO CHECK IF AN ISI VALUE IS FOR A PROCESS PERMANENT FILE.
                     062D   852 ;
                     062D   853 ; INPUTS:
                     062D   854 ;
                     062D   855 ;       R6 = ISI VALUE.
                     062D   856 ;
                     062D   857 ; OUTPUTS:
                     062D   858 ;
                     062D   859 ;       Z = 0 IF NOT PROCESS PERMANENT FILE.
                     062D   860 ;
                     062D   861 ;       Z = 1 IF PROCESS PERMANENT FILE WITH:
                     062D   862 ;
                     062D   863 ;               R5 = ADDRESS OF ASSOCIATED RAB.
                     062D   864 ;-
                     062D   865
                     062D   866 CHECKPPF:                                    ;CHECK FOR PROCESS PERMANENT FILE
51   00A0 CB   D0    062D   867          MOVL    PRC_L_STACKPT(R11),R1      ;GET CURRENT INDIRECT STACK POINTER
55     14 AB   D0    0632   868          MOVL    PRC_L_INDINPRAB(R11),R5    ;GET ADDRESS OF CURRENT INPUT RAB
04 A1    56   B1    0636   869          CMPW    R6,IDF_W_INPIFI(R1)        ;IFI MATCH?
           26   13    063A   870          BEQL    10$                        ;IF EQL YES
55     18 AB   D0    063C   871          MOVL    PRC_L_INDOUTRAB(R11),R5    ;GET ADDRESS OF CURRENT OUTPUT RAB
20 A1    56   B1    0640   872          CMPW    R6,IDF_W_OUTIFI(R1)        ;IFI MATCH?
           1C   13    0644   873          BEQL    10$                        ;IF EQL YES
                     0646   874 4$:
           61   D5    0646   875          TSTL    IDF_L_LNK(R1)              ;AT OUTER COMMAND LEVEL?
           05   13    0648   876          BEQL    6$
     51    61   D0    064A   877          MOVL    IDF_L_LNK(R1),R1           ;POINT BACK ONE LEVEL
           F7   11    064D   878          BRB     4$
                     064F   879 6$:
55     08 AB   D0    064F   880          MOVL    PRC_L_INPRAB(R11),R5       ;GET LEVEL-0 (SYS$COMMAND) INPUT RAB
04 A1    56   B1    0653   881          CMPW    R6,IDF_W_INPIFI(R1)        ;IFI MATCH?
           09   13    0657   882          BEQL    10$                        ;IF EQL YES
55     0C AB   D0    0659   883          MOVL    PRC_L_OUTRAB(R11),R5       ;GET LEVEL-0 OUTPUT RAB
0114 CB    56   B1    065D   884          CMPW    R6,PRC_W_OUTIFI(R11)       ;IFI MATCH?
           05   0662   885 10$:    RSB                                       ;
```

```
                                0663   887 ;+
                                0663   888 ; GETID - GET FILE ID
                                0663   889 ;
                                0663   890 ; THIS ROUTINE IS CALLED TO CONVERT THE FILE IDENTIFIER TO BINARY AND SEARCH THE
                                0663   891 ; FILE DESCRIPTOR LIST FOR A MATCH.
                                0663   892 ;
                                0663   893 ; INPUTS:
                                0663   894 ;
                                0663   895 ;       NONE
                                0663   896 ;
                                0663   897 ; OUTPUTS:
                                0663   898 ;
                                0663   899 ;       R0 LOW BIT CLEAR INDICATES FAILURE TO FIND SPECIFIED FILE ID.
                                0663   900 ;
                                0663   901 ;       R0 LOW BIT SET INDICATES SUCCESSFUL FILE DESCRIPTOR LIST SEARCH WITH:
                                0663   902 ;
                                0663   903 ;               R3 = LENGTH OF LOCIAL FILE NAME.
                                0663   904 ;               R4 = ADDRESS OF LOGICAL FILE NAME.
                                0663   905 ;               R5 = ADDRESS OF FILE DESCRIPTOR BLOCK.
                                0663   906 ;               R6 = FILE IDENTIFICATION NUMBER.
                                0663   907 ;               R7 = ADDRESS OF PREVIOUS FILE DESCRIPTOR.
                                0663   908 ;               R9 = ADDRESS OF ASSOCIATED RAB.
                                0663   909 ;
                                0663   910 ;       WRK_L_RSLNXT IS SET TO THE NEXT PARAMETER AFTER THE FILE ID.
                                0663   911 ;-
                                0663   912
                                0663   913 GETID:                                        ;GET FILE ID
BA AA   F9B6 CA   9E            0663   914         MOVAB   WRK_G_RESULT(R10),WRK_L_RSLNXT(R10) ;RELOAD RESULT DESCRIPTOR POINTE
          F994'   30            0669   915 10$:     BSBW    DCL$GETDVAL                 ;GET DESCRIPTOR VALUES
        55  03    D1            066C   916         CMPL    #PTR_K_PARAMETR,R5          ;PARAMETER DESCRIPTOR?
            F8    12            066F   917         BNEQ    10$                         ;IF NEQ NO
        53  51    7D            0671   918         MOVQ    R1,R3                       ;SAVE LOGICAL FILE PARAMETERS
        7E  51    7D            0674   919         MOVQ    R1,-(SP)                    ;BUILD LOGICAL NAME DESCRIPTOR
        55  5E    D0            0677   920         MOVL    SP,R5                       ;SAVE ADDRESS OF LOGICAL NAME DESCRIPTOR
     56   FF 8F    9A            067A   921         MOVZBL  #LNM$C_NAMLENGTH,R6         ;GET MAXIMUM LENGTH OF LOGICAL NAME
        5E  56    C2            067E   922         SUBL    R6,SP                       ;ALLOCATE LOGICAL NAME TRANSLATION BUFFER
            6E    9F            0681   923         PUSHAB  (SP)                        ;BUILD TRANSLATION BUFFER DESCRIPTOR
            56    DD            0683   924         PUSHL   R6                          ;
        56  5E    D0            0685   925         MOVL    SP,R6                       ;SAVE ADDRESS OF BUFFER DESCRIPTOR
        52  0A    D0            0688   926         MOVL    #10,R2                      ;SET MAXIMUM LOOP COUNT
                                068B   927 20$:     $TRNLOG_S (R5),(R5),(R6)            ;TRANSLATE LOGICAL NAME
        50  00'   B1            069E   928         CMPW    S^#SS$_NORMAL,R0            ;NORMAL COMPLETION?
            45    12            06A1   929         BNEQ    80$                         ;IF NEQ NO
        65  04    B1            06A3   930         CMPW    #4,(R5)                     ;TRANSLATED NAME LARGE ENOUGH?
            06    1A            06A6   931         BGTRU   30$                         ;IF GTRU NO
     08 A6  1B    B1            06A8   932         CMPW    #ESCAPE,8(R6)               ;PROCESS PERMANENT FILE IFI?
            0A    13            06AC   933         BEQL    40$                         ;IF EQL YES
  04 A5   04 A6    D0            06AE   934 30$:     MOVL    4(R6),4(R5)                 ;SET FOR NEXT TRANSLATION
          D5 52    F5            06B3   935         SOBGTR  R2,20$                      ;ANY MORE TANSLATIONS LEFT?
             30    11            06B6   936         BRB     80$                         ;
     56  0A A6    B0            06B8   937 40$:     MOVW    10(R6),R6                   ;GET INTERNAL FILE INDEX
   5E    010F CE    9E            06BC   938         MOVAB   LNM$C_NAMLENGTH+8+8(SP),SP ;REMOVE LOGICAL NAMES FROM STACK
           FF69    30            06C1   939         BSBW    CHECKPPF                    ;CHECK IF PROCESS PERMANENT FILE
             16    13            06C4   940         BEQL    70$                         ;IF EQL YES
     57   70 AB    9E            06C6   941         MOVAB   PRC_L_PPFLIST(R11),R7       ;GET ADDRESS OF PREVIOUS FILE DESCRIPTOR
        55  67    D0            06CA   942 50$:     MOVL    (R7),R5                     ;GET ADDRESS OF NEXT FILE DESCRIPTOR
             1E    13            06CD   943         BEQL    85$                         ;IF EQL END OF LIST
```

```
                                                                J 5
                1C A5    56   B1  06CF   944          CMPW    R6,RAB$L_CTX+4(R5)      ;FILE IDENTIFICATION MATCH?
                   05    13   13  06D3   945          BEQL    60$                     ;IF EQL YES
                   57    55   D0  06D5   946          MOVL    R5,R7                   ;SAVE ADDRESS OF PREVIOUS ENTRY
                         F0   11  06D8   947          BRB     50$                     ;
                         85   D5  06DA   948 60$:      TSTL    (R5)+                   ;POINT TO ACTUAL RAB
                   59    55   D0  06DC   949 70$:      MOVL    R5,R9                   ;SET ADDRESS OF ASSOCIATED RAB
                               06DF   950          STATUS  NORMAL                  ;SET NORMAL COMPLETION STATUS
                         0C   11  06E6   951          BRB     90$                     ;
                              06E8   952
                              06E8   953 ;
                              06E8   954 ; FILE IDENTIFICATION NOT FOUND
                              06E8   955 ;
                              06E8   956
                5E    010F CE   9E  06E8   957 80$:      MOVAB   LNM$C_NAMLENGTH+8+8(SP),SP ;REMOVE LOGICAL NAMES FROM STACK
                              06ED   958 85$:      STATUS  UNDFIL                  ;SET UNDEFINED FILE STATUS
                              06F4   959
                              06F4   960 ;
                              06F4   961 ; CALL THE CALLER BACK AS A CO-ROUTINE SO THAT /END= AND /ERR= QUALIFIERS CAN BE
                              06F4   962 ; PROCESSED IN ONE PLACE.
                              06F4   963 ;
                              06F4   964
                         9E   16  06F4   965 90$:      JSB     @(SP)+                  ;CALL CALLER BACK
                      21 50   E8  06F6   966          BLBS    R0,110$                 ;IF LBS SUCCESSFUL COMPLETION
                   57   00'8F   9A  06F9   967          MOVZBL  #CLI$K_OPEN_END_,R7     ;ASSUME END OF FILE
             50   00000000'8F   D1  06FD   968          CMPL    #RMS$_EOF,R0            ;END OF FILE?
                         02   12  0704   969          BNEQ    100$                    ;IF NEQ NO
                         13   10  0706   970          BSBB    LABEL_CHECK             ;CHECK FOR END OF FILE LABEL
                   57   00'8F   9A  0708   971 100$:    MOVZBL  #CLI$K_OPEN_ERRO,R7     ;SET FOR ERROR
                         0D   10  070C   972          BSBB    LABEL_CHECK             ;CHECK FOR ERROR LABEL
                07 F2 AA   02   E5  070E   973          BBCC    #WRK_V_CLOSERR,WRK_W_FLAGS2(R10),110$  ;SKIP IF /LOG
                              0713   974          STATUS  NORMAL                  ;OTHER SUPPRESS ERROR MESSAGE
                         05   13  071A   975 110$:    RSB                             ;
```

```
                     071B   977 ;+
                     071B   978 ; LABEL_CHECK - CHECK FOR LABEL
                     071B   979 ;
                     071B   980 ; THIS ROUTINE IS CALLED TO SCAN THE COMMAND LEVEL QUALIFIERS FOR AN /ERROR=
                     071B   981 ; OF /END_OF_FILE= QUALIFIER.
                     071B   982 ;
                     071B   983 ; INPUTS:
                     071B   984 ;
                     071B   985 ;       R7 = TYPE OF QUALIFIER TO SCAN FOR.
                     071B   986 ;       R8 = SAVED FINAL STATUS VALUE.
                     071B   987 ;
                     071B   988 ; OUTPUTS:
                     071B   989 ;
                     071B   990 ;       THE RESULT PARSE TABLE IS SCANNED FOR A QUALIFIER TYPE MATCH. IF A MATCH
                     071B   991 ;       IS FOUND, THEN A GOTO THE SPECIFIED LABEL IS EXECUTED. ELSE A RETURN TO
                     071B   992 ;       THE CALLER IS EXECUTED.
                     071B   993 ;-
                     071B   994
                     071B   995 LABEL_CHECK:                                 ;CHECK LABEL
BA AA   F986 CA   9E 071B   996         MOVAB   WRK_G_RESULT(R10),WRK_L_RSLNXT(R10) ;RELOAD RESULT DESCRIPTOR POINTE
58   50   D0 0721   997         MOVL    R0,R8                               ;SAVE FINAL STATUS VALUE
   F8D9'   30 0724   998 10$:    BSBW    DCL$GETDVAL                         ;GET NEXT DESCRIPTOR VALUES
55   04   D1 0727   999         CMPL    #PTR_K_ENDLINE,R5                   ;END OF DESCRIPTORS?
     18   13 072A  1000         BEQL    20$                                 ;IF EQL YES
55   00   D1 072C  1001         CMPL    #PTR_K_COMDQUAL,R5                  ;COMMAND QUALIFIER?
     F3   12 072F  1002         BNEQ    10$                                 ;IF NEQ NO
   F8CC'   30 0731  1003         BSBW    DCL$GETNVAL                         ;GET QUALIFIER NUMBER
57   51   D1 0734  1004         CMPL    R1,R7                               ;QUALIFIER TYPE MATCH?
     EB   12 0737  1005         BNEQ    10$                                 ;IF NEQ NO
50   58   D0 0739  1006         MOVL    R8,R0                               ;RESTORE ERROR STATUS CODE
   F8C1'   30 073C  1007         BSBW    DCL$SAVE_STATUS                     ;SET INTO $STATUS
     8E   D5 073F  1008         TSTL    (SP)+                               ;REMOVE RETURN FROM STACK
   F8BC'   31 0741  1009         BRW     DCL$GOTO                            ;EXECUTE GOTO
50   58   D0 0744  1010 20$:    MOVL    R8,R0                               ;RESTORE FINAL STATUS
          05 0747  1011         RSB                                         ;
             0748  1012
             0748  1013         .END
```

```
$$.TMP1                 = 00000001
$$.TMP2                 = 00000062
CHECKPPF                  0000062D R      02
CLI$K_CLOS_LOG            ********   X    02
CLI$K_OPEN_APPE           ********   X    02
CLI$K_OPEN_END            ********   X    02
CLI$K_OPEN_ERRO           ********   X    02
CLI$K_OPEN_READ           ********   X    02
CLI$K_OPEN_SHAR           ********   X    02
CLI$K_OPEN_WRIT           ********   X    02
CLI$K_READ_DELE           ********   X    02
CLI$K_READ_INDE           ********   X    02
CLI$K_READ_KEY            ********   X    02
CLI$K_READ_LOCK           ********   X    02
CLI$K_READ_MATC           ********   X    02
CLI$K_READ_PROM           ********   X    02
CLI$K_READ_TIME           ********   X    02
CLI$K_WRIT_SYMB           ********   X    02
CLI$K_WRIT_UPDA           ********   X    02
CLI$_BUFOVF             = 00038018
CLI$_INVRANGE           = 00038228
CLI$_INVRFM             = 000388D2
CLI$_IVCHAR             = 00038050
CLI$_IVKEYW             = 00038060
CLI$_NOCCAT             = 000380A8
CLI$_NORMAL             = 00030001
CLI$_SYMLNG             = 00038270
CLI$_SYMOVF             = 00038138
CLI$_UNDFIL             = 00038188
CLI$_UNDSYM             = 00038140
DATAP                     00000000 R      02
DCL$ALLDYNMEM             ********   X    02
DCL$ALLOCSYM              ********   X    02
DCL$CHKALPHA              ********   X    02
DCL$CLOSE                 00000020 RG     02
DCL$CNVASCBIN             ********   X    02
DCL$CNVNOEDIT             ********   X    02
DCL$COMPSTRING            ********   X    02
DCL$DEADYNMEM             ********   X    02
DCL$GETDVAL               ********   X    02
DCL$GETNVAL               ********   X    02
DCL$GOTO                  ********   X    02
DCL$OPEN                  000000A5 RG     02
DCL$OPEN_CREATE           00000593 RG     02
DCL$PUTMSG                ********   X    02
DCL$READ                  000002BE RG     02
DCL$SAVE_STATUS           ********   X    02
DCL$SYM_STRING            ********   X    02
DCL$WRITE                 000004AD RG     02
DFSPEC                    00000007 R      02
DSPXIT                    00000583 R      02
ESCAPE                  = 0000001B
FAB$B_ACMODES           = 0000004A
FAB$B_DNS               = 00000035
FAB$B_FAC               = 00000016
FAB$B_FNS               = 00000034
FAB$B_ORG               = 0000001D
FAB$B_RAT               = 0000001E
FAB$B_RFM               = 0000001F
FAB$B_SHR               = 00000017
FAB$C_IDX               = 00000020
FAB$C_SEQ               = 00000000
FAB$C_UDF               = 00000000
FAB$C_VFC               = 00000003
FAB$L_ALQ               = 00000010
FAB$L_DNA               = 00000030
FAB$L_FNA               = 0000002C
FAB$L_FOP               = 00000004
FAB$L_NAM               = 00000028
FAB$L_STV               = 0000000C
FAB$M_CR                = 00000002
FAB$M_DEL               = 00000004
FAB$M_GET               = 00000002
FAB$M_PPF               = 00040000
FAB$M_PRN               = 00000004
FAB$M_PUT               = 00000001
FAB$M_UPD               = 00000008
FAB$S_FILE_MODE         = 00000002
FAB$V_FILE_MODE         = 00000004
FAB$V_GET               = 00000001
FAB$W_DEQ               = 00000014
FAB$W_IFI               = 00000002
FAB$W_MRS               = 00000036
GETID                     00000663 R      02
IDF_B_OUTFLAGS            00000038
IDF_C_LENGTH              00000074
IDF_K_LENGTH              00000074
IDF_L_FILENAME            00000068
IDF_L_INPRABCTX           0000000C
IDF_L_LNK                 00000000
IDF_L_ONCTLY              00000060
IDF_L_ONERROR             00000008
IDF_L_OUTRABCTX           00000024
IDF_L_SEARCHCTX           00000064
IDF_Q_LABEL               00000018
IDF_Q_LOCAL               00000010
IDF_T_INPDVI              0000003C
IDF_T_OUTDVI              00000028
IDF_W_FLAG                0000005E
IDF_W_INPDID              00000052
IDF_W_INPFID              0000004C
IDF_W_INPIFI              00000004
IDF_W_INPRFA              00000058
IDF_W_ONLEVEL             00000006
IDF_W_OUTIFI              00000020
IDF_W_OUTISI              00000022
LABEL_CHECK               0000071B R      02
LNM$C_NAMLENGTH         = 000000FF
LNM$M_CONFINE           = 00000002
LNM$M_TERMINAL          = 00000200
LNM$PROCESS               00000014 R      02
LNM$V_CONCEALED         = 00000008
LNM$_ATTRIBUTES         = 00000003
LNM$_STRING             = 00000002
```

```
MAXRECSIZE              =  00000800
NAM$B_ESL               =  0000000B
NAM$B_ESS               =  0000000A
NAM$B_RSL               =  00000003
NAM$B_RSS               =  00000002
NAM$C_MAXRSS            =  000000FF
NAM$L_ESA               =  0000000C
NAM$L_FNB               =  00000034
NAM$L_RSA               =  00000004
NAM$T_DVI               =  00000014
NAM$V_CNCL_DEV          =  0000000C
PRC_B_CONTINUE             000000F3
PRC_B_DEFRADIX             000000AE
PRC_B_EXMDEPMOD            000000AD
PRC_B_EXMDEPWID            000000AC
PRC_B_EXONLYL              0000012D
PRC_B_FLAGS2               000000AF
PRC_B_IMGFLAG              00000078
PRC_B_OUTFLAGS             0000012C
PRC_B_PROMPTLEN            000000F0
PRC_C_LENGTH               00000534
PRC_G_COMMANDS             00000133
PRC_G_PROMPT               000000F4
PRC_K_DEC               =  00000001
PRC_K_LENGTH               00000534
PRC_L_CURRKEY              00000048
PRC_L_EXMDEPADR            000000A8
PRC_L_EXTARG               00000094
PRC_L_EXTBLK               0000008C
PRC_L_EXTCOD               0000009C
PRC_L_EXTHND               00000090
PRC_L_EXTPRM               00000098
PRC_L_IDFLNK               000000BC
PRC_L_IMGACTSTS            00000080
PRC_L_INDCLOCK             0000007C
PRC_L_INDEPTH              0000005C
PRC_L_INDFAB               0000001C
PRC_L_INDINPRAB            00000014
PRC_L_INDOUTRAB            00000018
PRC_L_INPRAB               00000008
PRC_L_LASTKEY              0000004C
PRC_L_LSTSTATUS            000000B0
PRC_L_ONCTLY               000000B8
PRC_L_ONERROR              0000006C
PRC_L_OUTOFBAND            000000B4
PRC_L_OUTRAB               0000000C
PRC_L_OUTRABCTX            00000118
PRC_L_PPFLIST              00000070
PRC_L_RECALLPTR            0000012F
PRC_L_RESTART              00000058
PRC_L_SAVAP                00000000
PRC_L_SAVFP                00000004
PRC_L_SEVERITY             00000050
PRC_L_SPWN                 000000C0
PRC_L_STACKLM              000000A4
PRC_L_STACKPT              000000A0
PRC_L_STATUS               00000054
PRC_L_STS                  00000084
PRC_L_STV                  00000088
PRC_L_SYMBOL               00000060
PRC_L_TMBX                 00000074
PRC_L_TRMLIST              00000010
PRC_M_DISABL            =  00000004
PRC_Q_ALLOCREG             00000020
PRC_Q_COMMAND              000000E0
PRC_Q_FLUSHTIME            000000D0
PRC_Q_GLOBAL               00000028
PRC_Q_IMAGENAME            000000D8
PRC_Q_KEYPAD               00000040
PRC_Q_LABEL                00000030
PRC_Q_LOCAL                00000038
PRC_Q_SAVEPRIV             000000E8
PRC_T_OUTDVI               0000011C
PRC_W_ASTIOSB              000000C6
PRC_W_ASTRETN              000000C8
PRC_W_ASTSTATUS            000000C4
PRC_W_ATTMBX               0000007A
PRC_W_FLAGS                00000068
PRC_W_INPCHAN              00000064
PRC_W_ONLEVEL              0000006A
PRC_W_OUTIFI               00000114
PRC_W_OUTISI               00000116
PRC_W_OUTMBXCHN            000000CA
PRC_W_OUTMBXREF            000000CE
PRC_W_OUTMBXSIZ            000000CC
PRC_W_PMPTCTRL             000000F1
PRC_W_WAITIOSB             00000066
PSL$C_SUPER             =  00000002
PSL$C_USER              =  00000003
PTR_B_LEVEL                00000004
PTR_B_NUMBER               00000005
PTR_B_PARMCNT              00000006
PTR_B_VALUE                00000000
PTR_C_LENGTH               0000000C
PTR_K_COLON             =  00000002
PTR_K_COMDQUAL          =  00000000
PTR_K_COMMA             =  00000005
PTR_K_ENDLINE           =  00000004
PTR_K_LENGTH               0000000C
PTR_K_PARAMETR          =  00000003
PTR_K_PLUS              =  00000004
PTR_L_DESCR                00000000
PTR_L_ENTITY               00000008
PTR_V_FLAGS             =  00000014
PTR_V_NEGATE            =  00000014
RAB$B_KRF               =  00000035
RAB$B_KSZ               =  00000034
RAB$B_PSZ               =  00000034
RAB$B_RAC               =  0000001E
RAB$B_TMO               =  0000001F
RAB$C_BLN               =  00000044
RAB$C_KEY               =  00000001
RAB$C_SEQ               =  00000000
RAB$L_CTX               =  00000018
```

```
RAB$L_KBF                       = 00000030
RAB$L_PBF                       = 00000030
RAB$L_RBF                       = 00000028
RAB$L_ROP                       = 00000004
RAB$L_UBF                       = 00000024
RAB$S_PPF_RAT                   = 00000008
RAB$V_EOF                       = 00000008
RAB$V_ETO                       = 0000001C
RAB$V_KGE                       = 00000015
RAB$V_KGT                       = 00000016
RAB$V_NLK                       = 00000014
RAB$V_PMT                       = 0000001E
RAB$V_PPF_RAT                   = 00000006
RAB$V_RRL                       = 00000003
RAB$V_TMO                       = 00000019
RAB$W_ISI                       = 00000002
RAB$W_RSZ                       = 00000022
RAB$W_USZ                       = 00000020
READ                              0000000C R       02
RMS$_EOF                          ********   X     02
SHR$_OPENIN                       ********   X     02
SHR$_OPENOUT                      ********   X     02
SS$_NORMAL                        ********   X     02
STS$K_ERROR                     = 00000002
STS$M_FAC_NO                    = 0FFF0000
STS$M_INHIB_MSG                 = 10000000
STS$M_SEVERITY                  = 00000007
STS$V_SEVERITY                  = 00000000
SYM_B_FLAGS                       0000000B
SYM_B_NONUNIQUE                   0000000B
SYM_B_TYPE                        0000000A
SYM_K_STRING                    = 00000000
SYM_L_BL                          00000004
SYM_L_FL                          00000000
SYM_T_SYMBOL                      0000000C
SYM_W_SIZE                        00000008
SYS$CLOSE                         ********   GX    02
SYS$CONNECT                       ********   GX    02
SYS$CREATE                        ********   GX    02
SYS$CRELNM                        ********   GX    02
SYS$DELETE                        ********   GX    02
SYS$DELLNM                        ********   GX    02
SYS$GET                           ********   GX    02
SYS$OPEN                          ********   GX    02
SYS$PUT                           ********   GX    02
SYS$TRNLOG                        ********   GX    02
SYS$UPDATE                        ********   GX    02
WRITE                             00000010 R       02
WRK_B_CMDOPT                      FFFFFFC3
WRK_B_MAXPARM                     FFFFFFD0
WRK_B_MINPARM                     FFFFFFD1
WRK_B_PARMCNT                     FFFFFFCE
WRK_B_PARMSUM                     FFFFFFCF
WRK_B_RECALLCNT                   FFFFFFC5
WRK_B_VALLEV                      FFFFFFC4
WRK_B_VERBTYP                     FFFFFFC2
WRK_C_LENGTH                      FFFFF486
WRK_G_BUFFER                      FFFFF492
WRK_G_INPBUF                      FFFFF896
WRK_G_RESULT                      FFFFF9B6
WRK_K_LENGTH                      FFFFF486
WRK_L_CHARPTR                     FFFFF48E
WRK_L_DISALLOW                    FFFFFFE6
WRK_L_ERRORRTN                    FFFFF9AE
WRK_L_EXPANDPTR                   FFFFF486
WRK_L_IMAGE                       FFFFFFE2
WRK_L_MARKPTR                     FFFFF48A
WRK_L_PAROUT                      FFFFFFD2
WRK_L_PMPTADDR                    FFFFF9A2
WRK_L_PROMPTRTN                   FFFFF9A6
WRK_L_PROPTR                      FFFFFFC6
WRK_L_QUABLK                      FFFFFFCA
WRK_L_READRTN                     FFFFF9AA
WRK_L_RECALLPTR                   FFFFFFEA
WRK_L_RSLEND                      FFFFFFB6
WRK_L_RSLNXT                      FFFFFFBA
WRK_L_SAVAP                       FFFFFFF8
WRK_L_SAVFP                       FFFFFFFC
WRK_L_SAVSP                       FFFFFFF4
WRK_L_SIGNALRTN                   FFFFFFD6
WRK_L_SPECRTN                     FFFFF9B2
WRK_L_TAB_VEC                     FFFFFFDE
WRK_L_VERB                        FFFFFFBE
WRK_V_CLOSERR                   = 00000002
WRK_V_COMMAND                   = 00000001
WRK_W_FLAGS                       FFFFFFF0
WRK_W_FLAGS2                      FFFFFFF2
WRK_W_IMGCHAN                     FFFFFFEE
WRK_W_PMPTLEN                     FFFFF99E
_$$_                            = 000000EF
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	FFFFFFFC (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DCL$ZCODE	00000748 (1864.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	18	00:00:00.09	00:00:00.84
Command processing	90	00:00:00.61	00:00:06.00
Pass 1	368	00:00:15.46	00:00:45.83
Symbol table sort	0	00:00:01.65	00:00:05.84
Pass 2	180	00:00:03.43	00:00:10.89
Symbol table output	37	00:00:00.22	00:00:00.94
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	695	00:00:21.48	00:01:10.36

The working set limit was 1500 pages.
80381 bytes (157 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1107 non-local and 84 local symbols.
1013 source lines were read in Pass 1, producing 20 object records in Pass 2.
59 pages of virtual memory were used to define 41 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYSLIB]SYSBLDMLB.MLB;1	0
_255DUA28:[DCL.OBJ]DCL.MLB;1	10
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	23
TOTALS (all libraries)	33

1350 GETS were required to define 33 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:FILECMDS/OBJ=OBJ$:FILECMDS MSRC$:FILECMDS/UPDATE=(ENH$:FILECMDS)+EXECML$/LIB+LIB$:DCL/LIB+SYS$LIBRARY:SYSBLDMLB/LIB
```

GETKEYNAM
LIS

GOTO
LIS

EXPRESS
LIS

HANDLE
LIS

IMAGECTRL
LIS

INITIAL
LIS

INDIRECT
LIS

FILECMDS
LIS

IF
LIS

IMAGEXECT
LIS