```
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
```

```
RRRRRRRR    EEEEEEEEEE    CCCCCCCC    AAAAAA    LL          LL          SSSSSSSS  UU      UU  BBBBBBBB
RRRRRRRR    EEEEEEEEEE    CCCCCCCC    AAAAAA    LL          LL          SSSSSSSS  UU      UU  BBBBBBBB
RR      RR  EE          CC          AA      AA  LL          LL        SS          UU      UU  BB      BB
RR      RR  EE          CC          AA      AA  LL          LL        SS          UU      UU  BB      BB
RR      RR  EE          CC          AA      AA  LL          LL        SS          UU      UU  BB      BB
RR      RR  EE          CC          AA      AA  LL          LL        SS          UU      UU  BB      BB
RRRRRRRR    EEEEEEEE    CC          AA      AA  LL          LL          SSSSSS    UU      UU  BBBBBBBB
RRRRRRRR    EEEEEEEE    CC          AA      AA  LL          LL          SSSSSS    UU      UU  BBBBBBBB
RR  RR      EE          CC          AAAAAAAAAA  LL          LL                SS  UU      UU  BB      BB
RR  RR      EE          CC          AAAAAAAAAA  LL          LL                SS  UU      UU  BB      BB
RR    RR    EE          CC          AA      AA  LL          LL                SS  UU      UU  BB      BB      ....
RR    RR    EE          CC          AA      AA  LL          LL                SS  UU      UU  BB      BB      ....
RR      RR  EEEEEEEEEE    CCCCCCCC  AA      AA  LLLLLLLLLL  LLLLLLLLLL  SSSSSSSS    UUUUUUUUUU  BBBBBBBB        ....
RR      RR  EEEEEEEEEE    CCCCCCCC  AA      AA  LLLLLLLLLL  LLLLLLLLLL  SSSSSSSS    UUUUUUUUUU  BBBBBBBB        ....
```

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

```
;     1          0001  0 MODULE recallsub (IDENT='V04-000',
;     2          0002  0          ADDRESSING_MODE(NONEXTERNAL=LONG_RELATIVE,
;     3          0003  0                          EXTERNAL=GENERAL)) =
;     4          0004  0
;     5          0005  1 BEGIN
;     6          0006  1
;     7          0007  1 !
;     8          0008  1 !*****************************************************************************
;     9          0009  1 !*                                                                           *
;    10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    12          0012  1 !*  ALL RIGHTS RESERVED.                                                      *
;    13          0013  1 !*                                                                           *
;    14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
;    15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
;    16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
;    17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
;    18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
;    19          0019  1 !*  TRANSFERRED.                                                              *
;    20          0020  1 !*                                                                           *
;    21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
;    22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT     *
;    23          0023  1 !*  CORPORATION.                                                              *
;    24          0024  1 !*                                                                           *
;    25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
;    26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    27          0027  1 !*                                                                           *
;    28          0028  1 !*                                                                           *
;    29          0029  1 !*****************************************************************************
;    30          0030  1 !
;    31          0031  1
;    32          0032  1 !++
;    33          0033  1 ! FACILITY:  Command recall routines
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !
;    37          0037  1 !       These routines are used to manage the command recall
;    38          0038    !       functions of the command language intepreter.
;    39          0039  1 !
;    40          0040  1 ! ENVIRONMENT:
;    41          0041  1 !
;    42          0042    !       VAX/VMS operating system. supervisor mode,
;    43          0043  1 !
;    44          0044  1 ! AUTHOR:  Peter George, March 1983
;    45          0045  1 !
;    46          0046  1 ! Modified by:
;    47          0047  1 !
;    48          0048  1 !       V03-005 PCG0005         Peter George    06-Feb-1984
;    49          0049  1 !               Be more discerning about when to insert a space
;    50          0050  1 !               in a recalled command line.
;    51          0051  1 !
;    52          0052  1 !       V03-004 PCG0004         Peter George    03-Jan-1984
;    53          0053  1 !               Modify the structure of the recall buffer.
;    54          0054  1 !
;    55          0055  1 !       V03-003 PCG0003         Peter George    18-Nov-1983
;    56          0056  1 !               Add a routine to get a command by number.
;    57          0057  1 !
```

```
;    58          0058  1 !       V03-002 PCG0002         Peter George    20-Apr-1983
;    59          0059  1 !               Fix bug in EDIT_COMMAND algo  thm.
;    60          0060  1 !
;    61          0061  1 !       V03-001 PCG0001         Peter George    30-Mar-1983
;    62          0062  1 !               Redo EDIT_COMMAND algorithm.
;    63          0063  1 !--
;    64          0064  1
;    65          0065  1 !
;    66          0066  1 ! Include files
;    67          0067  1 !
;    68          0068  1 LIBRARY 'SYS$LIBRARY:LIB';
;    69          0069  1 REQUIRE 'LIB$:DCLDEF';                  ! DCL definitions
```

```
;    71          1141  1 !
;    72          1142  1 ! Table of contents
;    73          1143  1 !
;    74          1144  1
;    75          1145  1 LINKAGE
;    76          1146  1     common_linkage = call : GLOBAL (wrk=10,prc=11),
;    77          1147  1     ptr_linkage = call : GLOBAL (ptr=9,wrk=10,prc=11);
;    78          1148  1
;    79          1149  1 FORWARD ROUTINE
;    80          1150  1     dcl$put_command : common_linkage,   ! Put command in buffer
;    81          1151  1     dcl$put_segment : common_linkage,   ! Add to last command in buffer
;    82          1152  1     compare_string : ptr_linkage,       ! Compare current and last commands
;    83          1153  1     insert_string : ptr_linkage,        ! Insert a string in the buffer
;    84          1154  1     zero_buffer : ptr_linkage,          ! Zero part of the buffer
;    85          1155  1     edit_command : ptr_linkage,         ! Edit previous command
;    86          1156  1     dcl$get_next_command:common_linkage,! Get next command from buffer
;    87          1157  1     dcl$get_prev_command:common_linkage,! Get previous command from buffer
;    88          1158  1     dcl$get_curr_   mand:common_linkage;! Get current command from buffer
;    89          1159  1
;    90          1160  1 !
;    91          1161  1 ! Change name of the PSECT's to conform to DCL standards.
;    92          1162  1 !
;    93          1163  1 PSECT PLIT = DCL$ZCODE(EXECUTE, ALIGN(0));
;    94          1164  1 PSECT CODE = DCL$ZCODE(EXECUTE, ALIGN(0));
;    95          1165  1
;    96          1166  1 LITERAL
;    97          1167  1     true - 1,
;    98          1168  1     false
;    99          1169  1
;   100          1170  1 !
;   101          1171  1 ! Macros to check for ends of command buffer
;   102          1172  1 !
;   103        M 1173  1 MACRO overflow (address) =
;   104          1174  1     (address) GEQU prc [prc_g_commands] + prc_c_cmdbufsiz%;
;   105          1175  1
;   106        M 1176  1 MACRO underflow (address) =
;   107          1177  1     (address) LSSU prc [prc_g_commands]%;
;   108          1178  1
```

```
;  110          1179  1 GLOBAL ROUTINE dcl$put_command (desc) : common_linkage =
;  111          1180  1
;  112          1181  1 !---
;  113          1182  1 !
;  114          1183  1 !       Put a command into the command buffer.
;  115          1184  1 !
;  116          1185  1 ! Inputs:
;  117          1186  1 !
;  118          1187  1 !       desc = address of descriptor of command to insert
;  119          1188  1 !       R10 = address of WRK data structure
;  120          1189  1 !       R11 = address of PRC data structure
;  121          1190  1 !       PRC_L_RECALLPTR = pointer to location to insert command at
;  122          1191  1 !
;  123          1192  1 ! Outputs:
;  124          1193  1 !
;  125          1194  1 !       The command is added to the buffer and PRC_L_RECALLPTR is updated
;  126          1195  1 !       to point to the next free space in the buffer.
;  127          1196  1 !
;  128          1197  1 !       The structure of the circular recall buffer is as follows:
;  129          1198  1 !
;  130          1199  1 !               0-byte,len-byte,char-string,len-byte,
;  131          1200  1 !               0-byte,len-byte,char-string,len-byte, ...
;  132          1201  1 !
;  133          1202  1 !       routine value = always true
;  134          1203  1 !
;  135          1204  1 !---
;  136          1205  1
;  137          1206  2 BEGIN
;  138          1207  2
;  139          1208  2
;  140          1209  2 MAP
;  141          1210  2     desc :      REF VECTOR;             ! Input command descriptor
;  142          1211  2
;  143          1212  2 GLOBAL REGISTER
;  144          1213  2     ptr=9 :     REF VECTOR[,BYTE];      ! Pointer into recall buffer
;  145          1214  2
;  146          1215  2 EXTERNAL REGISTER
;  147          1216  2     wrk=10 :    REF $BBLOCK,            ! Address of WRK data structure
;  148          1217  2     prc=11 :    REF $BBLOCK;            ! Address of PRC data structure
;  149          1218  2
;  150          1219  2 !
;  151          1220  2 ! Compare the new command to the previous command.  If identical, then do not
;  152          1221  2 ! insert the new command in the buffer.
;  153          1222  2 !
;  154          1223  2 IF compare_string (.desc)
;  155          1224  2     THEN RETURN true;
;  156          1225  2
;  157          1226  2 !
;  158          1227  2 ! Skip past the leading zero and insert the command length.
;  159          1228  2 !
;  160          1229  2 ptr = .prc [prc_l_recallptr] + 1;
;  161          1230  2 IF OVERFLOW (.ptr)
;  162          1231  2     THEN ptr = .ptr - prc_c_cmdbufsiz;
;  163          1232  2 ptr [0] = .desc [0];
;  164          1233  2
;  165          1234  2 !
;  166          1235  2 ! Copy the command string into the buffer and insert the trailing length byte.
```

```
;  167          1236  2 !
;  168          1237  2 ptr = .ptr + 1;
;  169          1238  2 IF OVERFLOW (.ptr)
;  170          1239  2    THEN ptr = prc [prc_g_commands];
;  171          1240  2 insert_string (.desc);
;  172          1241  2 ptr [0] = .desc [0];
;  173          1242  2
;  174          1243  2 !
;  175          1244  2 ! Zero any partially overwritten commands in the buffer and reset the
;  176          1245  2 ! pointer in the PRC data structure to the next free command space.
;  177          1246  2 !
;  178          1247  2 ptr = .ptr + 1;
;  179          1248  2 IF OVERFLOW (.ptr)
;  180          1249  2    THEN ptr= prc [prc_g_commands];
;  181          1250  2 prc [prc_l_recallptr] = .ptr;
;  182          1251  2 RETURN zero_buffer();
;  183          1252  2
;  184          1253  1 END;

                                                        .TITLE  RECALLSUB
                                                        .IDENT  \V04-000\


                                                        .PSECT  DCL$ZCODE,NOWRT,0

                                    0200 00000          .ENTRY  DCL$PUT_COMMAND, Save R9                        ; 1179
                                04  AC  DD 00002       PUSHL   DESC                                            ; 1223
                00000000V       EF      01  FB 00005   CALLS   #1, COMPARE_STRING                              ;
                                04      50  E9 0000C   BLBC    R0, 1$                                          ;
                                50      01  D0 0000F   MOVL    #1, R0                                          ; 1224
                                            04 00012    RET                                                     ;
                    59      012F    CB      01  C1 00013 1$:  ADDL3   #1, 303(PRC), PTR                       ; 1229
                                50      0534    CB  9E 00019  MOVAB   1332(R11), R0                           ; 1230
                                50              59  D1 0001E  CMPL    PTR, R0                                 ;
                                                05  1F 00021  BLSSU   2$                                      ;
                                59      FBFF    C9  9E 00023  MOVAB   -1025(R9), PTR                          ; 1231
                                89        04    BC  90 00028 2$:  MOVB    @DESC, (PTR)+                       ; 1232
                                50              59  D1 0002C  CMPL    PTR, R0                                 ; 1238
                                                05  1F 0002F  BLSSU   3$                                      ;
                                59      0133    CB  9E 00031  MOVAB   307(R11), PTR                           ; 1239
                                  04            AC  DD 00036 3$:  PUSHL   DESC                                ; 1240
                    00000000V   EF              01  FB 00039  CALLS   #1, INSERT_STRING                       ;
                                89        04    BC  90 00040  MOVB    @DESC, (PTR)+                           ; 1241
                                50      0534    CB  9E 00044  MOVAB   1332(R11), R0                           ; 1248
                                50              59  D1 00049  CMPL    PTR, R0                                 ;
                                                05  1F 0004C  BLSSU   4$                                      ;
                                59      0133    CB  9E 0004E  MOVAB   307(R11), PTR                           ; 1249
                        012F    CB              59  D0 00053 4$:  MOVL    PTR, 303(PRC)                       ; 1250
                    00000000V   EF              00  FB 00058  CALLS   #0, ZERO_BUFFER                         ; 1251
                                                04 0005F    RET                                                 ; 1253

; Routine Size:  96 bytes,    Routine Base:  DCL$ZCODE + 0000
```

```
;  186          1254  1 GLOBAL ROUTINE dcl$put_segment (desc) : common_linkage =
;  187          1255  1
;  188          1256  1 !---
;  189          1257  1 !
;  190          1258  1 !       Add a command segment to the last command in the command buffer.
;  191          1259  1 !       If it causes the command to be longer that WRK_C_INPBUFSIZ-1 in
;  192          1260  1 !       length, then insert it as a new entry.
;  193          1261  1 !
;  194          1262  1 ! Inputs:
;  195          1263  1 !
;  196          1264  1 !       desc = address of descriptor of command to insert
;  197          1265  1 !       R10 = address of WRK data structure
;  198          1266  1 !       R11 = address of PRC data structure
;  199          1267  1 !       PRC_L_RECALLPTR = pointer past end of last inserted command
;  200          1268  1 !
;  201          1269  1 ! Outputs:
;  202          1270  1 !
;  203          1271  1 !       The command segment is added to the buffer and PRC_L_RECALLPTR is
;  204          1272  1 !       updated to point to the next free space in the buffer.
;  205          1273  1 !
;  206          1274  1 !       routine value = always true
;  207          1275  1 !
;  208          1276  1 !---
;  209          1277  1
;  210          1278  2 BEGIN
;  211          1279  2
;  212          1280  2 MAP
;  213          1281  2     desc :      REF VECTOR;             ! Input command descriptor
;  214          1282  2
;  215          1283  2 GLOBAL REGISTER
;  216          1284  2     ptr=9 :     REF VECTOR[,BYTE];      ! Pointer into recall buffer
;  217          1285  2
;  218          1286  2 EXTERNAL REGISTER
;  219          1287  2     wrk=10 :    REF $BBLOCK,            ! Address of WRK data structure
;  220          1288  2     prc=11 :    REF $BBLOCK;            ! Address of PRC data structure
;  221          1289  2
;  222          1290  2 LOCAL
;  223          1291  2     lead_len :  REF VECTOR[,BYTE];      ! Pointer to leading length in buffer
;  224          1292  2
;  225          1293  2 !
;  226          1294  2 ! Get the length of the previous command.  If the total concatenated length
;  227          1295  2 ! of the command will now be greater than 255, then treat the new segment
;  228          1296  2 ! as a new command.
;  229          1297  2 !
;  230          1298  2 ptr = .prc [prc_l_recallptr] - 1;
;  231          1299  2 IF UNDERFLOW (.ptr)
;  232          1300  2     THEN ptr = .ptr + prc_c_cmdbufsiz;
;  233          1301  2 IF (.ptr [0] + .desc [0]) GTR wrk_c_inpbufsiz - 1
;  234          1302  2     THEN RETURN dcl$put_command (.desc);
;  235          1303  2
;  236          1304  2 !
;  237          1305  2 ! Point at the first character of the previous command string and save
;  238          1306  2 ! the address of the byte to insert the leading length at for later use.
;  239          1307  2 !
;  240          1308  2 ptr = .ptr - .ptr [0];
;  241          1309  2 IF UNDERFLOW (.ptr)
;  242          1310  2     THEN ptr = .ptr + prc_c_cmdbufsiz;
```

```
;  243          1311  2 lead_len = .ptr - 1;
;  244          1312  2 IF UNDERFLOW (.lead_len)
;  245          1313  2    THEN lead_len = .lead_len + prc_c_cmdbufsiz;
;  246          1314  2
;  247          1315  2 !
;  248          1316  2 ! Remove the trailing continuation character or comments from the previously
;  249          1317  2 ! inserted part of the command and insert the new segment at the end.
;  250          1318  2 !
;  251          1319  2 edit_command (.lead_len);
;  252          1320  2 insert_string (.desc);
;  253          1321  2
;  254          1322  2 !
;  255          1323  2 ! Set the length bytes.
;  256          1324  2 !
;  257          1325  2 ptr [0] = .ptr - .lead_len - 1;
;  258          1326  2 lead_len [0] = .ptr [0];
;  259          1327  2
;  260          1328  2 !
;  261          1329  2 ! Zero any partially overwritten commands in the buffer and reset the
;  262          1330  2 ! pointer in the PRC data structure to the next free command space.
;  263          1331  2 !
;  264          1332  2 ptr = .ptr + 1;
;  265          1333  2 IF OVERFLOW (.ptr)
;  266          1334  2    THEN ptr= prc [prc_g_commands];
;  267          1335  2 prc [prc_l_recallptr] = .ptr;
;  268          1336  2 RETURN zero_buffer();
;  269          1337  2
;  270          1338  1 END;


                                0204 00000          .ENTRY  DCL$PUT_SEGMENT, Save R2,R9                 ; 1254
59      012F    LB              01   C3 00002          SUBL3   #1, 303(PRC), PTR                           ; 1298
                50      0133    CB   9E 00008          MOVAB   307(R11), R0                                ; 1299
                50              59   D1 0000D          CMPL    PTR, R0                                     ;
                                05   1E 00010          BGEQU   1$                                          ;
                59      0401    C9   9E 00012          MOVAB   1025(R9), PTR                               ; 1300
                50              69   9A 00017 1$:      MOVZBL  (PTR), R0                                   ; 1301
                50        04    BC   C0 0001A          ADDL2   @DESC, R0                                   ;
            000000FF    8F              50   D1 0001E  CMPL    R0, #255                                    ;
                                09   15 00025          BLEQ    2$                                          ;
                              04    AC   DD 00027          PUSHL   DESC                                        ; 1302
                FF71        CF          01   FB 0002A          CALLS   #1, DCL$PUT_COMMAND                         ;
                                04   0002F              RET                                                 ;
                50              69   9A 00030 2$:      MOVZBL  (PTR), R0                                   ; 1308
                59              50   C2 00033          SUBL2   R0, PTR                                     ;
                50      0133    CB   9E 00036          MOVAB   307(R11), R0                                ; 1309
                50              59   D1 0003B          CMPL    PTR, R0                                     ;
                                05   1E 0003E          BGEQU   3$                                          ;
                59      0401    C9   9E 00040          MOVAB   1025(R9), PTR                               ; 1310
                52        FF    A9   9E 00045 3$:      MOVAB   -1(R9), LEAD_LEN                            ; 1311
                50              52   D1 00049          CMPL    LEAD_LEN, R0                                ; 1312
                                05   1E 0004C          BGEQU   4$                                          ;
                52      0401    C2   9E 0004E          MOVAB   1025(R2), LEAD_LEN                          ; 1313
                                52   DD 00053 4$:      PUSHL   LEAD_LEN                                    ; 1319
```

```
                    00000000V  EF              01  FB 00055          CALLS   #1, EDIT_COMMAND                        ;
                                            04  AC  DD 0005C          PUSHL   DESC                                   ; 1320
                    00000000V  EF              01  FB 00C5F          CALLS   #1, INSERT_STRING                      ;
                 50             59             52  C3 00066          SUBL3   LEAD_LEN, PTR, R0                      ; 1325
                 69             50             01  83 0006A          SUBB3   #1, R0, (PTR)                          ;
                                62             89  90 0006E          MOVB    (PTR)+, (LEAD_LEN)                     ; 1326
                                50      0534   CB  9E 00071          MOVAB   1332(R11), R0                          ; 1333
                                50             59  D1 00076          CMPL    PTR, R0                                ;
                                               05  1F 00079          BLSSU   5$                                     ;
                                59      0133   CB  9E 0007B          MOVAB   307(R11), PTR                          ; 1334
                         012F   CB              59  D0 00080  5$:     MOVL    PTR, 303(PRC)                          ; 1335
                    00000000V  EF              00  FB 00085          CALLS   #0, ZERO_BUFFER                        ; 1336
                                               04  0008C          RET                                            ; 1338

; Routine Size:  141 bytes,    Routine Base:  DCL$ZCODE + 0060
```

```
;  272     1339  1 ROUTINE compare_string (desc) : ptr_linkage =
;  273     1340  1
;  274     1341  1 !---
;  275     1342  1 !
;  276     1343  1 !       Compare the new command to the previous command.
;  277     1344  1 !       If identical, then return true.
;  278     1345  1 !
;  279     1346  1 ! Inputs:
;  280     1347  1 !
;  281     1348  1 !       R10 = address of WRK data structure
;  282     1349  1 !       R11 = address of PRC data structure
;  283     1350  1 !       PRC_L_RECALLPTR = pointer past end of last inserted command
;  284     1351  1 !
;  285     1352  1 ! Outputs:
;  286     1353  1 !
;  287     1354  1 !       routine value = true if strings are the same
;  288     1355  1 !                       false otherwise
;  289     1356  1 !---
;  290     1357  1
;  291     1358  2 BEGIN
;  292     1359  2
;  293     1360  2 MAP
;  294     1361  2     desc :      REF VECTOR;             ! Input command descriptor
;  295     1362  2
;  296     1363  2 EXTERNAL REGISTER
;  297     1364  2     ptr=9 :     REF VECTOR[,BYTE],      ! Pointer into recall buffer
;  298     1365  2     wrk=10 :    REF $BBLOCK,            ! Address of WRK data structure
;  299     1366  2     prc=11 :    REF $BBLOCK;            ! Address of PRC data structure
;  300     1367  2
;  301     1368  2 LOCAL
;  302     1369  2     len;
;  303     1370  2
;  304     1371  2 !
;  305     1372  2 ! Get length and address of previous command string.
;  306     1373  2 !
;  307     1374  2 ptr = .prc [prc_l_recallptr] - 1;
;  308     1375  2 IF UNDERFLOW (.ptr)
;  309     1376  2     THEN ptr = .ptr + prc_c_cmdbufsiz;
;  310     1377  2 len = .ptr [0];
;  311     1378  2 ptr = .ptr - .len;
;  312     1379  2 IF UNDERFLOW (.ptr)
;  313     1380  2     THEN ptr = .ptr + prc_c_cmdbufsiz;
;  314     1381  2
;  315     1382  2 !
;  316     1383  2 ! Compare the two strings and return false if they are different.
;  317     1384  2 !
;  318     1385  2 IF OVERFLOW (.ptr + .len - 1)                    ! Will we wrap around?
;  319     1386  3     THEN BEGIN                                    ! Yes, then compare in two pieces
;  320     1387  3          LOCAL temp_len;
;  321     1388  3          temp_len = prc [prc_g_commands] +        ! Get length of first piece
;  322     1389  3                     prc_c_cmdbufsiz - .ptr;     !
;  323     1390  3          IF CH$NEQ (.temp_len, .ptr,             ! Compare first piece
;  324     1391  3                     .desc [0], .desc [1], %C' ')
;  325     1392  3             THEN RETURN false;                   ! Return false if not equal
;  326     1393  3          IF CH$NEQ (.len - .temp_len,            ! Compare second piece
;  327     1394  3                     prc [prc_g_commands],        !
;  328     1395  3                     .desc [0] + .temp_len,       !
```

```
;  329          1396  3                  .desc [1] + .temp_len,                  !
;  330          1397  3                  %C' ')                                  !
;  331          1398  3              THEN RETURN false;                          ! Return false if not equal
;  332          1399  3          END                                             !
;  333          1400  2      ELSE IF CH$NEQ (.len, .ptr,                         ! Compare in whole
;  334          1401  2                  .desc [0], .desc [1], %C' ')            !
;  335          1402  2              THEN RETURN false;                          ! Return false if not equal
;  336          1403  2                                                          !
;  337          1404  2  RETURN true;                                           ! Return true if equal
;  338          1405  1  END;


                                                00FC 00000 COMPARE_STRING:
                                                           .WORD   Save R2,R3,R4,R5,R6,R7                          ; 1339
                59      012F    CB                 01   C3 00002                SUBL3   #1, 303(PRC), PTR               ; 1374
                                57      0133       CB   9E 00008                MOVAB   307(PRC), R7                    ; 1375
                                57                 59   D1 0000D                CMPL    PTR, R7                         ;
                                                   05   1E 00010                BGEQU   1$                              ;
                                59      0401       C9   9E 00012                MOVAB   1025(R9), PTR                   ; 1376
                                54                 69   9A 00017 1$:           MOVZBL  (PTR), LEN                      ; 1377
                                59                 54   C2 0001A                SUBL2   LEN, PTR                        ; 1378
                                57                 59   D1 0001D                CMPL    PTR, R7                         ; 1379
                                                   05   1E 00020                BGEQU   2$                              ;
                                59      0401       C9   9E 00022                MOVAB   1025(R9), PTR                   ; 1380
                                55        04       AC   D0 00027 2$:           MOVL    DESC, R5                        ; 1391
                                51        FF   A449  9E 0002B                MOVAB   -1(LEN)[PTR], R1                ; 1385
                                50      0534       CB   9E 00030                MOVAB   1332(R11), R0                   ;
                                50                 51   D1 00035                CMPL    R1, R0                          ;
                                                   20   1F 00038                BLSSU   3$                              ;
                        56                         50   59 C3 0003A             SUBL3   PTR, R0, TEMP_LEN               ; 1389
        04      BC      20                         69   56 2D 0003E             CMPC5   TEMP_LEN, (PTR), #32, @DESC, @4(R5) ; 1390
                                                04 B5       00044                                                      ;
                                                   20   12 00046              BNEQ    5$                              ;
                                54                 56   C2 00048             SUBL2   TEMP_LEN, R4                    ; 1393
                        50      04      BC         56   C3 0004B             SUBL3   TEMP_LEN, @DESC, R0             ; 1395
                50      20                         67   54 2D 00050             CMPC5   R4, (R7), #32, R0, @4(R5)[TEMP_LEN] ; 1394
                                                04 B546     00055                                                      ;
                                                   08   11 00058              BRB     4$                              ;
        04      BC      20                         69   54 2D 0005A 3$:         CMPC5   LEN, (PTR), #32, @DESC, @4(R5)  ; 1400
                                                04 B5       00060                                                      ;
                                                   04   12 00062 4$:           BNEQ    5$                              ;
                                50                 01   D0 00064              MOVL    #1, R0                          ; 1404
                                                        04 00067              RET                                     ;
                                                   50   D4 00068 5$:           CLRL    R0                              ; 1405
                                                        04 0006A              RET                                     ;

; Routine Size:  107 bytes,    Routine Base:  DCL$ZCODE + 00ED
```

```
;  340          1406  1 ROUTINE insert_string (desc) : ptr_linkage =
;  341          1407  1
;  342          1408  1 !---
;  343          1409  1 !       Insert a string in the buffer.
;  344          1410  1 !
;  345          1411  1 ! Inputs:
;  346          1412  1 !
;  347          1413  1 !       R9  = address to begin insertion at
;  348          1414  1 !       R10 = address of WRK data structure
;  349          1415  1 !       R11 = address of PRC data structure
;  350          1416  1 !
;  351          1417  1 ! Outputs:
;  352          1418  1 !
;  353          1419  1 !       R9 = address of first byte after the insertion
;  354          1420  1 !
;  355          1421  1 !       routine value = always true
;  356          1422  1 !---
;  357          1423  1
;  358          1424  2 BEGIN
;  359          1425  2
;  360          1426  2 MAP
;  361          1427  2     desc :      REF VECTOR;             ! Input command descriptor
;  362          1428  2
;  363          1429  2 EXTERNAL REGISTER
;  364          1430  2     ptr=9 :     REF VECTOR[,BYTE],      ! Pointer to retrieved command
;  365          1431  2     wrk=10 :    REF $BBLOCK,            ! Address of WRK data structure
;  366          1432  2     prc=11 :    REF $BBLOCK;            ! Address of PRC data structure
;  367          1433  2
;  368          1434  2 IF OVERFLOW (.ptr + .desc [0] - 1)             ! Will we wrap around?
;  369          1435  3    THEN BEGIN                                  ! Yes, then copy in two pieces
;  370          1436  3         LOCAL temp_len;
;  371          1437  3         temp_len = prc [prc_g_commands] +      ! Get length of first piece
;  372          1438  3                    prc_c_cmdbufsiz - .ptr;     !
;  373          1439  3         CH$MOVE (.temp_len, .desc [1], .ptr);  ! Move first piece
;  374          1440  3         CH$MOVE (.desc [0] - .temp_len,        ! Move second piece
;  375          1441  3                  .desc [1] + .temp_len,        !
;  376          1442  3                  prc [prc_g_commands]);        !
;  377          1443  3         ptr = prc [prc_g_commands] +           ! Update the ptr
;  378          1444  3                  .desc [0] - .temp_len;        !
;  379          1445  3         END                                    !
;  380          1446  3    ELSE BEGIN                                  ! No, then copy in whole
;  381          1447  3         CH$MOVE (.desc [0], .desc [1], .ptr);  !
;  382          1448  3         ptr = .ptr + .desc [0];                !
;  383          1449  2         END;                                   !
;  384          1450  2
;  385          1451  2 IF OVERFLOW(.ptr)                              ! Update the pointer
;  386          1452  2    THEN ptr = prc [prc_g_commands];            !
;  387          1453  2
;  388          1454  2 RETURN true;
;  389          1455  1 END;


                                01FC 00000 INSERT_STRING:
                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8                ; 1406
```

```
                                56       04   AC   D0 00002          MOVL    DESC, R6                                ; 1439
                                58       04   BC   D0 00006          MOVL    @DESC, R8                               ; 1434
                                50       FF A849   9E 0000A          MOVAB   -1(R8)[PTR], R0                         ;
                              0534       CB   9F 0000F               PUSHAB  1332(R11)                               ;
                                6E            50   D1 00013          CMPL    R0, (SP)                                ;
                                              23   1F 00016          BLSSU   1$                                      ;
                    57          6E            59   C3 00018          SUBL3   PTR, (SP), TEMP_LEN                     ; 1438
                    69    04    B6            57   28 0001C          MOVC3   TEMP_LEN, @4(R6), (PTR)                 ; 1439
                    50          58            57   C3 00021          SUBL3   TEMP_LEN, R8, R0                        ; 1440
              0133  CB    04 B647             50   28 00025          MOVC3   R0, @4(R6)[TEMP_LEN], 307(PRC)          ; 1442
                    50          5B            58   C1 0002D          ADDL3   R8, PRC, R0                             ; 1443
                                50            57   C2 00031          SUBL2   TEMP_LEN, R0                            ; 1444
                                59       0133   C0   9E 00034          MOVAB   307(R0), PTR                            ;
                                               08   11 00039          BRB     2$                                      ; 1434
                    69    04    B6            58   28 0003B 1$:      MOVC3   R8, @4(R6), (PTR)                       ; 1447
                                59            58   C0 00040          ADDL2   R8, PTR                                 ; 1448
                                6E            59   D1 00043 2$:      CMPL    PTR, (SP)                               ; 1451
                                              05   1F 00046          BLSSU   3$                                      ;
                                59       0133   CB   9E 00048          MOVAB   307(R11), PTR                           ; 1452
                                50            01   D0 0004D 3$:      MOVL    #1, R0                                  ; 1454
                                                   04 00050          RET                                             ; 1455

; Routine Size:  81 bytes,    Routine Base:  DCL$ZCODE + 0158
```

```
;  391          1456  1 ROUTINE zero_buffer : ptr_linkage =
;  392          1457  1
;  393          1458  1 !---
;  394          1459  1 !
;  395          1460  1 !       Zero any partially overwritten commands in the buffer.
;  396          1461  1 !
;  397          1462  1 ! Inputs:
;  398          1463  1 !
;  399          1464  1 !       R9  = address to start zeroing at
;  400          1465  1 !       R10 = address of WRK data structure
;  401          1466  1 !       R11 = address of PRC data structure
;  402          1467  1 !
;  403          1468  1 ! Outputs:
;  404          1469  1 !
;  405          1470  1 !       routine value = always true
;  406          1471  1 !---
;  407          1472  1
;  408          1473  2 BEGIN
;  409          1474  2
;  410          1475  2 EXTERNAL REGISTER
;  411          1476  2     ptr=9 :     REF VECTOR[,BYTE],      ! Pointer to retrieved command
;  412          1477  2     wrk=10 :    REF $BBLOCK,            ! Address of WRK data structure
;  413          1478  2     prc=11 :    REF $BBLOCK;            ! Address of PRC data structure
;  414          1479  2
;  415          1480  3 WHILE (.ptr [0] NEQ 0)
;  416          1481  3 DO BEGIN
;  417          1482  3    ptr [0] = 0;
;  418          1483  3    ptr = .ptr + 1;
;  419          1484  3    IF OVERFLOW (.ptr)
;  420          1485  3       THEN ptr = prc [prc_g_commands];
;  421          1486  2    END;
;  422          1487  2
;  423          1488  2 RETURN true;
;  424          1489  1 END;


                                        0000 ;          BUFFER:
                                                            .WORD   Save nothing                    ; 1456
                                50      0534    CB  9E 00002        MOVAB   1332(R11), R0           ; 1484
                                        69  95 00007  %.            TSTB    (PTR)                   ; 1480
                                        0E  13 00009                BEQL    2$                      ;
                                        89  94 0000B                CLRB    (PTR)+                  ; 1482
                                50          59  D1 0000D            CMPL    PTR, R0                 ; 1484
                                        F5  1F 00010                BLSSU   1$                      ;
                                59      0133    CB  9E 00012        MOVAB   307(R11), PTR           ; 1485
                                        EE  11 00017                BRB     1$                      ; 1480
                                50          01  D0 00019 2$:        MOVL    #1, R0                  ; 1488
                                            04  0001C               RET                             ; 1489

; Routine Size:  29 bytes,    Routine Base:  DCL$ZCODE + 01A9
```

```
;  426     1490  1 ROUTINE edit_command (len) : ptr_linkage =
;  427     1491  1
;  428     1492  1 !---
;  429     1493  1 !
;  430     1494  1 !       Remove the continuation character and/or comment characters from the
;  431     1495  1 !       end of the command line.  Insert a space in their place.
;  432     1496  1 !
;  433     1497  1 ! Inputs:
;  434     1498  1 !
;  435     1499  1 !       len = address of byte length of command
;  436     1500  1 !       R9 = ptr to first character of command string
;  437     1501  1 !       R10 = address of WRK data structure
;  438     1502  1 !       R11 = address of PRC data structure
;  439     1503  1 !
;  440     1504  1 ! Outputs:
;  441     1505  1 !
;  442     1506  1 !       R9 = pt- to end of edited command
;  443     1507  1 !
;  444     1508  1 !       routine value = always true
;  445     1509  1 !
;  446     1510  1 !---
;  447     1511  1
;  448     1512  2 BEGIN
;  449     1513  2
;  450     1514  2 MAP
;  451     1515  2     len :       REF VECTOR[,BYTE];      ! Address of length of command
;  452     1516  2
;  453     1517  2 EXTERNAL REGISTER
;  454     1518  2     ptr=9 :     REF VECTOR[,BYTE],      ! Pointer to end of command to edit
;  455     1519  2     wrk=10 :    REF $BBLOCK,            ! Address of WRK data structure
;  456     1520  2     prc=11 :    REF $BBLOCK;            ! Address of PRC data structure
;  457     1521  2
;  458     1522  2 LOCAL
;  459     1523  2     flags :     BITVECTOR[3];           ! Flags
;  460     1524  2
;  461     1525  2 LITERAL
;  462     1526  2     continue = 0,
;  463     1527  2     blank = 1,
;  464     1528  2     quote = 2;
;  465     1529  2
;  466     1530  2 flags = 0;
;  467     1531  2
;  468     1532  2 !
;  469     1533  2 ! Search for EOL or trailing comment.
;  470     1534  2 !
;  471     1535  2 INCR i FROM 1 TO .len [0]
;  472     1536  3 DO BEGIN
;  473     1537  3
;  474     1538  3     IF .ptr [0] EQL %C'"'
;  475     1539  3        THEN flags [quote] = NOT .flags [quote];
;  476     1540  3
;  477     1541  3     IF NOT .flags [quote]
;  478     1542  3        THEN IF .ptr [0] EQL %C'!'
;  479     1543  3                THEN EXITLOOP;
;  480     1544  3
;  481     1545  3     ptr = .ptr + 1;
;  482     1546  3     IF OVERFLOW (.ptr)
```

```
;  483          1547  3        THEN ptr = prc [prc_g_commands];
;  484          1548  2    END;
;  485          1549  2
;  486          1550  2 !
;  487          1551  2 ! Back up to previous character.
;  488          1552  2 !
;  489          1553  2 ptr = .ptr - 1;
;  490          1554  2 IF UNDERFLOW (.ptr)
;  491          1555  2    THEN ptr = .ptr + prc_c_cmdbufsiz;
;  492          1556  2
;  493          1557  2 !
;  494          1558  2 ! Delete trailing white space.
;  495          1559  2 !
;  496          1560  3 WHILE ((.ptr [0] EQL %X'20') OR (.ptr [0] EQL %X'09'))
;  497          1561  3 DO BEGIN
;  498          1562  3    ptr = .ptr - 1;
;  499          1563  3    IF UNDERFLOW (.ptr)
;  500          1564  3       THEN ptr = .ptr + prc_c_cmdbufsiz;
;  501          1565  2    END;
;  502          1566  2
;  503          1567  2 !
;  504          1568  2 ! Delete trailing continuation character.
;  505          1569  2 !
;  506          1570  2 IF .ptr [0] EQL %C'-'
;  507          1571  3 THEN BEGIN
;  508          1572  3      flags [continue] = true;
;  509          1573  3      ptr = .ptr - 1;
;  510          1574  3      IF UNDERFLOW (.ptr)
;  511          1575  3         THEN ptr = .ptr + prc_c_cmdbufsiz;
;  512          1576  2      END;
;  513          1577  2
;  514          1578  2 !
;  515          1579  2 ! Delete trailing white space.
;  516          1580  2 !
;  517          1581  3 WHILE ((.ptr [0] EQL %X'20') OR (.ptr [0] EQL %X'09'))
;  518          1582  3 DO BEGIN
;  519          1583  3    flags [blank] = true;
;  520          1584  3    ptr = .ptr - 1;
;  521          1585  3    IF UNDERFLOW (.ptr)
;  522          1586  3       THEN ptr = .ptr + prc_c_cmdbufsiz;
;  523          1587  2    END;
;  524          1588  2
;  525          1589  2 !
;  526          1590  2 ! Insert a space at the end.
;  527          1591  2 !
;  528          1592  2 ptr = .ptr + 1;
;  529          1593  2 IF OVERFLOW (.ptr)
;  530          1594  2    THEN ptr = prc [prc_g_commands];
;  531          1595  2 IF (.flags [continue] AND .flags [blank]) OR NOT .flags [continue]
;  532          1596  3    THEN BEGIN
;  533          1597  3         ptr [0] = %X'20';
;  534          1598  3         ptr = .ptr + 1;
;  535          1599  3         IF OVERFLOW (.ptr)
;  536          1600  3            THEN ptr = prc [prc_g_commands];
;  537          1601  2         END;
;  538          1602  2
;  539          1603  2 RETURN true;
```

```
;  540          1604  1 END;


                                000C 00000 EDIT_COMMAND:
                                           .WORD   Save R2,R3                      ; 1490
                            51  94 00002            CLRB    FLAGS                  ; 1530
                        53  04  BC 9A 00004         MOVZBL  @LEN, R3               ; 1535
                        52  0534 CB 9E 00008        MOVAB   1332(R11), R2          ; 1546
                            50  D4 0000D            CLRL    I                      ;
                            1D  11 0000F            BRB     4$                     ;
                        22      69 91 00011 1$:     CMPB    (PTR), #34             ; 1538
                            03  12 00014            BNEQ    2$                     ;
                        51      04 8C 00016         XORB2   #4, FLAGS              ; 1539
                    05  51      02 E0 00019 2$:     BBS     #2, FLAGS, 3$          ; 1541
                        21      69 91 0001D         CMPB    (PTR), #33             ; 1542
                            10  13 00020            BEQL    5$                     ;
                            59  D6 00022 3$:        INCL    PTR                    ; 1545
                        52      59 D1 00024         CMPL    PTR, R2                ; 1546
                            05  1F 00027            BLSSU   4$                     ;
                        59  0133 CB 9E 00029        MOVAB   307(R11), PTR          ; 1547
                    DF  50      53 F3 0002E 4$:     AOBLEQ  R3, I, 1$              ; 1535
                            59  D7 00032 5$:        DECL    PTR                    ; 1553
                        50  0133 CB 9E 00034        MOVAB   307(R11), R0           ; 1554
                        50      59 D1 00039 6$:     CMPL    PTR, R0                ;
                            05  1E 0003C            BGEQU   7$                     ;
                        59  0401 C9 9E 0003E        MOVAB   1025(R9), PTR          ; 1555
                        20      69 91 00043 7$:     CMPB    (PTR), #32             ; 1560
                            05  13 00046            BEQL    8$                     ;
                        09      69 91 00048         CMPB    (PTR), #9              ;
                            04  12 0004B            BNEQ    9$                     ;
                            59  D7 0004D 8$:        DECL    PTR                    ; 1562
                            E8  11 0004F            BRB     6$                     ; 1563
                        2D      69 91 00051 9$:     CMPB    (PTR), #45             ; 1570
                            0F  12 00054            BNEQ    11$                    ;
                        51      01 88 00056         BISB2   #1, FLAGS              ; 1572
                            59  D7 00059 10$:       DECL    PTR                    ; 1573
                        50      59 D1 0005B         CMPL    PTR, R0                ; 1574
                            05  1E 0005E            BGEQU   11$                    ;
                        59  0401 C9 9E 00060        MOVAB   1025(R9), PTR          ; 1575
                        20      69 91 00065 11$:    CMPB    (PTR), #32             ; 1581
                            05  13 00068            BEQL    12$                    ;
                        09      69 91 0006A         CMPB    (PTR), #9              ;
                            05  12 0006D            BNEQ    13$                    ;
                        51      02 88 0006F 12$:    BISB2   #2, FLAGS              ; 1583
                            E5  11 00072            BRB     10$                    ; 1584
                            59  D6 00074 13$:       INCL    PTR                    ; 1592
                        52      59 D1 00076         CMPL    PTR, R2                ; 1593
                            03  1F 00079            BLSSU   14$                    ;
                        59      50 D0 0007B         MOVL    R0, PTR                ; 1594
                        07      51 E9 0007E 14$:    BLBC    FLAGS, 15$             ; 1595
                    03  51      01 E0 00081         BBS     #1, FLAGS, 15$         ;
                        08      51 E8 00085         BLBS    FLAGS, 16$             ;
                        89      20 90 00088 15$:    MOVB    #32, (PTR)+            ; 1597
                        52      59 D1 0008B         CMPL    PTR, R2                ; 1599
```

```
                03  1F 0008E           BLSSU   16$                                 ;
            59  50  D0 00090           MOVL    R0, PTR                             ; 1600
            50  01  D0 00093 16$:       MOVL    #1, R0                              ; 1603
                    04 00096           RET                                         ; 1604

; Routine Size:  151 bytes,    Routine Base:  DCL$ZCODE + 01C6
```

```
;  542          1605  1 GLOBAL ROUTINE dcl$get_prev_command (desc) : common_linkage =
;  543          1606  1
;  544          1607  1 !---
;  545          1608  1 !       Get the previous command from the command buffer and put it into
;  546          1609  1 !       the input buffer.
;  547          1610  1 !
;  548          1611  1 ! Inputs:
;  549          1612  1 !
;  550          1613  1 !       desc = address of descriptor in which to return recalled command
;  551          1614  1 !       R10 = address of WRK data structure
;  552          1615  1 !       R11 = address of PRC data structure
;  553          1616  1 !       WRK_L_RECALLPTR = pointer to last recalled command
;  554          1617  1 !
;  555          1618  1 ! Outputs:
;  556          1619  1 !
;  557          1620  1 !       The command is copied into the input buffer and the descriptor
;  558          1621  1 !       is initialized.
;  559          1622  1 !
;  560          1623  1 !       routine value = true if success, false if empty buffer
;  561          1624  1 !---
;  562          1625  1
;  563          1626  2 BEGIN
;  564          1627  2
;  565          1628  2 GLOBAL REGISTER
;  566          1629  2     ptr=9 :     REF VECTOR[,BYTE];      ! Pointer to retrieved command
;  567          1630  2
;  568          1631  2 EXTERNAL REGISTER
;  569          1632  2     wrk=10 :    REF $BBLOCK,            ! Address of WRK data structure
;  570          1633  2     prc=11 :    REF $BBLOCK;            ! Address of PRC data structure
;  571          1634  2
;  572          1635  2 !
;  573          1636  2 ! Back up one command.
;  574          1637  2 !
;  575          1638  2 ptr = .wrk [wrk_l_recallptr] - 1;
;  576          1639  2 IF UNDERFLOW (.ptr)
;  577          1640  2     THEN ptr = .ptr + prc_c_cmdbufsiz;
;  578          1641  2 IF .ptr [0] EQL 0 THEN RETURN false;
;  579          1642  2 ptr = .ptr - .ptr [0] - 2;
;  580          1643  2 IF UNDERFLOW (.ptr)
;  581          1644  2     THEN ptr = .ptr + prc_c_cmdbufsiz;
;  582          1645  2 wrk [wrk_l_recallptr] = .ptr;
;  583          1646  2
;  584          1647  2 !
;  585          1648  2 ! Now return the current command.
;  586          1649  2 !
;  587          1650  2 RETURN dcl$get_curr_command (.desc);
;  588          1651  1 END;


                                    0200 00000          .ENTRY  DCL$GET_PREV_COMMAND, Save R9                   ; 1605
                59      EA   AA                 01   C3 00002  SUBL3   #1, -22(WRK), PTR                        ; 1638
                             51      0133       CB   9E 00007  MOVAB   307(R11), R1                            ; 1639
                             51                 59   D1 0000C  CMPL    PTR, R1                                 ;
                                                05   1E 0000F  BGEQU   1$                                      ;
```

```
                        59     0401    C9  9E 00011          MOVAB   1025(R9), PTR                    ; 1640
                                       69  95 00016 1$:      TSTB    (PTR)                           ; 1641
                                       24  13 00018          BEQL    3$
                        50             69  9A 0001A          MOVZBL  (PTR), R0                       ; 1642
                50      59             50  C3 0001D          SUBL3   R0, PTR, R0
                        59       FE    A0  9E 00021          MOVAB   -2(R0), PTR
                        51             59  D1 00025          CMPL    PTR, R1                         ; 1643
                                       05  1E 00028          BGEQU   2$
                        59     0401    C9  9E 0002A          MOVAB   1025(R9), PTR                   ; 1644
                   EA   AA             59  D0 0002F 2$:      MOVL    PTR, -22(WRK)                   ; 1645
                        04             AC  DD 00033          PUSHL   DESC                            ; 1650
          00000000V     EF             01  FB 00036          CALLS   #1, DCL$GET_CURR_COMMAND
                                           04 0003D          RET
                                       50  D4 0003E 3$:      CLRL    R0                              ; 1651
                                           04 00040          RET

; Routine Size:  65 bytes,    Routine Base:  DCL$ZCODE + 025D
```

```
;  590          1652  1 GLOBAL ROUTINE dcl$get_next_command (desc) : common_linkage =
;  591          1653  1
;  592          1654  1 !---
;  593          1655  1 !       Get the next command from the command buffer and put it into
;  594          1656  1 !       the input buffer.
;  595          1657  1 !
;  596          1658  1 ! Inputs:
;  59           1659  1 !
;  598          1660  1 !       desc = address of descriptor in which to return recalled command
;  599          1661  1 !       R10 = address of WRK data structure
;  600          1662  1 !       R11 = address of PRC data structure
;  601          1663  1 !       WRK_L_RECALLPTR = pointer to last recalled command
;  602          1664  1 !
;  603          1665  1 ! Outputs:
;  604          1666  1 !
;  605          1667  1 !       The command is copied into the input buffer and the descriptor
;  606          1668  1 !       is initialized.
;  607          1669  1 !
;  608          1670  1 !       routine value = true if success, false if empty buffer
;  609          1671  1 !---
;  610          1672  1
;  611          1673  2 BEGIN
;  612          1674  2
;  613          1675  2 GLOBAL REGISTER
;  614          1676  2     ptr=9 : REF VECTOR[,BYTE];          ! Pointer to retrieved command
;  615          1677  2
;  616          1678  2 EXTERNAL REGISTER
;  617          1679  2     wrk=10 :    REF $BBLOCK,            ! Address of WRK data structure
;  618          1680  2     prc=11 :    REF $BBLOCK;            ! Address of PRC data structure
;  619          1681  2
;  620          1682  2 !
;  621          1683  2 ! Skip past the current command.
;  622          1684  2 !
;  623          1685  2 ptr = .wrk [wrk_l_recallptr] + 1;
;  624          1686  2 IF OVERFLOW (.ptr)
;  625          1687  2     THEN ptr = prc [prc_q_commands];
;  626          1688  2 IF .ptr [0] EQL 0 THEN RETURN false;
;  627          1689  2 ptr = .ptr + .ptr [0] + 2;
;  628          1690  2 IF OVERFLOW (.ptr)
;  629          1691  2     THEN ptr = .ptr - prc_c_cmdbufsiz;
;  630          1692  2 wrk [wrk_l_recallptr] = .ptr;
;  631          1693  2
;  632          1694  2 !
;  633          1695  2 ! Now return the current command.
;  634          1696  2 !
;  635          1697  2 RETURN dcl$get_curr_command (.desc);
;  636          1698  1 END;


                                        0200 00000          .ENTRY  DCL$GET_NEXT_COMMAND, Save R9                   ; 1652
                59      EA   AA         01   C1 00002          ADDL3   #1, -22(WRK), PTR                               ; 1685
                             51    0534  CB   9E 00007          MOVAB   1332(R11), R1                                   ; 1686
                             51         59   D1 0000C          CMPL    PTR, R1                                         ;
                                        05   1F 0000F          BLSSU   1$                                              ;
```

```
                    59     0133   CB  9E 00011          MOVAB   307(R11), PTR                    ; 1687
                    50            69  9A 00016 1$:      MOVZBL  (PTR), R0                       ; 1688
                                  1E  13 00019          BEQL    3$                              ;
                    59       02 A049  9E 0001B          MOVAB   2(R0)[PTR], PTR                 ; 1689
                    51            59  D1 00020          CMPL    PTR, R1                         ; 1690
                                  05  1F 00023          BLSSU   2$                              ;
                    59     FBFF   C9  9E 00025          MOVAB   -1025(R9), PTR                  ; 1691
                EA  AA            59  D0 0002A 2$:      MOVL    PTR, -22(WRK)                   ; 1692
                              04  AC  DD 0002E          PUSHL   DESC                            ; 1697
00000000V       EF                01  FB 00031          CALLS   #1, DCL$GET_CURR_COMMAND        ;
                                      04 00038          RET                                     ;
                                  50  D4 00039 3$:      CLRL    R0                              ; 1698
                                      04 0003B          RET                                     ;

; Routine Size:  60 bytes,    Routine Base:  DCL$ZCODE + 029E
```

```
;  638          1699  1 GLOBAL ROUTINE dcl$get_curr_command (desc) : common_linkage =
;  639          1700  1
;  640          1701  1 !---
;  641          1702  1 !       Get the current command from the command buffer and put it into
;  642          1703  1 !       the input buffer.
;  643          1704  1 !
;  644          1705  1 ! Inputs:
;  645          1706  1 !
;  646          1707  1 !       desc = address of descriptor in which to return recalled command
;  647          1708  1 !       R10 = address of WRK data structure
;  648          1709  1 !       R11 = address of PRC data structure
;  649          1710  1 !       WRK_L_RECALLPTR = pointer to last recalled command
;  650          1711  1 !
;  651          1712  1 ! Outputs:
;  652          1713  1 !
;  653          1714  1 !       The command is copied into the input buffer and the descriptor
;  654          1715  1 !       is initialized.
;  655          1716  1 !
;  656          1717  1 !       routine value = true if success, false if empty buffer
;  657          1718  1 !---
;  658          1719  1
;  659          1720  2 BEGIN
;  660          1721  2
;  661          1722  2 MAP
;  662          1723  2     desc :      REF VECTOR;             ! Command descriptor
;  663          1724  2
;  664          1725  2 GLOBAL REGISTER
;  665          1726  2     ptr=9 :     REF VECTOR[,BYTE];      ! Pointer to retrieved command
;  666          1727  2
;  667          1728  2 EXTERNAL REGISTER
;  668          1729  2     wrk=10 :    REF $BBLOCK,            ! Address of WRK data structure
;  669          1730  2     prc=11 :    REF $BBLOCK;            ! Address of PRC data structure
;  670          1731  2
;  671          1732  2 !
;  672          1733  2 ! Init the output descriptor.
;  673          1734  2 !
;  674          1735  2 ptr = .wrk [wrk_l_recallptr] + 1;
;  675          1736  2 IF OVERFLOW (.ptr)
;  676          1737  2     THEN ptr = .ptr - prc_c_cmdbufsiz;
;  677          1738  2 desc [0] = .ptr [0];
;  678          1739  2 desc [1] = wrk [wrk_g_inpbuf] - 2;
;  679          1740  2
;  680          1741  2 !
;  681          1742  2 ! Find the start of the command string.
;  682          1743  2 !
;  683          1744  2 IF .ptr [0] EQL 0 THEN RETURN false;
;  684          1745  2 ptr = .ptr + 1;
;  685          1746  2 IF OVERFLOW (.ptr)
;  686          1747  2     THEN ptr = prc [prc_g_commands];
;  687          1748  2
;  688          1749  2 !
;  689          1750  2 ! Copy the command text into the input buffer.
;  690          1751  2 !
;  691          1752  2 IF OVERFLOW (.ptr + .desc [0] - 1)                    ! Wrap around?
;  692          1753  3     THEN BEGIN                                         ! Yes, then copy in two pieces
;  693          1754  3          LOCAL temp_len;
;  694          1755  3          temp_len = prc [prc_g_commands]               ! Get length of first piece
```

```
;  695          1756  3                    + prc_c_cmdbufsiz - .ptr;                          !
;  696          1757  3            CH$MOVE (.temp_len, .ptr, .desc [1]);                      ! Move the first piece
;  697          1758  3            CH$MOVE (.desc [0] - .temp_len,                            ! Move the second piece
;  698          1759  3                     prc [prc_g_commands],                             !
;  699          1760  3                    .desc [1] + .temp_len);                            !
;  700          1761  3            END
;  701          1762  2        ELSE CH$MOVE (.desc [0], .ptr, .desc [1]);                     ! No, then copy in whole
;  702          1763  2
;  703          1764  2    RETURN true;
;  704          1765  1    END;


                                                03FC 00000          .ENTRY  DCL$GET_CURR_COMMAND, Save R2,R3,R4,R5,R6,- ; 1699
                                                                            R7,R8,R9
                59       EA   AA                01   C1 00002          ADDL3   #1, -22(WRK), PTR                           ; 1735
                              51       0534     CB   9E 00007          MOVAB   1332(R11), R1                               ; 1736
                              51                59   D1 0000C          CMPL    PTR, R1                                     ;
                                                05   1F 0000F          BLSSU   1$                                          ;
                              59       FBFF     C9   9E 00011          MOVAB   -1025(R9), PTR                              ; 1737
                              56         04     AC   D0 00016 1$:      MOVL    DESC, R6                                    ; 1738
                              66                69   9A 0001A          MOVZBL  (PTR), (R6)                                 ;
                        04    A6       F894     CA   9E 0001D          MOVAB   -1900(R10), 4(R6)                           ; 1739
                                               69   95 00023          TSTB    (PTR)                                       ; 1744
                                               38   13 00025          BEQL    5$                                          ;
                                               59   D6 00027          INCL    PTR                                         ; 1745
                              51               59   D1 00029          CMPL    PTR, R1                                     ; 1746
                                               05   1F 0002C          BLSSU   2$                                          ;
                              59       0133     CB   9E 0002E          MOVAB   307(R11), PTR                               ; 1747
                              58                66   D0 00033 2$:      MOVL    (R6), R8                                    ; 1752
                              50          FF A849   9E 00036          MOVAB   -1(R8)[PTR], R0                             ;
                              51                50   D1 0003B          CMPL    R0, R1                                      ;
                                                16   1F 0003E          BLSSU   3$                                          ;
                     57                         51   59 C3 00040          SUBL3   PTR, R1, TEMP_LEN                           ; 1756
                04   B6                        69   57 28 00044          MOVC3   TEMP_LEN, (PTR), @4(R6)                     ; 1757
                                               58   57 C2 00049          SUBL2   TEMP_LEN, R8                                ; 1758
                04 B647       0133     CB         58 28 0004C          MOVC3   R8, 307(PRC), @4(R6)[TEMP_LEN]              ; 1760
                                               05   11 00054          BRB     4$                                          ; 1752
                04   B6                        69   58 28 00056 3$:      MOVC3   R8, (PTR), @4(R6)                           ; 1762
                                               50   01 D0 0005B 4$:      MOVL    #1, R0                                      ; 1764
                                                    04 0005E          RET                                                 ;
                                               50   D4 0005F 5$:      CLRL    R0                                          ; 1765
                                                    04 00061          RET                                                 ;

; Routine Size:  98 bytes,    Routine Base:  DCL$ZCODE + 02DA                                                                ;
```

```
;  706          1766  1 END
;  707          1767  0 ELUDOM


;                                  PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  DCL$ZCODE                     828  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(0)


;                                Library Statistics
;                                          -------- Symbols --------      Pages       Processing
;       File                               Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1            18619         5         0        1000        00:01.8


;                                     COMMAND QUALIFIERS

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

; Size:          828 code + 0 data bytes
; Run Time:          00:28.3
; Elapsed Time:      01:35.5
; Lines/CPU Min:      3751
; Lexemes/CPU-Min:  33038
; Memory Used:   207 pages
; Compilation Complete
```

RPCLINT
LIS

RECALLSUB
LIS

MESSAGE
LIS

READREC
LIS

PARSENT
LIS

ON
LIS