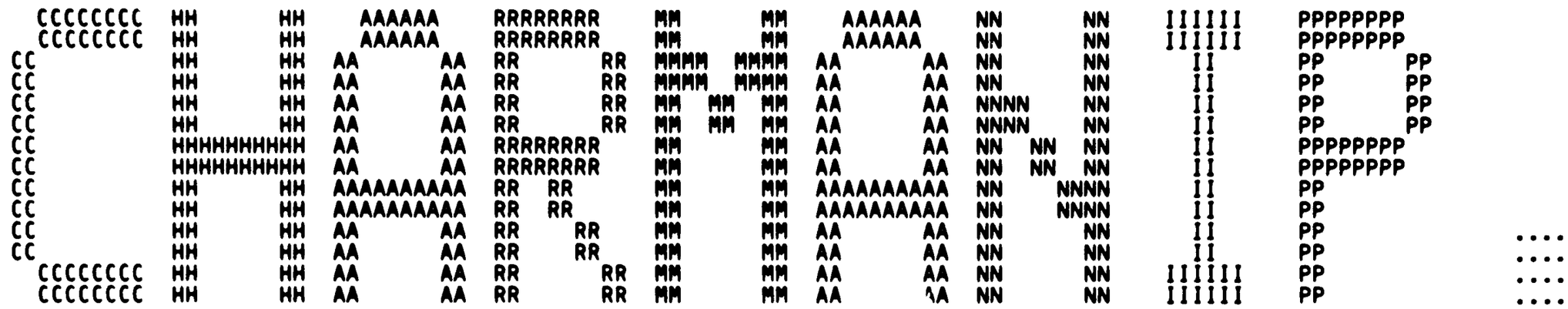
```
DDDDDDDDDDD         CCCCCCCCCCC  LLL
DDDDDDDDDDDD        CCCCCCCCCCCC LLL
DDDDDDDDDDDD        CCCCCCCCCCCC LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDD          DDD  CCC           LLL
DDD  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

**FILE**ID**CHARMANIP



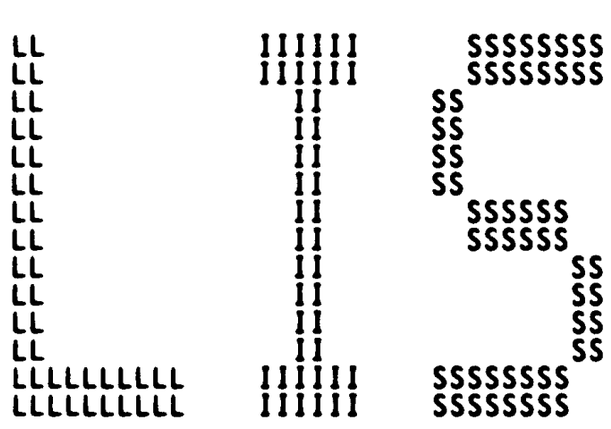



Table of contents

```
0000     1          .TITLE  CHARMANIP - CHARACTER MANIPULATION ROUTINES
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 ; CHARACTER MANIPULATION ROUTINES
0000    29 ;
0000    30 ; D. N. CUTLER 28-MAR-77
0000    31 ;
0000    32 ; MODIFIED BY:
0000    33 ;
0000    34 ;       V03-008 PCG0008         Peter George    15-Jul-1983
0000    35 ;               Check for maximum token size.
0000    36 ;               Modify upcase algorithm to allow multinational chars.
0000    37 ;               Move "+" back to insignificant blank separators.
0000    38 ;
0000    39 ;       V03-007 PCG0007         Peter George    15-Jun-1983
0000    40 ;               Make "+-(" significant blank separators.
0000    41 ;
0000    42 ;       V03-006 PCG0006         Peter George    30-Apr-1983
0000    43 ;               Change WRK_L_PARMCNT to WRK_B_PARMCNT.
0000    44 ;
0000    45 ;       V03-005 PCG0005         Peter George    24-Feb-1983
0000    46 ;               Set PTR_B_NUMBER, PTR_B_PARMCNT, and PTR_L_ENTITY
0000    47 ;               when creating a PTR block.
0000    48 ;
0000    49 ;       V03-004 PCG0004         Peter George    28-Oct-1982
0000    50 ;               Remove ASCIC user prompt.  Read escape sequences.
0000    51 ;
0000    52 ;       V03-003 PCG0003         Peter George    15-Oct-1982
0000    53 ;               Do not supply a prompt string to the user input
0000    54 ;               routine.
0000    55 ;
0000    56 ;       V03-002 PCG0002         Peter George    19-Jul-1982
0000    57 ;               Support CLI$DCL_PARSE continuation routines.
```

```
0000     58 ;                Add value level support to DCL$GENDESCR.
0000     59 ;                Set continuation flag for reprompt I/O.
0000     60 ;
0000     61 ;        V03-001 PCG0001         Peter George    09-Jun-1982
0000     62 ;                Do not perform indirection when looking ahead with
0000     63 ;                DCL$TESTBLANK.
0000     64 ;---
```

```
                        0000     66 ;
                        0000     67 ; MACRO LIBRARY CALLS
                        0000     68 ;
                        0000     69
                        0000     70          PRCDEF                          ;DEFINE PROCESS WORK AREA
                        0000     71          WRKDEF                          ;DEFINE COMMAND WORK AREA
                        0000     72          PTRDEF                          ;DEFINE RESULT PARSE DESCRIPTOR FORMAT
                        0000     73          $CLIMSGDEF                      ;DEFINE ERROR/STATUS VALUES
                        0000     74
                        0000     75 ;
                        0000     76 ; LOCAL DATA
                        0000     77 ;
                        0000     78
                    00000000     79          .PSECT  DCL$ZCODE,BYTE,RD,NOWRT
                        0000     80
                        0000     81 DCL$HYPHEN::                             ;HYPHEN STRING
                  00 2D 0000     82          .ASCII  /-/<0>                  ;
                        0002     83 TERMCLASS:                               ;ITEM TERMINATOR STRING
            20 3A 2F 2B 2C 29   0002     84          .ASCII  \),+/: \                ;
                        0008     85 TERMTAB:                                 ;TERMINATOR TABLE
00 23 5E 3D 3F 3E 5D 29 2B 2C 2F 0008     86          .ASCII  \/,+)]>?=^#\<0>         ;
                        0013     87 BLANKTAB:                                ;SIGNIFICANT BLANK SEPARATOR
21 20 3B 2E 3C 5B 28 27 2D 2A 40 3A 0013     88          .ASCII  \:@*-'([<.; !   \       ;
                  09 001F
                        0020     89 ENDTERM:                                 ;REF LABEL
```

```
                     0020     91          .SBTTL  LOCATE STRING BY INDEX NUMBER
                     0020     92 ;+
                     0020     93 ; DCL$LOCATE - LOCATE STRING BY INDEX NUMBER
                     0020     94 ;
                     0020     95 ; THIS ROUTINE IS CALLED TO LOCATE A STRING IN A CHARACTER ARRAY BY COUNTING
                     0020     96 ; THE OCCURANCES OF CHARACTERS WITH BIT 7 SET.
                     0020     97 ;
                     0020     98 ; INPUTS:
                     0020     99 ;
                     0020    100 ;        R2 = ADDRESS OF CHARACTER ARRAY.
                     0020    101 ;        R3 = STRING INDEX NUMBER.
                     0020    102 ;
                     0020    103 ; OUTPUTS:
                     0020    104 ;
                     0020    105 ;        R1 = LENGTH OF STRING.
                     0020    106 ;        R2 = ADDRESS OF STRING.
                     0020    107 ;-
                     0020    108
                     0020    109 DCL$LOCATE::                                ;LOCATE STRING BY INDEX NUMBER
     51    D4        0020    110          CLRL    R1                         ;CLEAR INITIAL LENGTH
52   51    C0        0022    111 10$:     ADDL    R1,R2                      ;POINT TO NEXT ENTRY IN TABLE
51    82   9A        0025    112          MOVZBL  (R2)+,R1                   ;GET COUNT
  F7 53    F4        0028    113          SOBGEQ  R3,10$                     ;ANY MORE STRINGS TO SKIP OVER?
           05        002B    114          RSB                                ;
```

```
                                002C   116          .SBTTL  GENERATE RESULT PARSE DESCRIPTOR
                                002C   117 ;+
                                002C   118 ; DCL$GENDESCR - GENERATE RESULT PARSE DESCRIPTOR
                                002C   119 ;
                                002C   120 ; THIS ROUTINE IS CALLED TO GENERATE A LONGWORD DESCRIPTOR FOR A RESULT PARSE
                                002C   121 ; ITEM.
                                002C   122 ;
                                002C   123 ; INPUTS:
                                002C   124 ;
                                002C   125 ;       R4 = ITEM NUMBER.
                                002C   126 ;       R5 = ITEM TYPE.
                                002C   127 ;       R6 = ITEM FLAGS.
                                002C   128 ;       R7 = ITEM VALUE.
                                002C   129 ;       R8 = STARTING ADDRESS OF ITEM.
                                002C   130 ;       R9 = ITEM ENTITY BLOCK ADDRESS.
                                002C   131 ;       WRK_L_EXPANDPTR = ADDRESS OF TERMINATOR CHARACTER IN EXPANSION BUFFER.
                                002C   132 ;       WRK_B_VALLEV = VALUE DEPTH OF DESCRIBED ITEM.
                                002C   133 ;       WRK_B_PARMCNT = NUMBER OF LAST PARAMETER PARSED.
                                002C   134 ;
                                002C   135 ; OUTPUTS:
                                002C   136 ;
                                002C   137 ;       THE SPECIFIED FIELDS ARE PACKED TOGE   ER TO FORM A RESULT PARSE
                                002C   138 ;       DESCRIPTOR LONGWORD AND THE RESULT     VALUE IS STORED IN THE RESULT
                                002C   139 ;       PARSE TABLE.
                                002C   140 ;-
                                002C   141
                                002C   142 DCL$GENDESCR::                                ;GENERATE RESULT PARSE DESCRIPTOR
          51    BA AA    D0     002C   143          MOVL    WRK_L_RSLNXT(R10),R1          ;GET ADDRESS OF NEXT FREE DESCRIPTOR
             B6 AA    9F     0030   144          PUSHAB  WRK_G_RESULT+WRK_C_RSLBUFSIZ(R10) ;GET ENDING ADDRESS OF RESULT BUFF
             51    8E    D1     0033   145          CMPL    (SP)+,R1                      ;ROOM FOR ANOTHER DESCRIPTOR?
                   0A    1A     0036   146          BGTRU   20$                           ;BRANCH IF OK
                         0038   147          STATUS  RSLOVF                        ;SET ERROR STATUS
                02D3     31     003F   148 10$:     BRW     ERROR                         ;
                         0042   149 20$:     STATUS  TKNOVF                        ;SET ERROR STATUS
          57    000000FF 8F     D1     0049   150          CMPL    #255,R7               ;CHECK SIZE OF TOKEN
                         ED     1F     0050   151          BLSSU   10$                   ;BRANCH IF TOO LARGE
             08    00    57     F0     0052   152          INSV    R7,#PTR_V_VALUE,#PTR_S_VALUE,- ;INSERT VALUE FIELD
                         61     0056   153                  PTR_L_DESCR(R1)               ;
                F492     CA     9F     0057   154          PUSHAB  WRK_G_BUFFER(R10)     ;GET ADDRESS OF EXPANSION BUFFER
             50    58    8E     C3     005B   155          SUBL3   (SP)+,R8,R0           ;CALCULATE OFFSET TO ITEM
                   08    50     F0     005F   156          INSV    R0,#PTR_V_OFFSET,-    ;INSERT OFFSET TO ITEM
                   61    0C     0062   157                  #PTR_S_OFFSET,PTR_L_DESCR(R1) ;
             04    14    56     F0     0064   158          INSV    R6,#PTR_V_FLAGS,#PTR_S_  AGS,- ;INSERT ITEM FLAGS
                         61     0068   159                  PTR_L_DESCR(R1)               ;
             04    1C    55     F0     0069   160          INSV    R5,#PTR_V_TYPE,#PTR_S_TYPE,- ;INSERT ITEM TYPE
                         61     006D   161                  PTR_L_DESCR(R1)               ;
               05 A1    54     90     006E   162          MOVB    R4,PTR_B_NUMBER(R1)   ;INSERT ITEM NUMBER
               08 A1    59     D0     0072   163          MOVL    R9,PTR_L_ENTITY(R1)   ;INSERT ENTITY BLOCK ADDRESS
                     CE AA    90     0076   164          MOVB    WRK_B_PARMCNT(R10),-  ;INSERT LAST PARAMETER NUMBER
                   06 A1            0079   165                  PTR_B_PARMCNT(R1)     ;
                     C4 AA    90     007B   166          MOVB    WRK_B_VALLEV(R10),-   ;INSERT VALUE DEPTH
                   04 A1            007E   167                  PTR_B_LEVEL(R1)       ;
                         0080   168
             BA AA    0C     C0     0080   169          ADDL    #PTR_C_LENGTH,WRK_L_RSLNXT(R10) ;POINT TO NEXT ITEM IN BUFFER
                   BA AA    D0     0084   170          MOVL    WRK_L_RSLNXT(R10),-   ;REMEMBER LAST TOKEN
                   B6 AA            0087   171                  WRK_L_RSLEND(R10)     ;
             50    F486     DA     9A     0089   172          MOVZBL  @WRK_L_EXPANDPTR(R10),R0 ;GET TERMINATOR CHARACTER
```

```
                                008E   174          .SBTTL  SET TERMINATOR CLASS IN LAST DESCRIPTOR
                                008E   175 ;+
                                008E   176 ; DCL$GENTERM - SET TERMINATOR CLASS IN PREVIOUS TOKEN DESCRIPTOR
                                008E   177 ;
                                008E   178 ; THIS ROUTINE IS CALLED TO SET THE TERMINATOR CLASS IN THE TOKEN
                                008E   179 ; DESCRIPTOR MOST RECENTLY STORED VIA DCL$GENDESCR.
                                008E   180 ;
                                008E   181 ; INPUTS:
                                008E   182 ;
                                008E   183 ;       R0 = TERMINATOR CHARACTER
                                008E   184 ;
                                008E   185 ; OUTPUTS:
                                008E   186 ;
                                008E   187 ;       NONE
                                008E   188 ;-
                                008E   189
                                008E   190 DCL$GENTERM::
                  3D    50  91  008E   191          CMPB    R0,#^A'='               ;EQUAL TERMINATOR?
                        11  13  0091   192          BEQL    20$                     ;IF YES, SET CLASS=COLON
                  28    50  91  0093   193          CMPB    R0,#^A'('               ;LEFT PARENTHESIS?
                        11  13  0096   194          BEQL    30$                     ;IF YES, SET CLASS
              FF64 CF   06  50  3A  0098   195          LOCC    R0,#TERMTAB-TERMCLASS,TERMCLASS ;SEARCH FOR TERMINATOR IN TABLE
                        0C  12  009E   196          BNEQ    50$                     ;IF FOUND - USE INDEX # AS CLASS
                                00A0   197          ASSUME  PTR_K_BLANK EQ 1
                        50  D6  00A0   198          INCL    R0                      ;SET TERMINATOR CLASS TO BLANK
                        08  11  00A2   199          BRB     50$
                  50    02  9A  00A4   200 20$:     MOVZBL  #PTR_K_COLON,R0         ;SET TERMINATOR CLASS TO COLON
                        03  11  00A7   201          BRB     50$
                  50    07  9A  00A9   202 30$:     MOVZBL  #PTR_K_LPAREN,R0        ;SET TERMINATOR CLASS TO LPAREN
            51    BA  AA  D0  00AC   203 50$:     MOVL    WRK_C_RSLNXT(R10),R1    ;GET ADDRESS OF NEXT FREE DESCRIPTOR
      04    18    50  F0  00B0   204          INSV    R0,#PTR_V_TERM,#PTR_S_TERM,- ;INSERT TERMINATOR CLASS NUMBER
              F4  A1      00B4   205                  -PTR_C_LENGTH(R1)       ;IN LAST TOKEN DESCRIPTOR STORED
                        05  00B6   206          RSB
```

```
                     00B7   208          .SBTTL  COMPRESS QUOTED STRING IN EXPANSION BUFFER
                     00B7   209 ;+
                     00B7   210 ; DCL$COMPRESS - COMPRESS QUOTED STRING IN EXPANSION BUFFER
                     00B7   211 ;
                     00B7   212 ; THIS ROUTINE IS CALLED TO COMPRESS A QUOTED STRING WHICH IS
                     00B7   213 ; THE LAST TOKEN IN THE EXPANSION BUFFER.  THE QUOTES ARE REMOVED
                     00B7   214 ; FROM THE STRING AND THE EXPANSION BUFFER POINTER IS RESET TO THE
                     00B7   215 ; NEW END OF THE STRING.
                     00B7   216 ;
                     00B7   217 ; INPUTS:
                     00B7   218 ;
                     00B7   219 ;       R1 = LENGTH OF STRING.
                     00B7   220 ;       R2 = ADDRESS OF STRING.
                     00B7   221 ;
                     00B7   222 ; OUTPUTS:
                     00B7   223 ;
                     00B7   224 ;       R1 = LENGTH OF COMPRESSED STRING.
                     00B7   225 ;       R2 = ADDRESS OF COMPRESSED STRING.
                     00B7   226 ;
                     00B7   227 ;       WRK_L_EXPANDPTR POINTS TO THE END OF THE COMPRESSED STRING.
                     00B7   228 ;
                     00B7   229 ;       R3,R4 DESTROYED.
                     00B7   230 ;-
                     00B7   231
                     00B7   232 DCL$COMPRESS::
           07    10  00B7   233          BSBB    DCL$COMPSTRING          ;COMPRESS STRING IN PLACE
F486 CA   52    51  C1 00B9   234          ADDL3   R1,R2,WRK_L_EXPANDPTR(R10) ;RESET POINTER JUST PAST NEW STRING
                    05 00BF   235          RSB
```

```
                     00C0   237          .SBTTL  COMPRESS QUOTED STRING IN-PLACE
                     00C0   238 ;+
                     00C0   239 ; DCL$COMPSTRING - COMPRESS QUOTED STRING IN PLACE
                     00C0   240 ;
                     00C0   241 ; THIS ROUTINE IS CALLED TO COMPRESS A QUOTED STRING.
                     00C0   242 ;
                     00C0   243 ; INPUTS:
                     00C0   244 ;
                     00C0   245 ;       R1 = LENGTH OF STRING.
                     00C0   246 ;       R2 = ADDRESS OF STRING.
                     00C0   247 ;
                     00C0   248 ; OUTPUTS:
                     00C0   249 ;
                     00C0   250 ;       THE SPECIFIED STRING IS COMPRESSED REMOVING PAIRED QUOTES.
                     00C0   251 ;
                     00C0   252 ;               R1 = LENGTH OF COMPRESSED STRING.
                     00C0   253 ;               R2 = ADDRESS OF COMPRESSED STRING.
                     00C0   254 ;
                     00C0   255 ;       R3,R4 DESTROYED.
                     00C0   256 ;-
                     00C0   257
                     00C0   258 DCL$COMPSTRING::                     ;COMPRESS QUOTED STRING
          53   52    D0  00C0   259          MOVL    R2,R3           ;COPY BASE ADDRESS OF STRING
          54   53    D0  00C3   260          MOVL    R3,R4           ;COPY BASE ADDRESS OF STRING
               50    D4  00C6   261          CLRL    R0              ;CLEAR QUOTATION FLAG
          63   22    91  00C8   262 10$:     CMPB    #^A/"/,(R3)     ;NEXT CHARACTER A QUOTE?
               12    12  00CB   263          BNEQ    20$             ;IF NEQ NO
               53    D6  00CD   264          INCL    R3              ;ADJUST PAST QUOTE
       0F 50   00    E3  00CF   265          BBCS    #0,R0,30$       ;IF CLR, JUST ENTERING QUOTE
               51    D7  00D3   266          DECL    R1              ;DECREMENT NUMBER OF CHARACTERS REMAINING
               0E    15  00D5   267          BLEQ    40$             ;IF LEQ NONE
          63   22    91  00D7   268          CMPB    #^A/"/,(R3)     ;NEXT CHARACTER ALSO A QUOTE?
               03    13  00DA   269          BEQL    20$             ;IF EQL YES
               50    01  CA 00DC   270          BICL    #1,R0           ;CLEAR QUOTE IN PROGRESS
               84    83  90 00DF   271 20$:     MOVB    (R3)+,(R4)+     ;COPY CHARACTER AND COMPRESS
                  E3 51  F5 00E2   272 30$:     SOBGTR  R1,10$          ;ANY MORE CHARACTERS TO SCAN?
          51   54    52  C3 00E5   273 40$:     SUBL3   R2,R4,R1        ;CALCULATE LENGTH OF COMPRESSED STRING
                         05 00E9   274          RSB
```

```
                        00EA   276          .SBTTL  MARK POSITION IN EXPANSION BUFFER
                        00EA   277 ;+
                        00EA   278 ; DCL$MARK - MARK CURRENT POSITION IN EXPANSION BUFFER
                        00EA   279 ;
                        00EA   280 ; MARK THE CURRENT POSITION OF THE SEGMENT BEING PARSED
                        00EA   281 ; SO THAT IF ANY ERROR OCCURS LATER, THE SEGMENT CAN BE
                        00EA   282 ; SHOWN IN THE ERROR MESSAGE.  THE MARKED POSITION IS
                        00EA   283 ; ALSO USED TO OBTAIN THE LENGTH AND ADDRESS OF THE STRING
                        00EA   284 ; MOVED INTO THE EXPANSION BUFFER SINCE THE LAST MARK.
                        00EA   285 ;
                        00EA   286 ; INPUTS:
                        00EA   287 ;
                        00EA   288 ;       WRK_L_EXPANDPTR = CURRENT POINTER INTO EXPANSION BUFFER
                        00EA   289 ;
                        00EA   290 ; OUTPUTS:
                        00EA   291 ;
                        00EA   292 ;       WRK_L_MARKPTR = POSITION OF SEGMENT NOW BEING PARSED
                        00EA   293 ;--
                        00EA   294 DCL$MARK::
F48A CA   F486 CA   D0  00EA   295          MOVL    WRK_L_EXPANDPTR(R10),WRK_L_MARKPTR(R10) ;COPY CURRENT EXPANSION BUFF
                    05  00F1   296          RSB
```

```
                                00F2    298          .SBTTL  GET DESCRIPTOR OF EXPANSION STRING
                                00F2    299 ;+
                                00F2    300 ; DCL$MARKEDTOKEN - GET DESCRIPTOR OF MARKED EXPANSION STRING
                                00F2    301 ;
                                00F2    302 ; THIS ROUTINE IS CALLED TO OBTAIN THE LENGTH AND ADDRESS OF THE
                                00F2    303 ; STRING COMPOSED OF ALL CHARACTERS WRITTEN INTO THE EXPANSION BUFFER
                                00F2    304 ; SINCE THE LAST CALL TO DCL$MARK.
                                00F2    305 ;
                                00F2    306 ; INPUTS:
                                00F2    307 ;
                                00F2    308 ;       R10 = ADDRESS OF COMMAND WORK AREA
                                00F2    309 ;
                                00F2    310 ; OUTPUTS:
                                00F2    311 ;
                                00F2    312 ;       R1 = LENGTH OF STRING
                                00F2    313 ;       R2 = ADDRESS OF STRING
                                00F2    314 ;       Z SET IF R1=0
                                00F2    315 ;-
                                00F2    316 DCL$MARKEDTOKEN::
     52   F48A CA   D0   00F2    317          MOVL    WRK_L_MARKPTR(R10),R2   ;GET ADDRESS OF STRING
51   F486 CA   52   C3   00F7    318          SUBL3   R2,WRK_L_EXPANDPTR(R10),R1 ;COMPUTE LENGTH OF STRING
                        05   00FD    319          RSB
```

```
                     00FE    321          .SBTTL  GET NEXT NONBLANK CHARACTER
                     00FE    322 ;+
                     00FE    323 ; DCL$GETNBLK - GET NEXT NONBLANK CHARACTER FROM INPUT BUFFER
                     00FE    324 ;
                     00FE    325 ; THIS ROUTINE IS CALLED TO OBTAIN THE NEXT NONBLANK CHARACTER FROM THE INPUT
                     00FE    326 ; BUFFER.
                     00FE    327 ;
                     00FE    328 ; INPUTS:
                     00FE    329 ;
                     00FE    330 ;       NONE.
                     00FE    331 ;
                     00FE    332 ; OUTPUTS:
                     00FE    333 ;
                     00FE    334 ;       CHARACTERS ARE OBTAINED FROM THE INPUT BUFFER UNTIL A NONBLANK
                     00FE    335 ;       CHARACTER IS ENCOUNTERED.
                     00FE    336 ;
                     00FE    337 ;       R0 = NONBLANK CHARACTER.
                     00FE    338 ;-
                     00FE    339
                     00FE    340 DCL$GETNBLK::                           ;GET NEXT NONBLANK CHARACTER
0082   30    00FE    341          BSBW    DCL$SETNBLK                    ;POINT TO NEXT NONBLANK CHARACTER
0117   31    0101    342          BRW     DCL$GETCHAR                    ;RETURN NEXT CHARACTER TO CALLER
```

```
                     0104    344          .SBTTL  MOVE CHARACTER TO EXPANSION BUFFER AND GET BLANK TOKEN
                     0104    345 ;+
                     0104    346 ; DCL$MOVBTOKN - MOVE CHARACTER TO EXPANSION BUFFER AND GET BLANK TOKEN
                     0104    347 ;
                     0104    348 ; THIS ROUTINE IS CALLED TO PERFORM THE COMBINED OPERATION OF MOVING THE NEXT
                     0104    349 ; CHARACTER TO THE EXPANSION BUFFER AND THEN OBTAINING THE NEXT BLANK TOKEN.
                     0104    350 ;
                     0104    351 ; INPUTS:
                     0104    352 ;
                     0104    353 ;       NONE.
                     0104    354 ;
                     0104    355 ; OUTPUTS:
                     0104    356 ;
                     0104    357 ;       THE NEXT CHARACTER IS MOVED TO THE EXPANSION BUFFER AND THEN THE NEXT
                     0104    358 ;       BLANK TOKEN IS OBTAINED.
                     0104    359 ;-
                     0104    360
                     0104    361 DCL$MOVBTOKN::                              ;MOVE CHARACTER AND GET TOKEN
00E8   30            0104    362          BSBW    DCL$MOVCHAR                ;MOVE CHARACTER TO EXPANSION BUFFER
  05   11            0107    363          BRB     DCL$GTBTOKEN               ;GET NEXT BLANK TOKEN
```

```
                     0109   365          .SBTTL  MOVE CHARACTER TO EXPANSION BUFFER AND GET TOKEN
                     0109   366 ;+
                     0109   367 ; DCL$MOVTOKN - MOVE CHARACTER TO EXPANSION BUFFER AND GET TOKEN
                     0109   368 ;
                     0109   369 ; THIS ROUTINE IS CALLED TO PERFORM THE COMBINED OPERATION OF MOVING THE NEXT
                     0109   370 ; CHARACTER TO THE EXPANSION BUFFER AND THEN OBTAINING THE NEXT TOKEN.
                     0109   371 ;
                     0109   372 ; INPUTS:
                     0109   373 ;
                     0109   374 ;       NONE.
                     0109   375 ;
                     0109   376 ; OUTPUTS:
                     0109   377 ;
                     0109   378 ;       THE NEXT CHARACTER IS MOVED TO THE EXPANSION BUFFER AND THEN THE NEXT
                     0109   379 ;       TOKEN IS OBTAINED.
                     0109   380 ;-
                     0109   381
                     0109   382 DCL$MOVTOKN::                          ;MOVE CHARACTER AND GET TOKEN
00E3   30            0109   383          BSBW    DCL$MOVCHAR            ;MOVE CHARACTER TO EXPANSION BUFFER
```

```
                                                                                .SBTTL  GETOKEN - GET NEXT TOKEN
                                                                                ;+
                                                                                ; DCL$GETOKEN - GET TOKEN FROM INPUT BUFFER WITH INSIGNIFICANT LEADING BLANKS
                                                                                ; DCL$GTBTOKEN - GET TOKEN FROM INPUT BUFFER WITH SIGNIFICANT LEADING BLANKS
                                                                                ;
                                                                                ; THIS ROUTINE IS CALLED TO SCAN THE INPUT BUFFER UNTIL A DELIMITER IS FOUND.
                                                                                ; WHILE THE BUFFER IS BEING SCANNED, THE RESULTANT TOKEN IS COPIED INTO THE
                                                                                ; EXPANSION BUFFER. WHEN A TERMINATOR IS RECOGNIZED, THE DESCRIPTOR FOR THE
                                                                                ; TOKEN IS RETURNED TO THE CALLER ALONG WITH THE TERMINATOR.
                                                                                ;
                                                                                ; INPUTS:
                                                                                ;
                                                                                ;       NONE.
                                                                                ;
                                                                                ; OUTPUTS:
                                                                                ;
                                                                                ;       R0 = TERMINATOR CHARACTER.
                                                                                ;       R1 = LENGTH OF OUTPUT TOKEN.
                                                                                ;       R2 = ADDRESS OF OUTPUT TOKEN.
                                                                                ;
                                                                                ;       Z = 1 IF NO TOKEN IS BEING RETURNED.
                                                                                ;       Z = 0 IF A TOKEN IS BEING RETURNED.
                                                                                ;-

                                                                                DCL$GETOKEN::                           ;GET TOKEN FROM INPUT BUFFER
                                                                                        BSBB    DCL$SETNBLK             ;POINT TO NEXT NONBLANK CHARACTER
                                                                                DCL$GTBTOKEN::                          ;GET TOKEN FROM INPUT BUFFER
                                                                                        SUBL3   R10,-                   ;FIND OFFSET OF FIRST BYTE IN TOKEN
                                                                                                WRK_L_EXPANDPTR(R10),R2 ;
                                                                                10$:    BSBW    DCL$MOVCHAR             ;MOVE CHARACTER TO EXPANSION BUFFER
                                                                                        BBS     #WRK_V_QUOTE,-          ;IF SET, QUOTE IN PROGRESS
                                                                                                WRK_B_FLAGS(R10),10$    ;
                                                                                        BBS     #WRK_V_STAR,-           ;IF SET, ASTERISK IS TERMINATOR
                                                                                                WRK_B_FLAGS(R10),20$    ;
                                                                                        CMPB    #^A'*',R0               ;ASTERISK?
                                                                                        BEQL    10$                     ;IF EQL YES
                                                                                        BBC     #WRK_V_ESCAPE,-         ;IF SET, NULL IS TERMINATOR
                                                                                                WRK_B_FLAGS(R10),20$    ;
                                                                                        CMPB    #^X00,R0                ;NULL?
                                                                                        BNEQ    10$                     ;NO, THEN LOOP
                                                                                20$:    LOCC    R0,#ENDTERM-TERMTAB,-   ;SEARCH FOR TERMINATOR
                                                                                                TERMTAB                 ;
                                                                                        BEQL    10$                     ;IF EQL TERMINATOR NOT FOUND
                                                                                        BICW    #WRK_M_ESCAPE,-         ;CLEAR ESCAPE FLAG
                                                                                                WRK_B_FLAGS(R10)        ;
                                                                                        DECL    WRK_L_EXPANDPTR(R10)    ;BACK UP TO TERMINATOR
                                                                                        MOVZBL  (R1),R0                 ;RETRIEVE TERMINATOR
                                                                                        CMPB    #^A' ',R0               ;WAS TERMINATOR A BLANK?
                                                                                        BNEQ    30$                     ;IF NEQ NO
                                                                                        MOVL    WRK_L_CHARPTR(R10),R1   ;GET CURRENT CHARACTER POINTER
                                                                                        LOCC    1(R1),#BLANKTAB-TERMTAB,TERMTAB ;SEARCH FOR TERMINATOR
                                                                                        BNEQ    10$                     ;IF NEQ INSIGNIFICANT BLANK
                                                                                        MOVZBL  #^A' ',R0               ;RESET BLANK TERMINATOR
                                                                                30$:    BSBW    DCL$BACKUPCHAR          ;RESTORE TERMINATOR TO INPUT BUFFER
                                                                                        ADDL    R10,R2                  ;FIND ADDRESS OF FIRST BYTE IN TOKEN
                                                                                        SUBL3   R2,WRK_L_EXPANDPTR(R10),R1 ;CALCULATE LENGTH OF ITEM
                                                                                        BEQL    50$                     ;IF EQL NO TOKEN
```

```
                26   62    91  0169   442          CMPB    (R2),#^A'&'             ;TOKEN START WITH AMPERSAND?
                     0D    12  016C   443          BNEQ    50$                     ;IF NEQ NO
              F9B2   CA    D5  016E   444          TSTL    WRK_L_SPECRTN(R10)      ;ANY SPECIAL PROCESSING ROUTINE?
                     07    13  0172   445          BEQL    50$                     ;IF NOT, SKIP IT
                50   62    9A  0174   446          MOVZBL  (R2),R0                 ;PICK UP CHARACTER
              F9B2   DA    16  0177   447          JSB     @WRK_L_SPECRTN(R10)     ;CALL SPECIAL PROCESSING ROUTINE
                                0178   448                                          ;AND RETURN WITH SUBSTITUTED VALUE
                               05  0178   449 50$:    RSB
```

```
                     017C    451          .SBTTL  FORCE NONBLANK CHARACTER
                     017C    452 ;+
                     017C    453 ; DCL$FORNBLK - FORCE NONBLANK CHARACTER
                     017C    454 ;
                     017C    455 ; THIS ROUTINE IS CALLED TO FORCE AN INPUT AND THEN SET THE CHARACTER POINTER
                     017C    456 ; TO THE NEXT NONBLANK CHARACTER.
                     017C    457 ;
                     017C    458 ; INPUTS:
                     017C    459 ;
                     017C    460 ;       NONE.
                     017C    461 ;
                     017C    462 ; OUTPUTS:
                     017C    463 ;
                     017C    464 ;       THE NEXT CHARACTER POINTER IS SET TO POINT TO A STRING CONTAINING
                     017C    465 ;       A HYPHEN FOLLOWED BY AN END OF LINE. A SET NONBLANK OPERATION IS
                     017C    466 ;       THEN PERFORMED.
                     017C    467 ;-
                     017C    468
                     017C    469 DCL$FORNBLK::                               ;FORCE NONBLANK CHARACTER
F48E CA   FE7F CF 9E 017C    470          MOVAB   DCL$HYPHEN-1,WRK_L_CHARPTR(R10) ;SET TO FORCE INPUT ON NEXT GET
```

```
                     0183   472          .SBTTL  POINT TO NEXT NONBLANK CHARACTER
                     0183   473 ;+
                     0183   474 ; DCL$SETNBLK - POINT TO NEXT NONBLANK CHARACTER IN INPUT BUFFER
                     0183   475 ;
                     0183   476 ; THIS ROUTINE IS CALLED TO SET THE INPUT BUFFER POINTER TO THE NEXT NONBLANK
                     0183   477 ; CHARACTER.
                     0183   478 ;
                     0183   479 ; INPUTS:
                     0183   480 ;
                     0183   481 ;       NONE.
                     0183   482 ;
                     0183   483 ; OUTPUTS:
                     0183   484 ;
                     0183   485 ;       CHARACTERS ARE OBTAINED FROM THE INPUT BUFFER UNTIL A NONBLANK CHAR-
                     0183   486 ;       ACTER IS ENCOUNTERED AND THE INPUT BUFFER POINTER IS RETURNED TO ITS
                     0183   487 ;       POSITION BEFORE THE LAST GET CHARACTER OPERATION.
                     0183   488 ;
                     0183   489 ;       R0 = NONBLANK CHARACTER.
                     0183   490 ;-
                     0183   491
                     0183   492          .ENABL  LSB
                     0183   493 DCL$SETNBLK::                                ;POINT TO NEXT NONBLANK CHARACTER
7E   F0 AA   01   04 EF   0183   494          EXTZV   #WRK_V_QUOTE,#1,WRK_W_FLAGS(R10),-(SP) ; SAVE QUOTATION FLAG
                008F 30   0189   495 10$:     BSBW    DCL$GETCHAR             ;GET NEXT CHARACTER FROM INPUT BUFFER
                  1C 13   018C   496          BEQL    50$                     ;IF EQL END OF LINE
              20  50 91   018E   497          CMPB    R0,#^A' '               ;BLANK?
                  F6 13   0191   498          BEQL    10$                     ;IF EQL YES
                  0A 11   0193   499          BRB     30$                     ;BACKUP BEFORE CHARACTER AND EXIT
```

```
                     0195   501          .SBTTL  PEEK AT NEXT CHARACTER
                     0195   502 ;+
                     0195   503 ; DCL$SETCHAR - PEEK AT NEXT CHARACTER IN INPUT BUFFER
                     0195   504 ;
                     0195   505 ; THIS ROUTINE IS CALLED TO OBTAIN THE NEXT CHARACTER IN THE INPUT BUFFER
                     0195   506 ; WITHOUT MOVING THE BUFFER POINTER.
                     0195   507 ;
                     0195   508 ; INPUTS:
                     0195   509 ;
                     0195   510 ;       NONE.
                     0195   511 ;
                     0195   512 ; OUTPUTS:
                     0195   513 ;
                     0195   514 ;       THE NEXT CHARACTER IS OBTAINED FROM THE INPUT BUFFER AND THE CHARACTER
                     0195   515 ;       POINTER IS RETURNED TO ITS POSITION BEFORE THE GET CHARACTER OPERATION.
                     0195   516 ;
                     0195   517 ;       R0 = NEXT CHARACTER IN INPUT BUFFER.
                     0195   518 ;-
                     0195   519
                     0195   520 DCL$SETCHAR::                                ;PEEK AT NEXT CHARACTER
7E   F0 AA   01   04 EF   0195   521          EXTZV   #WRK_V_QUOTE,#1,WRK_W_FLAGS(R10),-(SP) ; SAVE QUOTATION FLAG
                  7E 10   019B   522          BSBB    DCL$GETCHAR             ;GET NEXT CHARACTER FROM INPUT BUFFER
                  0B 13   019D   523          BEQL    50$                     ;IF EQL END OF LINE
              F48E CA D7  019F   524 30$:     DECL    WRK_L_CHARPTR(R10)      ;BACK UP INPUT BUFFER POINTER
F0 AA   01   04   8E F0   01A3   525 40$:     INSV    (SP)+,#WRK_V_QUOTE,#1,WRK_W_FLAGS(R10);RESTORE QUOTATION FLAG
                  05 01A9   526          RSB                             ;
                  8E D5   01AA   527 50$:     TSTL    (SP)+                   ;IGNORE SAVED QUOTATION FLAG ON EOL
                  05 01AC   528          RSB
                     01AD   529          .DSABL  LSB
```

```
                          01AD   531          .SBTTL  BYPASS INSIGNIFICANT BLANKS
                          01AD   532 ;+
                          01AD   533 ; DCL$TESTBLANK - BYPASS INSIGNIFICANT BLANKS IN INPUT BUFFER
                          01AD   534 ;
                          01AD   535 ; THIS ROUTINE IS CALLED TO TEST IF THE NEXT CHARACTER IN THE INPUT BUFFER IS
                          01AD   536 ; A SIGNIFICANT BLANK, AND IF IT IS NOT SIGNIFICANT, THROW IT AWAY.
                          01AD   537 ;
                          01AD   538 ; INPUTS:
                          01AD   539 ;
                          01AD   540 ;       NONE.
                          01AD   541 ;
                          01AD   542 ; OUTPUTS:
                          01AD   543 ;
                          01AD   544 ;       R0 = NEXT CHARACTER IN INPUT BUFFER
                          01AD   545 ;-
                          01AD   546
                          01AD   547 DCL$TESTBLANK::                          ;TEST FOR SIGNIFICANT BLANK
              40    10    01AD   548          BSBB    DCL$MOVCHAR            ;PEEK AT NEXT CHARACTER IN INPUT BUFFER
        50    20    91    01AF   549          CMPB    #^A/ /,R0              ;BLANK?
              28    12    01B2   550          BNEQ    DCL$BACKUPMOVE         ;IF NEQ NO
              0D    E1    01B4   551          BBC     #WRK_V_USRMODE,-       ;IF SET THEN USER MODE PARSE
           04 F0    AA    01B6   552                  WRK_W_FLAGS(R10),10$   ;
              DA    10    01B9   553          BSBB    DCL$SETCHAR            ;PEEK AT NEXT CHARACTER IN INPUT BUFFER
              0A    11    01BB   554          BRB     20$                    ;SKIP SUPERVISOR MODE PEEK AHEAD
                          01BD   555 10$:      SETBIT  PRC_V_IND,PRC_W_FLAGS(R11) ;DISABLE INDIRECT DURING LOOK AHEAD
              D2    10    01C1   556          BSBB    DCL$SETCHAR            ;PEEK AT NEXT CHARACTER IN INPUT BUFFER
                          01C3   557          CLRBIT  PRC_V_IND,PRC_W_FLAGS(R11) ;RE-ENABLE INDIRECT
     FE3B CF    0B    50    3A    01C7   558 20$:      LOCC    R0,#BLANKTAB-TERMTAB,TERMTAB ;SEARCH FOR TERMINATOR
              0D    13    01CD   559          BEQL    DCL$BACKUPMOVE         ;BR IF BLANK IS SIGNIFICANT
        50    F486  CA    D0    01CF   560          MOVL    WRK_L_EXPANDPTR(R10),R0 ;GET POINTER INTO EXPANSION BUFFER
          FF  A0    61    90    01D4   561          MOVB    (R1),-1(R0)            ;PUT REAL TERMINATOR OVER BLANK
              F48E  CA    D6    01D8   562          INCL    WRK_L_CHARPTR(R10)     ;ADJUST POINTER FOR THE BLANK
```

```
                     01DC    564          .SBTTL  BACKUP CHARACTER FROM EXPANSION TO INPUT BUFFER
                     01DC    565 ;+
                     01DC    566 ; DCL$BACKUPMOVE - BACKUP ONE CHARACTER FROM EXPANSION TO INPUT BUFFER
                     01DC    567 ;
                     01DC    568 ; THIS ROUTINE RESTORES THE CHARACTER MOST RECENTLY WRITTEN INTO THE EXPANSION
                     01DC    569 ; BUFFER BACK INTO THE INPUT BUFFER AND RESETS THE POINTERS.  THIS EFFECTIVELY
                     01DC    570 ; TAKES BACK THE MOST RECENT 'MOVCHAR'.
                     01DC    571 ;
                     01DC    572 ; INPUTS:
                     01DC    573 ;
                     01DC    574 ;       R10 = ADDRESS OF COMMAND WORK AREA
                     01DC    575 ;
                     01DC    576 ; OUTPUTS:
                     01DC    577 ;
                     01DC    578 ;       THE CHARACTER IS COPIED BACK INTO THE INPUT BUFFER AND THE POINTERS
                     01DC    579 ;       ARE RESET.
                     01DC    580 ;-
                     01DC    581 DCL$BACKUPMOVE::
         F486 CA   D7  01DC    582          DECL    WRK_L_EXPANDPTR(R10)    ;BACK UP EXPANSION BUFFER POINTER
50       F486 DA   90  01E0    583          MOVB    @WRK_L_EXPANDPTR(R10),R0 ;GET CHARACTER
```

```
                     01E5    585          .SBTTL  BACKUP CHARACTER TO INPUT BUFFER
                     01E5    586 ;+
                     01E5    587 ; DCL$BACKUPCHAR - BACKUP ONE CHARACTER TO INPUT BUFFER
                     01E5    588 ;
                     01E5    589 ; THIS ROUTINE WRITES A GIVEN CHARACTER BACK INTO THE INPUT BUFFER (AT THE
                     01E5    590 ; BEGINNING) AND RESETS THE INPUT POINTER.
                     01E5    591 ;
                     01E5    592 ; INPUTS:
                     01E5    593 ;
                     01E5    594 ;       R0 = CHARACTER TO BE WRITTEN
                     01E5    595 ;       R10 = ADDRESS OF COMMAND WORK AREA
                     01E5    596 ;
                     01E5    597 ; OUTPUTS:
                     01E5    598 ;
                     01E5    599 ;       THE CHARACTER IS APPENDED TO THE FRONT OF THE INPUT BUFFER
                     01E5    600 ;       AND THE POINTERS ARE RESET.
                     01E5    601 ;-
                     01E5    602 DCL$BACKUPCHAR::
F48E DA   50   90    01E5    603          MOVB    R0,@WRK_L_CHARPTR(R10)  ;WRITE CHARACTER INTO INPUT BUFFER
     F48E CA   D7    01EA    604          DECL    WRK_L_CHARPTR(R10)      ;BACK UP INPUT BUFFER POINTER
               05    01EE    605          RSB
```

```
                     01EF   607          .SBTTL  MOVE CHARACTER FROM INPUT TO EXPANSION BUFFER.
                     01EF   608 ;+
                     01EF   609 ; DCL$MOVCHAR - MOVE CHARACTER TO EXPANSION BUFFER
                     01EF   610 ;
                     01EF   611 ; THIS ROUTINE IS CALLED TO MOVE A CHARACTER FROM THE INPUT BUFFER TO THE
                     01EF   612 ; EXPANSION BUFFER.
                     01EF   613 ;
                     01EF   614 ; INPUTS:
                     01EF   615 ;
                     01EF   616 ;       NONE.
                     01EF   617 ;
                     01EF   618 ; OUTPUTS:
                     01EF   619 ;
                     01EF   620 ;       THE NEXT CHARACTER IS OBTAINED FROM THE INPUT BUFFER AND WRITTEN INTO
                     01EF   621 ;       THE EXPANSION BUFFER.
                     01EF   622 ;
                     01EF   623 ;       R0 = CHARACTER MOVED
                     01EF   624 ;       Z SET IF R0=0
                     01EF   625 ;-
                     01EF   626
                     01EF   627 DCL$MOVCHAR::                           ;MOVE CHARACTER TO EXPANSION BUFFER
 2A   10             01EF   628          BSBB    DCL$GETCHAR            ;GET NEXT CHARACTER FROM INPUT BUFFER
```

```
                        01F1    630          .SBTTL  PUT CHARACTER IN EXPANSION BUFFER
                        01F1    631 ;+
                        01F1    632 ; DCL$PUTCHAR - PUT CHARACTER IN EXPANSION BUFFER
                        01F1    633 ;
                        01F1    634 ; THIS ROUTINE IS CALLED TO WRITE A CHARACTER INTO THE EXPANSION BUFFER.
                        01F1    635 ;
                        01F1    636 ; INPUTS:
                        01F1    637 ;
                        01F1    638 ;       R0 = CHARACTER TO BE WRITTEN IN EXPANSION BUFFER.
                        01F1    639 ;       WRK_L_EXPANDPTR = ADDRESS OF NEXT BYTE IN EXPANSION BUFFER.
                        01F1    640 ;
                        01F1    641 ; OUTPUTS:
                        01F1    642 ;
                        01F1    643 ;       THE SPECIFIED CHARACTER IS WRITTEN INTO THE EXPANSION BUFFER AFTER
                        01F1    644 ;       HAVING CHECKED FOR BUFFER OVERFLOW.
                        01F1    645 ;
                        01F1    646 ;       R0 = CHARACTER WRITTEN
                        01F1    647 ;       Z SET IF R0=0
                        01F1    648 ;-
                        01F1    649
                        01F1    650 DCL$PUTCHAR::                               ;PUT CHARACTER IN EXPANSION BUFFER
     F892 CA    9F      01F1    651          PUSHAB  WRK_G_BUFFER+WRK_C_CMDBUFSIZ(R10) ;END OF EXPANSION AREA
F486 CA    8E    D1      01F5    652          CMPL    (SP)+,WRK_L_EXPANDPTR(R10) ;ROOM FOR ANOTHER CHARACTER?
          0C    1B      01FA    653          BLEQU   10$                     ;IF LEQU NO
F486 DA    50    90      01FC    654          MOVB    R0,@WRK_L_EXPANDPTR(R10) ;STORE CHARACTER IN EXPANSION BUFFER
     F486 CA    D6      0201    655          INCL    WRK_L_EXPANDPTR(R10)    ;INCREMENT EXPANSION POINTER
          50    95      0205    656          TSTB    R0                      ;RETURN WITH PSL SET FOR CHARACTER
                05      0207    657          RSB                             ;
                        0208    658 10$:     STATUS  BUFOVF                  ;SET ERROR STATUS
          0D    E0      020F    659          BBS     #WRK_V_USRMODE,-        ;IF SET THEN USER MODE PARSE
       04 F0 AA                 0211    660                  WRK_W_FLAGS(R10),20$ ;
    68 AB    20    AA      0214    661          BICW    #PRC_M_IND,PRC_W_FLAGS(R11) ;CLEAR INDIRECT IN PROGRESS
          00FA    31      0218    662 20$:     BRW     ERROR                   ;
```

```
                                    021B   664          .SBTTL  GET CHARACTER FROM INPUT BUFFER
                                    021B   665 ;+
                                    021B   666 ; DCL$GETCHAR - GET CHARACTER FROM INPUT BUFFER
                                    021B   667 ;
                                    021B   668 ; THIS ROUTINE IS CALLED TO GET THE NEXT CHARACTER FROM THE INPUT BUFFER.
                                    021B   669 ; IT HANDLES INDIRECT FILES, QUOTES, CONTINUATION, AND BLANK SUPPRESSION.
                                    021B   670 ;
                                    021B   671 ; INPUTS:
                                    021B   672 ;
                                    021B   673 ;       NONE.
                                    021B   674 ;
                                    021B   675 ; OUTPUTS:
                                    021B   676 ;
                                    021B   677 ;       THE NEXT CHARACTER IS READ FROM THE INPUT BUFFER AND RETURNED TO THE
                                    021B   678 ;       CALLER.
                                    021B   679 ;
                                    021B   680 ;       R0 = CHARACTER READ.
                                    021B   681 ;-
                                    021B   682 DCL$GETCHAR::                            ;GET NEXT CHARACTER FROM INPUT BUFFER
                          51    DD  021B   683          PUSHL   R1                      ;SAVE REGISTER
                                    021D   684 NEXTCHAR:                                ;GET NEXT CHARACTER
                F48E      CA    D6  021D   685          INCL    WRK_L_CHARPTR(R10)      ;UPDATE INPUT BUFFER POINTER
                                    0221   686 CURRCHAR:                                ;GET CURRENT CHARACTER
           51   F48E      CA    D0  0221   687          MOVL    WRK_L_CHARPTR(R10),R1   ;GET CHARACTER POINTER
                50        81    9A  0226   688          MOVZBL  (R1)+,R0                ;GET NEXT CHARACTER FROM INPUT BUFFER
                          65    13  0229   689          BEQL    EOL                     ;IF EQL END OF LINE
                50        22    91  022B   690          CMPB    #^A/'/,R0               ;QUOTE?
                          4A    13  022E   691          BEQL    QUOTE                   ;IF EQL YES
          3F F0 AA        04    E0  0230   692          BBS     #WRK_V_QUOTE,WRK_W_FLAGS(R10),EXITCHAR ;EXIT IF IN QUOTED STRING
          1A F0 AA        0A    E0  0235   693          BBS     #WRK_V_INPSUBST,WRK_W_FLAGS(R10),20$ ;IF SUBSTITUTITION PASS
             50     40    8F    91  023A   694          CMPB    #^A/@/,R0               ;AT SIGN?
                          40    13  023E   695          BEQL    INDIRECT
                50        21    91  0240   696          CMPB    #^A/!/,R0               ;EXCLAMATION?
                          47    13  0243   697          BEQL    TRUNCATE
                50        09    91  0245   698          CMPB    #^A/    /,R0            ;TAB?
                          50    13  0248   699          BEQL    TAB
                50        20    91  024A   700          CMPB    #^A/ /,R0               ;BLANK?
                          4E    13  024D   701          BEQL    BLANK
                50        2D    91  024F   702          CMPB    #^A/-/,R0               ;HYPHEN?
                          72    13  0252   703          BEQL    CONTINUATION
          1B F0 AA        0B    E0  0254   704 20$:     BBS     #WRK_V_NOUPCASE,WRK_W_FLAGS(R10),EXITCHAR ;SKIP IF NO UPCASING
             61 8F        50    91  0259   705          CMPB    R0,#^A/a/               ;CHECK LOW LIMIT OF LOW RANGE
                          15    1F  025D   706          BLSSU   EXITCHAR                ;BR IF FAILED
             7A 8F        50    91  025F   707          CMPB    R0,#^A/z/               ;CHECK HIGH LIMIT OF LOW RANGE
                          0C    1B  0263   708          BLEQU   25$                     ;BR IF VALID CHARACTER
             E0 8F        50    91  0265   709          CMPB    R0,#^XE0                ;CHECK LOW LIMIT OF HIGH RANGE
                          09    1F  0269   710          BLSSU   EXITCHAR                ;BR IF FAILED
             FE 8F        50    91  026B   711          CMPB    R0,#^XFE                ;CHECK HIGH LIMIT OF HIGH RANGE
                          03    1A  026F   712          BGTRU   EXITCHAR                ;BR IF FAILED
                50        20    8A  0271   713 25$:     BICB    #^X20,R0                ;UPCASE THE CHARACTER
                                    0274   714
                                    0274   715 EXITCHAR:                                ;EXIT WITH CHARACTER
                          51 8ED0   0274   716          POPL    R1                      ;RESTORE REGISTER
                          50    95  0277   717          TSTB    R0                      ;SET CONDITION CODES BASED ON CHARACTER
                                05  0279   718          RSB                             ;
                                    027A   719 ;
                                    027A   720 ; STRING QUOTATION DELIMITER DETECTED.  INVERT QUOTE FLAG.
```

```
                     027A   721 ;
F0 AA   10   AC      027A   722 QUOTE:  XORW    #WRK_M_QUOTE,WRK_W_FLAGS(R10) ;INVERT QUOTE FLAG
        F4   11      027E   723         BRB     EXITCHAR                      ;
                     0280   724
                     0280   725 ;
                     0280   726 ; INDIRECTION (@) REQUESTED
                     0280   727 ;
                     0280   728 INDIRECT:
   F9B2 CA   D5      0280   729         TSTL    WRK_L_SPECRTN(R10)            ;ANY SPECIAL PROCESSING ROUTINE?
        EE   13      0284   730         BEQL    EXITCHAR                      ;IF NOT, RETURN @ CHARACTER
   F9B2 DA   16      0286   731         JSB     @WRK_L_SPECRTN(R10)           ;CALL SPECIAL PROCESSING ROUTINE
        E8   11      028A   732         BRB     EXITCHAR                      ;EXIT WITH NEXT CHARACTER
                     028C   733
                     028C   734 ;
                     028C   735 ; COMMENT CHARACTER DETECTED.  FAKE AN END OF LINE CONDITION.
                     028C   736 ;
                     028C   737 TRUNCATE:
        71   94      028C   738         CLRB    -(R1)                         ;SET END OF LINE IN BUFFER
        91   11      028E   739         BRB     CURRCHAR
                     0290   740 ;
                     0290   741 ; END OF LINE DETECTED.  TERMINATE ANY STRING AND EXIT
                     0290   742 ;
F0 AA   10   AA      0290   743 EOL:    BICW    #WRK_M_QUOTE,WRK_W_FLAGS(R10) ;CLEAR QUOTE IN PROGRESS
   F48E CA   D7      0294   744         DECL    WRK_L_CHARPTR(R10)            ;BACK UP INPUT BUFFER POINTER
        DA   11      0298   745         BRB     EXITCHAR
                     029A   746 ;
                     029A   747 ; TAB DETECTED.  CONVERT TO A SPACE.
                     029A   748 ;
   50   20   9A      029A   749 TAB:    MOVZBL  #^A/ /,R0                     ;SET CHARACTER TO A BLANK
                     029D   750 ;
                     029D   751 ; COMPRESS MULTIPLE SPACES AND TABS INTO A SINGLE SPACE
                     029D   752 ;
F48E DA   50   90    029D   753 BLANK:  MOVB    R0,@WRK_L_CHARPTR(R10)        ;STORE BLANK IN INPUT BUFFER
     81   20   91    02A2   754         CMPB    #^A/ /,(R1)+                  ;NEXT CHARACTER A BLANK?
          06   13    02A5   755         BEQL    30$                           ;IF EQL YES
  FF A1   09   91    02A7   756         CMPB    #^A/    /,-1(R1)              ;NEXT CHARACTER A TAB?
          06   12    02AB   757         BNEQ    40$                           ;IF NEQ NO
     F48E CA   D6    02AD   758 30$:    INCL    WRK_L_CHARPTR(R10)            ;INCREMENT CHARACTER POINTER
          EA   11    02B1   759         BRB     BLANK                         ;
                     02B3   760 ;
                     02B3   761 ; STRIP ANY TRAILING SPACES ON THE END OF THE LINE
                     02B3   762 ;
          71   95    02B3   763 40$:    TSTB    -(R1)                         ;NEXT CHARACTER END OF LINE?
          D5   13    02B5   764         BEQL    TRUNCATE                      ;IF EQL YES
     61   21   91    02B7   765         CMPB    #^A/!/,(R1)                   ;NEXT CHARACTER EXCLAMATION?
          D0   13    02BA   766         BEQL    TRUNCATE                      ;IF EQL YES
     81   2D   91    02BC   767         CMPB    #^A/-/,(R1)+                  ;HYPHEN?
          B3   12    02BF   768         BNEQ    EXITCHAR                      ;IF NEQ NO
                     02C1   769         SETBIT  WRK_V_TRAILSPC,WRK_W_FLAGS(R10) ;INDICATE TRAILING SPACES SEEN
                     02C6   770 ;
                     02C6   771 ; CONTINUATION CHARACTER DETECTED.  IF ITS THE LAST CHARACTER ON
                     02C6   772 ; THE LINE, READ THE NEXT INPUT RECORD AND READ ITS FIRST CHARACTER.
                     02C6   773 ;
                     02C6   774 CONTINUATION:
          61   95    02C6   775         TSTB    (R1)                          ;NEXT CHARACTER END OF LINE?
          17   13    02C8   776         BEQL    NEXT_RECORD                   ;IF EQL YES
     61   21   91    02CA   777         CMPB    #^A/!/,(R1)                   ;NEXT CHARACTER EXCLAMATION?
```

```
                12   13  02CD   778          BEQL    NEXT_RECORD             ;IF EQL YES
             81 20   91  02CF   779          CMPB    #^A/ /,(R1)+            ;NEXT CHARACTER A BLANK?
                F2   13  02D2   780          BEQL    CONTINUATION            ;IF EQL YES
          FF A1 09   91  02D4   781          CMPB    #^A/    /,-1(R1)        ;NEXT CHARACTER A TAB?
                EC   13  02D8   782          BEQL    CONTINUATION            ;IF EQL YES
                     02DA   783          CLRBIT  WRK_V_TRAILSPC,WRK_W_FLAGS(R10) ;CLEAR TRAILING SPACE FLAG
                93   11  02DF   784          BRB     EXITCHAR                ;
                     02E1   785
                     02E1   786 ;
                     02E1   787 ; READ THE NEXT RECORD, AND RESTART THE GETCHAR SCAN.
                     02E1   788 ;
                     02E1   789 NEXT_RECORD:
             F9AA CA D5  02E1   790          TSTL    WRK_L_READRTN(R10)      ;ANY INPUT ROUTINE?
                8D   13  02E5   791          BEQL    EXITCHAR                ;IF NOT, RETURN EOL CHARACTER
                0D   E1  02E7   792          BBC     #WRK_V_USRMODE,-        ;BRANCH IF DCL INPUT ROUTINE
          10 F0 AA        02E9   793                  WRK_W_FLAGS(R10),10$
          50 F9AA CA D0  02EC   794          MOVL    WRK_L_READRTN(R10),R0   ;GET ADDRESS OF USER CONTIN ROUTINE
                51   D4  02F1   795          CLRL    R1                      ;DO NOT SUPPLY A PROMPT
                   FD0A' 30  02F3   796          BSBW    DCL$USER_INPUT          ;CALL USER'S ROUTINE FOR INPUT
                   FF22  30  02F6   797          BSBW    DCL$GETCHAR             ;GET FIRST CHARACTER IN INPUT LINE
                   FF78  31  02F9   798          BRW     EXITCHAR                ;EXIT WITH FIRST CHARACTER
          50 FD01 CF 9E  02FC   799 10$:    MOVAB   DCL$HYPHEN+1,R0         ;GET ADDRESS OF FORCED INPUT CHARACTER
             51   50  D1  0301   800          CMPL    R0,R1                   ;ARE WE FAKING A CONTINUATION?
                04   13  0304   801          BEQL    15$                     ;YES, THEN DON'T SET THE FLAG
          F0 AA 08   A8  0306   802          BISW    #WRK_M_CONTIN,WRK_W_FLAGS(R10) ;SET CONTINUATION IN PROGRESS
             F9AA DA 16  030A   803 15$:    JSB     @WRK_L_READRTN(R10)     ;CALL INPUT ROUTINE
          F0 AA 08   AA  030E   804          BICW    #WRK_M_CONTIN,WRK_W_FLAGS(R10) ;SET CONTINUATION IN PROGRESS
                   FF5F  31  0312   805          BRW     EXITCHAR                ;EXIT WITH FIRST CHARACTER
                         0315   806
                         0315   807 ;
                         0315   808 ; ERROR EXIT FROM CHARACTER INPUT ROUTINES
                         0315   809 ;
                         0315   810
          F9AE CA D5  0315   811 ERROR:  TSTL    WRK_L_ERRORRTN(R10)     ;ANY ERROR HANDLER ROUTINE?
                04   13  0319   812          BEQL    90$                     ;IF NOT, IGNORE ERROR
          F9AE DA 16  031B   813          JSB     @WRK_L_ERRORRTN(R10)    ;CALL ERROR HANDLER
                05   031F   814 90$:    RSB
                     0320   815
                     0320   816          .END
```

```
BLANK                   0000029D R      02
BLANKTAB                00000013 R      02
CLI$_BUFOVF           = 00038018
CLI$_RSLOVF           = 00038118
CLI$_TKNOVF           = 000382A0
CONTINUATION            000002C6 R      02
CURRCHAR                00000221 R      02
DCL$BACKUPCHAR          000001E5 RG     02
DCL$BACKUPMOVE          000001DC RG     02
DCL$COMPRESS            000000B7 RG     02
DCL$COMPSTRING          000000C0 RG     02
DCL$FORNBLK             0000017C RG     02
DCL$GENDESCR            0000002C RG     02
DCL$GENTERM             0000008E RG     02
DCL$GETCHAR             0000021B RG     02
DCL$GETNBLK             000000FE RG     02
DCL$GETOKEN             0000010C RG     02
DCL$GTBTOKEN            0000010E RG     02
DCL$HYPHEN              00000000 RG     02
DCL$LOCATE              00000020 RG     02
DCL$MARK                000000EA RG     02
DCL$MARKEDTOKEN         000000F2 RG     02
DCL$MOVBTOKN            00000104 RG     02
DCL$MOVCHAR             000001EF RG     02
DCL$MOVTOKN             00000109 RG     02
DCL$PUTCHAR             000001F1 RG     02
DCL$SETCHAR             00000195 RG     02
DCL$SETNBLK             00000183 RG     02
DCL$TESTBLANK           000001AD RG     02
DCL$USER_INPUT          ********   X    02
ENDTERM                 00000020 R      02
EOL                     00000290 R      02
ERROR                   00000315 R      02
EXITCHAR                00000274 R      02
INDIRECT                00000280 R      02
NEXTCHAR                0000021D R      02
NEXT_RECORD             000002E1 R      02
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_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_IND             = 00000020
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_V_IND             = 00000005
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
PTR_B_LEVEL             00000004
```

```
PTR_B_NUMBER                    00000005
PTR_B_PARMCNT                   00000006
PTR_B_VALUE                     00000000
PTR_C_LENGTH                    0000000C
PTR_K_BLANK                   = 00000001
PTR_K_COLON                   = 00000002
PTR_K_LENGTH                    0000000C
PTR_K_LPAREN                  = 00000007
PTR_L_DESCR                     00000000
PTR_L_ENTITY                    00000008
PTR_S_FLAGS                   = 00000004
PTR_S_OFFSET                  = 0000000C
PTR_S_TERM                    = 00000004
PTR_S_TYPE                    = 00000004
PTR_S_VALUE                   = 00000008
PTR_V_FLAGS                   = 00000014
PTR_V_OFFSET                  = 00000008
PTR_V_TERM                    = 00000018
PTR_V_TYPE                    = 0000001C
PTR_V_VALUE                   = 00000000
QUOTE                           0000027A R      02
TAB                             0000029A R      02
TERMCLASS                       00000002 R      02
TERMTAB                         00000008 R      02
TRUNCATE                        0000028C 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_CMDBUFSIZ               = 00000400
WRK_C_LENGTH                    FFFFF486
WRK_C_RSLBUFSIZ               = 00000600
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_M_CONTIN                  = 00000008
WRK_M_ESCAPE                  = 00004000
WRK_M_QUOTE                   = 00000010
WRK_V_ESCAPE                  = 0000000E
WRK_V_INPSUBST                = 0000000A
WRK_V_NOUPCASE                = 0000000B
WRK_V_QUOTE                   = 00000004
WRK_V_STAR                    = 00000005
WRK_V_TRAILSPC                = 00000009
WRK_V_USRMODE                 = 0000000D
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 | 00000320 ( 800.) | 02 ( 2.) | NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE |

### Performance indicators

| Phase | Page faults | CPU Time | Elapsed Time |
|---|---|---|---|
| Initialization | 9 | 00:00:00.09 | 00:00:01.47 |
| Command processing | 97 | 00:00:00.85 | 00:00:06.64 |
| Pass 1 | 235 | 00:00:07.22 | 00:00:21.03 |
| Symbol table sort | 8 | 00:00:00.67 | 00:00:03.70 |
| Pass 2 | 143 | 00:00:01.96 | 00:00:04.27 |
| Symbol table output | 22 | 00:00:00.20 | 00:00:00.64 |
| 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 | 516 | 00:00:11.03 | 00:00:37.77 |

The working set limit was 1200 pages.
34647 bytes (68 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 464 non-local and 29 local symbols.
816 source lines were read in Pass 1, producing 15 object records in Pass 2.
32 pages of virtual memory were used to define 18 macros.

### Macro library statistics

| Macro library name | Macros defined |
|---|---|
| _$255$DUA28:[SYSLIB]SYSBLDMLB.MLB;1 | 0 |
| _$255$DUA28:[DCL.OBJ]DCL.MLB;1 | 8 |
| _$255$DUA28:[SYS.OBJ]LIB.MLB;1 | 0 |
| _$255$DUA28:[SYSLIB]STARLET.MLB;2 | 4 |
| TOTALS (all libraries) | 12 |

581 GETS were required to define 12 macros.

There were no errors, warnings or information messages.

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

INTDEF
SDL

COMMAND
LIS

DESCRVAL
LIS

CONVERT
LIS

CLIMAC
MAR

CLIMSG
LIS

CONNECT
LIS

DCLPARSE
LIS

CHARMANIP
LIS

EXAMDEP
LIS

EXIT
LIS

INTIMAGES
MAR

DCXSTART
LIS

CLIGBL
LIS

DISALLOW
LIS

CLINT
LIS

CANCEL
LIS