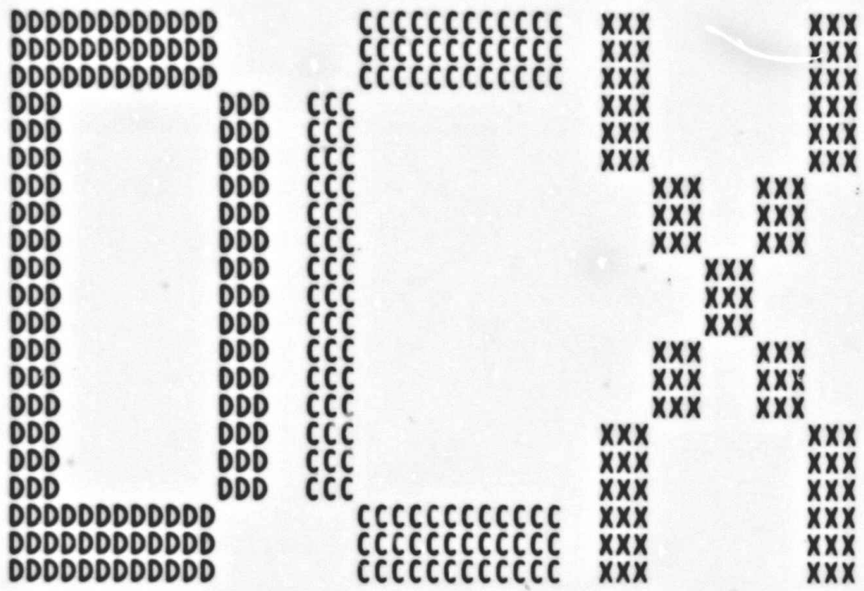
DDDDDDDDDDDD CCCCCCCCCCCC XXX XXX
DDDDDDDDDDDD CCCCCCCCCCCC XXX XXX
DDDDDDDDDDDD CCCCCCCCCCCC XXX XXX
DDD DDD CCC XXX XXX
DDD DDD CCC XXX XXX
DDD DDD CCC XXX XXX
DDD DDD CCC XXX XXX
DDD DDD CCC XXX XXX
DDD DDD CCC XXX
DDD DDD CCC XXX
DDD DDD CCC XXX
DDD DDD CCC XXX XXX
DDD DDD CCC XXX XXX
DDD DDD CCC XXX XXX
DDD DDD CCC XXX XXX
DDD DDD CCC XXX XXX
DDDDDDDDDDDD CCCCCCCCCCCC XXX XXX
DDDDDDDDDDDD CCCCCCCCCCCC XXX XXX
DDDDDDDDDDDD CCCCCCCCCCCC XXX XXX

FILEID**EXPAND

```
EEEEEEEEEE  XX      XX  PPPPPPPP     AAAAAA    NN      NN  DDDDDDDD
EEEEEEEEEE  XX      XX  PPPPPPPP     AAAAAA    NN      NN  DDDDDDDD
EE          XX      XX  PP      PP  AA    AA  NN      NN  DD      DD
EE          XX      XX  PP      PP  AA    AA  NN      NN  DD      DD
EE            XX  XX    PP      PP  AA    AA  NNNN    NN  DD      DD
EE            XX  XX    PP      PP  AA    AA  NNNN    NN  DD      DD
EEEEEEEE        XX      PPPPPPPP    AA    AA  NN  NN  NN  DD      DD
EEEEEEEE        XX      PPPPPPPP    AA    AA  NN  NN  NN  DD      DD
EE            XX  XX    PP          AAAAAAAAAA  NN    NNNN  DD      DD
EE            XX  XX    PP          AAAAAAAAAA  NN    NNNN  DD      DD
EE          XX      XX  PP          AA    AA  NN      NN  DD      DD      ....
EE          XX      XX  PP          AA    AA  NN      NN  DD      DD      ....
EEEEEEEEEE  XX      XX  PP          AA    AA  NN      NN  DDDDDDDD        ....
EEEEEEEEEE  XX      XX  PP          AA    AA  NN      NN  DDDDDDDD        ....

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 dcx_expand (                       ! Data expansion routines
;     2          0002  0                 LANGUAGE (BLISS32),
;     3          0003  0                 IDENT = 'V04-000'
;     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:
;    34          0034  1 !
;    35          0035  1 !       DCX -- Data Compression / Expansion Facility
;    36          0036  1 !
;    37          0037  1 !   ABSTRACT:
;    38          0038  1 !
;    39          0039  1 !       The Data Compression / Expansion procedures provide a general
;    40          0040  1 !       method for reducing the storage requirement for a arbitrary data.
;    41          0041  1 !
;    42          0042  1 ! ENVIRONMENT:
;    43          0043  1 !
;    44          0044  1 !       VAX native, user mode.
;    45          0045  1 !
;    46          0046  1 !--
;    47          0047  1 !
;    48          0048  1 !
;    49          0049  1 ! AUTHOR:  David Thiel
;    50          0050  1 !
;    51          0051  1 ! CREATION DATE: July, 1981
;    52          0052  1 !
;    53          0053  1 ! MODIFIED BY:
;    54          0054  1 !
;    55          0055  1 !--
```

```
;    57          0056  1 %SBTTL  'Declarations';
;    58          0057  1
;    59          0058  1 LIBRARY
;    60          0059  1         'sys$library:starlet';  ! System macros
;    61          0060  1 REQUIRE
;    62          0061  1         'prefix';               ! DCX macros
;    63          0204  1 REQUIRE
;    64          0205  1         'dcxdef';               ! DCX public structure definitions
;    65          0299  1 REQUIRE
;    66          0300  1         'dcxprvdef';            ! DCX private structure definitions
;    67          0466  1
;    68          0467  1 EXTERNAL ROUTINE
;    69          0468  1     dcx$ctx_check : lkg_ctx_check,      ! Check context block
;    70          0469  1     dcx$map_check : lkg_map_check,      ! Check map
;    71          0470  1     dcx$get_vm,                         ! Allocate memory
;    72          0471  1     dcx$free_vm,                        ! Deallocate memory
;    73          0472  1     dcx$long_move : lkg_long_move NOVALUE,      ! Copy arbitrary length data
;    74          0473  1     lib$scopy_r_dx :                    ! General string copy
;    75          0474  1         ADDRESSING_MODE (GENERAL);
;    76          0475  1
;    77          0476  1 EXTERNAL LITERAL
;    78          0477  1     dcx$_normal,
;    79          0478  1     dcx$_invctx,
;    80          0479  1     dcx$_invdata,
;    81          0480  1     dcx$_trunc,
;    82          0481  1     lib$_strtru;
;    83          0482  1
;    84          0483  1 FORWARD ROUTINE
;    85          0484  1     dcx$expand_init,                    ! initialize for data expansion
;    86          0485  1     dcx$expand_data,                    ! expand data record
;    87          0486  1     dcx$do_expansion_0,                 ! internal expansion routine -- type 0
;    88          0487  1     dcx$expand_done;                    ! delete expansion context
```

```
;    90          0488  1 %SBTTL  'dcx$expand_init - Initialization for data expansion'
;    91          0489  1
;    92          0490  1 GLOBAL ROUTINE dcx$expand_init (context_addr, map_addr) =
;    93          0491  2 BEGIN
;    94          0492  2 !++
;    95          0493  2 !
;    96          0494  2 ! Initialization for data expansion.
;    97          0495  2 ! Allocate and initialize context area.
;    98          0496  2 !
;    99          0497  2 ! Inputs:
;   100          0498  2 !
;   101          0499  2 !       context_addr.mz.r       Address of context longword
;   102          0500  2 !       map_addr.ra.r           Address of map
;   103          0501  2 !
;   104          0502  2 ! Outputs:
;   105          0503  2 !
;   106          0504  2 !       context_addr.mz.r       Address of context block is stored
;   107          0505  2 !
;   108          0506  2 ! Return value:
;   109          0507  2 !
;   110          0508  2 !       status.wlc.v
;   111          0509  2 !
;   112          0510  2 !               dcx$_normal     All is well
;   113          0511  2 !               dcx$_invmap     Invalid map structure
;   114          0512  2 !               lib$_insvirmem  Error allocating memory
;   115          0513  2 !--
;   116          0514  2
;   117          0515  2 BIND
;   118          0516  2     ctx = .context_addr : REF BBLOCK,   ! address of context block
;   119          0517  2     dcxmap = .map_addr : REF BBLOCK;    ! address of map
;   120          0518  2
;   121          0519  2 ctx = 0;                                ! assume failure
;   122          0520  2 perform (dcx$map_check (.dcxmap));      ! validate map
;   123          0521  2 perform (dcx$get_vm (ctx$k_fixed_len + exp$k_length, ctx));
;   124          0522  2 IF true
;   125          0523  2 THEN
;   126          0524  3     BEGIN
;   127          0525  3
;   128          0526  3     BIND
;   129          0527  3         exp = ctx [ctx$l_specific] : BBLOCK;
;   130          0528  3
;   131          0529  3     LOCAL
;   132          0530  3         cptr : REF VECTOR [, LONG],
;   133          0531  3         dcxsbm : REF BBLOCK;
;   134          0532  3
;   135          0533  3     ctx [ctx$l_size] = ctx$k_fixed_len + exp$k_length;
;   136          0534  3     ctx [ctx$b_type] = ctx$c_expnd;
;   137          0535  3     ctx [ctx$w_version] = ctx$c_version;
;   138          0536  3     ctx [ctx$l_sanity] = ctx$c_sanity;
;   139          0537  3     ctx [ctx$l_map] = .dcxmap;
;   140          0538  3     perform (dcx$get_vm (4 * 4 * .dcxmap [dcxmap$w_nsubs], exp [exp$l_map_segs]));
;   141          0539  3     cptr = .exp [exp$l_map_segs];
;   142          0540  3     dcxsbm = .dcxmap + .dcxmap [dcxmap$w_sub0];
;   143          0541  3     INCR i FROM 0 to .dcxmap [dcxmap$w_nsubs]-1 DO
;   144          0542  4         BEGIN
;   145          0543  4         cptr [0] = .dcxsbm + .dcxsbm [dcxsbm$w_flags];
;   146          0544  4         cptr [1] = .dcxsbm + .dcxsbm [dcxsbm$w_nodes];
```

```
;  147          0545  4          IF .dcxsbm [dcxsbm$w_next] EQL 0
;  148          0546  4          THEN
;  149          0547  4              cptr [2] = 0
;  150          0548  4          ELSE
;  151          0549  4              cptr [2] = .dcxsbm + .dcxsbm [dcxsbm$w_next] - 2 * .dcxsbm [dcxsbm$b_min_char];
;  152          0550  4          cptr [3] = .dcxsbm;
;  153          0551  4          cptr = cptr [4];
;  154          0552  4          dcxsbm = .dcxsbm + .dcxsbm [dcxsbm$w_size];
;  155          0553  3          END;
;  156          0554  2      END;
;  157          0555  2 RETURN dcx$_normal;
;  158          0556  2
;  159          0557  1 END;                                    ! of dcx$expand_init

                                                                .TITLE  DCX_EXPAND
                                                                .IDENT  \V04-000\

                                                                .EXTRN  LIB$ANALYZE_SDESC_R2
                                                                .EXTRN  DCX$CTX_CHECK, DCX$MAP_CHECK
                                                                .EXTRN  DCX$GET_VM, DCX$FREE_VM
                                                                .EXTRN  DCX$LONG_MOVE, LIB$SCOPY_R_DX
                                                                .EXTRN  DCX$_NORMAL, DCX$_INVCTX
                                                                .EXTRN  DCX$_INVDATA, DCX$_TRUNC
                                                                .EXTRN  LIB$_STRTRU

                                                                .PSECT  $CODE$,NOWRT,2

                                    003C 00000                  .ENTRY  DCX$EXPAND_INIT, Save R2,R3,R4,R5               ; 0490
                                04  BC   D4 00002               CLRL    @CONTEXT_ADDR                                   ; 0519
                            53  08  BC   D0 00005               MOVL    @MAP_ADDR, R3                                   ; 0520
                            50       53   D0 00009               MOVL    R3, R0
                                 0000G 30   0000C               BSBW    DCX$MAP_CHECK
                            37        50   E9 0000F               BLBC    STATUS, 1$
                                04  AC   DD 00012               PUSHL   CONTEXT_ADDR                                    ; 0521
                                     18   DD 00015               PUSHL   #24
                         0000G  CF        02   FB 00017               CALLS   #2, DCX$GET_VM
                            2A           50   E9 0001C               BLBC    STATUS, 1$
                            52  04  BC   D0 0001F               MOVL    @CONTEXT_ADDR, R2                               ; 0527
                            62       18   D0 00023               MOVL    #24, (R2)                                       ; 0533
                          04 A2          02   90 00026               MOVB    #2, 4(R2)                                       ; 0534
                             08  A2      B4 0002A               CLRW    8(R2)                                           ; 0535
                          0C A2 4F317C65 8F   D0 00D2D               MOVL    #1328643173, 12(R2)                             ; 0536
                          10 A2          53   D0 00035               MOVL    R3, 16(R2)                                      ; 0537
                             14  A2      9F   00039               PUSHAB  20(R2)                                          ; 0538
                            50   10  A3   3C   0003C               MOVZWL  16(R3), R0
                      7E    50           04   78 00040               ASHL    #4, R0, -(SP)
                         0000G  CF        02   FB 00044               CALLS   #2, DCX$GET_VM
                            58           50   E9 00049 1$:            BLBC    STATUS, 6$
                            50   14  A2   D0 0004C               MOVL    20(R2), CPTR                                    ; 0539
                            51   12  A3   3C 00050               MOVZWL  18(R3), DCXSBM                                  ; 0540
                            51       53   C0 00054               ADDL2   R3, DCXSBM
                            55   10  A3   3C 00057               MOVZWL  16(R3), R5                                      ; 0541
                            54           01   CE 0005B               MNEGL   #1, I
                                         39   11 0005E               BRB     5$
                            52   06  A1   3C 00060 2$:            MOVZWL  6(DCXSBM), R2                                   ; 0543
                      60    51           52   C1 00064               ADDL3   R2, DCXSBM, (CPTR)
```

```
                                52          08    A1  3C 00068          MOVZWL  8(DCXSBM), R2                           ; 0544
                          04    A0          51          52  C1 0006C          ADDL3   R2, DCXSBM, 4(CPTR)
                                52          0A    A1  3C 00071          MOVZWL  10(DCXSBM), R2                          ; 0545
                                                    05  12 00075          BNEQ    3$
                                            08      A0  D4 00077          CLRL    8(CPTR)                                 ; 0547
                                                    10  11 0007A          BRB     4$
                                53          51          52  C1 0007C 3$:      ADDL3   R2, DCXSBM, R3                  ; 0549
                                52          02      A1  9A 00080          MOVZBL  2(DCXSBM), R2
                                52                  02  C4 00084          MULL2   #2, R2
                          08    A0          53          52  C3 00087          SUBL3   R2, R3, 8(CPTR)
                                      0C    A0          51  D0 0008C 4$:      MOVL    DCXSBM, 12(CPTR)                ; 0550
                                50                  10  C0 00090          ADDL2   #16, CPTR                       ; 0551
                                52                  61  3C 00093          MOVZWL  (DCXSBM), R2                    ; 0552
                                51                  52  C0 00096          ADDL2   R2, DCXSBM
                                C3          54          55  F2 00099 5$:      AOBLSS  R5, I, 2$                       ; 0541
                                            50 00000000G    8F  D0 0009D          MOVL    #DCX$_NORMAL, R0                ; 0555
                                                        04  000A4 6$:      RET                                     ; 0557

; Routine Size:  165 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  161          0558  1 %SBTTL  'dcx$expand_data - Expand data record'
;  162          0559  1
;  163          0560  1 GLOBAL ROUTINE dcx$expand_data (context_addr, in_rec : REF BBLOCK, out_rec : REF BBLOCK, out_len) =
;  164          0561  2 BEGIN
;  165          0562  2 !++
;  166          0563  2 !
;  167          0564  2 ! Expand data record
;  168          0565  2 !
;  169          0566  2 ! Inputs:
;  170          0567  2 !
;  171          0568  2 !       context_addr.mz.r       Address of context longword
;  172          0569  2 !       in_rec.rt.dx            Descriptor for input (text) data record
;  173          0570  2 !       out_rec.wt.dx           Descriptor for output (text) data buffer
;  174          0571  2 !
;  175          0572  2 ! Outputs:
;  176          0573  2 !
;  177          0574  2 !       context_addr.mz.r       Context block accumulates data
;  178          0575  2 !       out_rec.wt.dx           Buffer is filled with output record
;  179          0576  2 !       out_len.wwu.r           Word in which to store length of
;  180          0577  2 !                               output record (optional)
;  181          0578  2 !
;  182          0579  2 ! Return value:
;  183          0580  2 !
;  184          0581  2 !       status.wlc.v
;  185          0582  2 !
;  186          0583  2 !               dcx$_normal     All is well
;  187          0584  2 !               dcx$_invctx     Invalid context block
;  188          0585  2 !               dcx$_invmap     Invalid map
;  189          0586  2 !--
;  190          0587  2
;  191          0588  2 BIND
;  192          0589  2     ctx = .context_addr : REF BBLOCK,   ! context block
;  193          0590  2     dcxmap = ctx [ctx$l_map] : REF BBLOCK,      ! map address
;  194          0591  2     res_len = .out_len : WORD;          ! result length
;  195          0592  2
;  196          0593  2 LOCAL
;  197          0594  2     in_len,                         ! input length (bytes)
;  198          0595  2     in_addr,                        ! input data address
;  199          0596  2     status;                         ! return status
;  200          0597  2
;  201          0598  2 BUILTIN
;  202          0599  2     NULLPARAMETER;
;  203          0600  2
;  204          0601  2 perform (dcx$ctx_check (.ctx, ctx$c_expnd));
;  205          0602  2 perform (dcx$map_check (.dcxmap));
;  206          0603  2 perform (lib$analyze_sdesc_r2 (.in_rec; status, in_len, in_addr); .status);
;  207          0604  2
;  208          0605  2 CASE .out_rec [dsc$b_class]
;  209          0606  2     FROM MIN (dsc$k_class_z, dsc$k_class_s)
;  210          0607  2     TO MAX (dsc$k_class_z, dsc$k_class_s)
;  211          0608  2     OF
;  212          0609  2 SET
;  213          0610  2 [dsc$k_class_z, dsc$k_class_s]:
;  214          0611  3     BEGIN
;  215          0612  3
;  216          0613  3     LOCAL
;  217          0614  3         result : LONG;
```

```
;  218          0615  3
;  219          0616  3      status = dcx$do_expansion_0 (
;  220          0617  3          .ctx, .in_addr, .in_len, .out_rec [dsc$a_pointer], .out_rec [dsc$w_length],
;  221          0618  3          result);
;  222          0619  3      CH$FILL (%C' ', .out_rec [dsc$w_length] - .result, .out_rec [dsc$a_pointer]+.result);
;  223          0620  3      IF NOT NULLPARAMETER (4)
;  224          0621  3      THEN
;  225          0622  3          res_len = .result;
;  226          0623  2      END;
;  227          0624  2  [inrange, outrange]:
;  228          0625  3      BEGIN
;  229          0626  3
;  230          0627  3      LOCAL
;  231          0628  3          result : LONG,
;  232          0629  3          status1 : LONG,
;  233          0630  3          res_buf : VECTOR [65535, BYTE]; ! result buffer
;  234          0631  3
;  235          0632  3      status = dcx$do_expansion_0 (
;  236          0633  3          .ctx, .in_addr, .in_len, res_buf, %ALLOCATION (res_buf),
;  237          0634  3          result);
;  238          0635  3      status1 = lib$scopy_r_dx (result, res_buf, .out_rec);
;  239          0636  3      IF .status1 EQL lib$_strtru
;  240          0637  3      THEN
;  241          0638  4          BEGIN
;  242          0639  4          IF NOT NULLPARAMETER (4)
;  243          0640  4          THEN
;  244          0641  4              lib$analyze_sdesc_r2 (.out_rec; status, res_len);
;  245          0642  4          status1 = dcx$_trunc;
;  246          0643  4          END
;  247          0644  3      ELSE IF NOT NULLPARAMETER (4)
;  248          0645  3      THEN
;  249          0646  3          res_len = .result;
;  250          0647  3      IF .status AND NOT .status1
;  251          0648  3      THEN
;  252          0649  3          status = .status1;
;  253          0650  2      END;
;  254          0651  2  TES;
;  255          0652  2  RETURN .status;
;  256          0653  2
;  257          0654  1  END;                                   ! Of dcx$expand_data


                                    00FC 00000          .ENTRY  DCX$EXPAND_DATA, Save R2,R3,R4,R5,R6,R7      ; 0560
                 57 00000000G   00   9E 00002          MOVAB   LIB$ANALYZE_SDESC_R2, R7                     ;
                 5E FFFEFFF8    EE   9E 00009          MOVAB   -65544(SP), SP                               ;
           52        04    BC        10  C1 00010      ADDL3   #16, @CONTEXT_ADDR, R2                       ; 0590
                 51              02   D0 00015          MOVL    #2, R1                                       ; 0601
                 50        04    BC   D0 00018          MOVL    @CONTEXT_ADDR, R0                            ;
                         0000G   30   0001C             BSBW    DCX$CTX_CHECK                                ;
                 0F              50   E9 0001F          BLBC    STATUS, 1$                                   ;
                 50              62   D0 00022          MOVL    (R2), R0                                     ; 0602
                         0000G   30   00025             BSBW    DCX$MAP_CHECK                                ;
                 06              50   E9 00028          BLBC    STATUS, 1$                                   ;
                 50        08    AC   D0 0002B          MOVL    IN_REC, R0                                   ; 0603
```

```
                                    67  16 0002F          JSB     LIB$ANALYZE_SDESC_R2                                    ;
                                50  E8 00031 1$:        BLBS    STATUS, 2$                                              ;
                                    04 00034          RET                                                             ;
                                53      0C  AC  D0 00035 2$:        MOVL    OUT_REC, R3                                             ; 0605
                             01 00      03  A3  8F 00039          CASEB   3(R3), #0, #1                                   ;
                                0070        0070    0003E 3$:        .WORD   7$-3$,-                                         ;
                                                                    7$-3$                                                   ;
                                5E  DD 00042          PUSHL   SP                                                      ; 0632
                             7E FFFF    8F  3C 00044          MOVZWL  #65535, -(SP)                                   ;
                                   10   AE  9F 00049          PUSHAB  RES_BUF                                         ;
                                51  DD 0004C          PUSHL   IN_LEN                                                  ; 0633
                                52  DD 0004E          PUSHL   IN_ADDR                                                 ;
                                   04   BC  DD 00050          PUSHL   @CONTEXT_ADDR                                   ;
                          0000V CF      06  FB 00053          CALLS   #6, DCX$DO_EXPANSION_0                          ;
                                56  50  D0 00058          MOVL    R0, STATUS                                              ;
                                53  DD 0005B          PUSHL   R3                                                      ; 0635
                                   0C   AE  9F 0005D          PUSHAB  RES_BUF                                         ;
                                   08   AE  9F 00060          PUSHAB  RESULT                                          ;
                   00000000G 00     03  FB 00063          CALLS   #3, LIB$SCOPY_R_DX                              ;
                                54  50  D0 0006A          MOVL    R0, STATUS1                                             ;
                   00000000G 8F     54  D1 0006D          CMPL    STATUS1, #LIB$_STRTRU                           ; 0636
                                1F  12 00074          BNEQ    5$                                                      ;
                             04     6C  91 00076          CMPB    (AP), #4                                        ; 0639
                                11  1F 00079          BLSSU   4$                                                      ;
                                   10   AC  D5 0007B          TSTL    16(AP)                                          ;
                                0C  13 0007E          BEQL    4$                                                      ;
                             50     53  D0 00080          MOVL    R3, R0                                          ; 0641
                                67  16 00083          JSB     LIB$ANALYZE_SDESC_R2                                    ;
                             56     50  D0 00085          MOVL    R0, R6                                                  ;
                          10 BC     51  D0 00088          MOVL    R1, @OUT_LEN                                            ;
                             54 00000000G   8F  D0 0008C 4$:        MOVL    #DCX$_TRUNC, STATUS1                            ; 0642
                                0E  11 00093          BRB     6$                                                      ; 0636
                             04     6C  91 00095 5$:        CMPB    (AP), #4                                        ; 0644
                                09  1F 00098          BLSSU   6$                                                      ;
                                   10   AC  D5 0009A          TSTL    16(AP)                                          ;
                                04  13 0009D          BEQL    6$                                                      ;
                          10 BC     6E  B0 0009F          MOVW    RESULT, @OUT_LEN                                        ; 0646
                             42     56  E9 000A3 6$:        BLBC    STATUS, 8$                                              ; 0647
                             3F     54  E8 000A6          BLBS    STATUS1, 8$                                             ;
                             56     54  D0 000A9          MOVL    STATUS1, STATUS                                         ; 0649
                                3A  11 000AC          BRB     8$                                                      ; 0605
                                   04   AE  9F 000AE 7$:        PUSHAB  RESULT                                          ; 0616
                             7E     63  3C 000B1          MOVZWL  (R3), -(SP)                                     ; 0617
                                   04   A3  DD 000B4          PUSHL   4(R3)                                           ;
                                51  DD 000B7          PUSHL   IN_LEN                                                  ;
                                52  DD 000B9          PUSHL   IN_ADDR                                                 ;
                                   04   BC  DD 000BB          PUSHL   @CONTEXT_ADDR                                   ;
                          0000V CF      06  FB 000BE          CALLS   #6, DCX$DO_EXPANSION_0                          ;
                             56     50  D0 000C3          MOVL    R0, STATUS                                              ;
                             51     63  3C 000C6          MOVZWL  (R3), R1                                        ; 0619
                             51  04    AE  C2 000C9          SUBL2   RESULT, R1                                              ;
                          50 04 A3  04    AE  C1 000CD          ADDL3   RESULT, 4(R3), R0                               ;
                       51 20     6E  00  2C 000D3          MOVC5   #0, (SP), #32, R1, (R0)                         ;
                                60      000D8                                                                           ;
                             04     6C  91 000D9          CMPB    (AP), #4                                        ; 0620
                                0A  1F 000DC          BLSSU   8$                                                      ;
                                   10   AC  D5 000DE          TSTL    16(AP)                                          ;
                                                                                                                        ;
```

```
                                     05  13 000E1          BEQL    8$                                          ;
                          10  BC     04  AE  B0 000E3      MOVW    RESULT, @OUT_LEN                            ; 0622
                              50         56  D0 000E8 8$:  MOVL    STATUS, R0                                  ; 0652
                                         04 000EB          RET                                                 ; 0654

; Routine Size:  236 bytes,    Routine Base:  $CODE$ + 00A5
```

```
;  259          0655  1 %SBTTL  'dcx$do_expansion_0 - Type 0 expansion'
;  260          0656  1
;  261          0657  1 ROUTINE dcx$do_expansion_0 (
;  262          0658  1         ctx : REF BBLOCK, in_addr1 : REF VECTOR [, BYTE], in_bytes, out_addr1 : REF VECTOR [, BYTE], out_byt
;  263          0659  2 BEGIN
;  264          0660  2 !++
;  265          0661  2 !
;  266          0662  2 ! Expand data record using type 0 expansion
;  267          0663  2 !
;  268          0664  2 ! Inputs:
;  269          0665  2 !
;  270          0666  2 !       ctx                     Address of context longword
;  271          0667  2 !       in_addr1                Address of input record
;  272          0668  2 !       in_bytes                Length of input record
;  273          0669  2 !       out_addr1               Address of output buffer
;  274          0670  2 !       out_bytes               Length of output buffer
;  275          0671  2 !
;  276          0672  2 ! Outputs:
;  277          0673  2 !
;  278          0674  2 !       ctx                     Context block accumulates data
;  279          0675  2 !       res_addr                Address in which to store result length
;  280          0676  2 !
;  281          0677  2 ! Status value:
;  282          0678  2 !
;  283          0679  2 !       dcx$_normal             All is well
;  284          0680  2 !       dcx$_invdata            Invalid encoded data
;  285          0681  2 !       dcx$_trunc              Result buffer too small - output truncated
;  286          0682  2 !--
;  287          0683  2
;  288          0684  2 BIND
;  289          0685  2     dcxmap = ctx [ctx$l_map] : REF BBLOCK;      ! map address
;  290          0686  2
;  291          0687  2 .res_addr = 0;                  ! initialize resulting length
;  292          0688  2 IF .dcxmap [dcxmap$w_nsubs] NEQ 0
;  293          0689  2 THEN
;  294          0690  2     BEGIN
;  295          0691  2
;  296          0692  2     LOCAL
;  297          0693  2         res_len : LONG,
;  298          0694  2         in_addr : REF VECTOR [, BYTE],
;  299          0695  2         out_addr : REF VECTOR [, BYTE],
;  300          0696  2         cptr : REF VECTOR [4, LONG],
;  301          0697  2         offset : LONG;
;  302          0698  2
;  303          0699  2     BIND
;  304          0700  2         exp = ctx [ctx$l_specific] : BBLOCK,
;  305          0701  2         base_ptr = exp [exp$l_map_segs] : REF VECTOR [, LONG];
;  306          0702  2
;  307          0703  2     in_addr = .in_addr1;
;  308          0704  2     out_addr = .out_addr1;
;  309          0705  2     res_len = 0;
;  310          0706  2     offset = 0;
;  311          0707  2     cptr = base_ptr [0];
;  312          0708  2     DECR byte_counter FROM .in_bytes - 1 TO 0 DO
;  313          0709  4         BEGIN
;  314          0710  4
;  315          0711  4         LOCAL
```

```
;  316          0712  4              byte_full_of_bits : BITVECTOR [8];
;  317          0713  4
;  318          0714  4          byte_full_of_bits = CH$RCHAR_A (in_addr);
;  319          0715  4          INCR bit_counter FROM 0 TO 7 DO
;  320          0716  5              BEGIN
;  321          0717  5
;  322          0718  5              BIND
;  323          0719  5                  flags = cptr [0] : REF BITVECTOR,
;  324          0720  5                  nodes = cptr [1] : REF VECTOR [, BYTE],
;  325          0721  5                  next = cptr [2] : REF VECTOR [, WORD];
;  326          0722  5
;  327          0723  5              IF .byte_full_of_bits [.bit_counter]
;  328          0724  5              THEN
;  329          0725  5                  offset = .offset + 1;
;  330          0726  5              IF NOT .flags [.offset]
;  331          0727  5              THEN
;  332          0728  6                  BEGIN
;  333          0729  6                  offset = .nodes [.offset];
;  334          0730  6                  IF (offset = .offset + .offset) EQL 0
;  335          0731  6                  THEN
;  336          0732  6                      RETURN dcx$_normal;
;  337          0733  6                  END
;  338          0734  5              ELSE IF .res_len LSS .out_bytes
;  339          0735  5              THEN
;  340          0736  6                  BEGIN
;  341          0737  6                  out_addr [.res_len] = .nodes [.offset];
;  342          0738  6                  res_len = .res_len + 1;
;  343          0739  6                  .res_addr = ..res_addr + 1;
;  344          0740  6                  IF .next NEQA 0
;  345          0741  6                  THEN
;  346          0742  6                      cptr = base_ptr [4 * .next [.nodes [.offset]]];
;  347          0743  6                  offset = 0;
;  348          0744  6                  END
;  349          0745  5              ELSE
;  350          0746  5                  RETURN dcx$_trunc;
;  351          0747  4              END;
;  352          0748  3          END;
;  353          0749  3      RETURN dcx$_invdata;
;  354          0750  3      END
;  355          0751  2  ELSE IF .in_bytes GTRU .out_bytes
;  356          0752  2  THEN
;  357          0753  3      BEGIN
;  358          0754  3      dcx$long_move (.out_bytes, .in_addr1, .out_addr1);
;  359          0755  3      .res_addr = .out_bytes;
;  360          0756  3      RETURN dcx$_trunc;
;  361          0757  3      END
;  362          0758  2  ELSE
;  363          0759  3      BEGIN
;  364          0760  3      dcx$long_move (.in_bytes, .in_addr1, .out_addr1);
;  365          0761  3      .res_addr = .in_bytes;
;  366          0762  3      RETURN dcx$_normal;
;  367          0763  2      END;
;  368          0764  1  END;                                   ! Of dcx$do_expansion_0
```

```
                                      03FC 00000 DCX$DO_EXPANSION_0:
                                                       .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9                      ; 0657
          50     04    AC         10    C1 00002       ADDL3   #16, CTX, R0                                      ; 0685
                             18   BC    D4 00007       CLRL    @RES_ADDR                                         ; 0687
                       50         60    D0 0000A       MOVL    (R0), R0                                          ; 0688
                             10   A0    B5 0000D       TSTW    16(R0)                                            ;
                                  6B    13 00010       BEQL    8$                                                ;
          56     04    AC         14    C1 00012       ADDL3   #20, CTX, R6                                      ; 0700
                       59    08   AC    D0 00017       MOVL    IN_ADDR1, IN_ADDR                                 ; 0703
                       54    10   AC    D0 0001B       MOVL    OUT_ADDR1, OUT_ADDR                               ; 0704
                                  57    D4 0001F       CLRL    RES_LEN                                           ; 0705
                                  52    D4 00021       CLRL    OFFSET                                            ; 0706
                       50         66    D0 00023       MOVL    (R6), CPTR                                        ; 0707
                       55    0C   AC    D0 00026       MOVL    IN_BYTES, BYTE_COUNTER                            ; 0708
                                  46    11 0002A       BRB     7$                                                ;
                       58         89    90 0002C  1$:  MOVB    (IN_ADDR)+, BYTE_FULL_OF_BITS                     ; 0714
                                  53    D4 0002F       CLRL    BIT_COUNTER                                       ; 0715
          02           58         53    E1 00031  2$:  BBC     BIT_COUNTER, BYTE_FULL_OF_BITS, 3$                ; 0723
                                  52    D6 00035       INCL    OFFSET                                            ; 0725
          0C     00    B0         52    E0 00037  3$:  BBS     OFFSET, @0(CPTR), 4$                              ; 0726
                       52      04 B042  9A 0003C       MOVZBL  @4(CPTR)[OFFSET], OFFSET                          ; 0729
                       52         52    C0 00041       ADDL2   OFFSET, OFFSET                                    ; 0730
                                  28    12 00044       BNEQ    6$                                                ;
                                  6C    11 00046       BRB     11$                                               ; 0732
                 14    AC         57    D1 00048  4$:  CMPL    RES_LEN, OUT_BYTES                                ; 0734
                                  4A    18 0004C       BGEQ    9$                                                ;
                       51      04 B042  9A 0004E       MOVZBL  @4(CPTR)[OFFSET], R1                              ; 0737
                     8744         51    90 00053       MOVB    R1, (RES_LEN)+[OUT_ADDR]                          ;
                             18   BC    D6 00057       INCL    @RES_ADDR                                         ; 0739
                             08   A0    D5 0005A       TSTL    8(CPTR)                                           ; 0740
                                  0D    13 0005D       BEQL    5$                                                ;
                       51      08 B041  3C 0005F       MOVZWL  @8(CPTR)[R1], R1                                  ; 0742
                       51         04    C4 00064       MULL2   #4, R1                                            ;
                       50      00 B641  DE 00067       MOVAL   @0(R6)[R1], CPTR                                  ;
                                  52    D4 0006C  5$:  CLRL    OFFSET                                            ; 0743
          BF           53         07    F3 0006E  6$:  AOBLEQ  #7, BIT_COUNTER, 2$                               ; 0715
                       B7         55    F4 00072  7$:  SOBGEQ  BYTE_COUNTER, 1$                                  ; 0708
                       50 00000000G     8F    D0 00075       MOVL    #DCX$_INVDATA, R0                          ; 0751
                                  04       0007C       RET                                                       ;
                 14    AC    0C   AC    D1 0007D  8$:  CMPL    IN_BYTES, OUT_BYTES                               ;
                                  1C    1B 00082       BLEQU   10$                                               ;
                       52    10   AC    D0 00084       MOVL    OUT_ADDR1, R2                                     ; 0754
                       51    08   AC    D0 00088       MOVL    IN_ADDR1, R1                                      ;
                       50    14   AC    D0 0008C       MOVL    OUT_BYTES, R0                                     ;
                            0000G       30 00090       BSBW    DCX$LONG_MOVE                                     ;
                 18    BC    14   AC    D0 00093       MOVL    OUT_BYTES, @RES_ADDR                              ; 0755
                       50 00000000G     8F    D0 00098  9$:  MOVL    #DCX$_TRUNC, R0                              ; 0756
                                  04       0009F       RET                                                       ;
                       52    10   AC    D0 000A0 10$:  MOVL    OUT_ADDR1, R2                                     ; 0760
                       51    08   AC    D0 000A4       MOVL    IN_ADDR1, R1                                      ;
                       50    0C   AC    D0 000A8       MOVL    IN_BYTES, R0                                      ;
                            0000G       30 000AC       BSBW    DCX$LONG_MOVE                                     ;
                 18    BC    0C   AC    D0 000AF       MOVL    IN_BYTES, @RES_ADDR                               ; 0761
                       50 00000000G     8F    D0 000B4 11$:  MOVL    #DCX$_NORMAL, R0                             ; 0762
                                  04       000BB       RET                                                       ; 0764
```

```
; Routine Size:  188 bytes,    Routine Base:  $CODE$ + 0191
```

```
;   370          0765  1 %SBTTL  'dcx$expand_done -- Release data expansion context'
;   371          0766  1
;   372          0767  1 GLOBAL ROUTINE dcx$expand_done (context_addr) =
;   373          0768  2 BEGIN
;   374          0769  2 !++
;   375          0770  2 !
;   376          0771  2 ! Release data expansion context
;   377          0772  2 !
;   378          0773  2 ! Inputs:
;   379          0774  2 !
;   380          0775  2 !       context_addr.mz.r       Address of context longword
;   381          0776  2 !
;   382          0777  2 ! Outputs:
;   383          0778  2 !
;   384          0779  2 !       context_addr.mz.r       Context block accumulates data
;   385          0780  2 !
;   386          0781  2 ! Return value:
;   387          0782  2 !
;   388          0783  2 !       status.wlc.v
;   389          0784  2 !
;   390          0785  2 !               dcx$_normal     All is well
;   391          0786  2 !               dcx$_invctx     Invalid context block
;   392          0787  2 !               dcx$_invmap     Invalid map structure
;   393          0788  2 !--
;   394          0789  2
;   395          0790  2 BIND
;   396          0791  2     ctx = .context_addr : REF BBLOCK,   ! address of context block
;   397          0792  2     exp = ctx [ctx$l_specific] : BBLOCK,
;   398          0793  2     dcxmap = ctx [ctx$l_map] : REF BBLOCK;      ! address of map
;   399          0794  2
;   400          0795  2 perform (dcx$ctx_check (.ctx, ctx$c_expnd));
;   401          0796  2 perform (dcx$free_vm (4 * 4 * .dcxmap [dcxmap$w_nsubs], .exp [exp$l_map_segs]));
;   402          0797  2 perform (dcx$free_vm (.ctx [ctx$l_size], .ctx));
;   403          0798  2 ctx = 0;                                  ! mark context as gone
;   404          0799  2 RETURN dcx$_normal;
;   405          0800  2
;   406          0801  1 END;                                      ! Of dcx$expand_done


                                        0004 00000      .ENTRY  DCX$EXPAND_DONE, Save R2                        ;  0767
                                52      04   BC   D0 00002  MOVL   @CONTEXT_ADDR, R2                            ;  0792
                                51           02   D0 00006  MOVL   #2, R1                                      ;  0795
                                50           52   D0 00009  MOVL   R2, R0                                      ;
                                        0000G 30 0000C      BSBW   DCX$CTX_CHECK                                ;
                                2D           50   E9 0000F  BLBC   STATUS, 1$                                  ;
                                        14   A2   DD 00012  PUSHL  20(R2)                                       ;  0796
                                50      10   A2   D0 00015  MOVL   16(R2), R0                                   ;
                                51      10   A0   3C 00019  MOVZWL 16(R0), R1                                  ;
                             7E         51        04   78 0001D  ASHL   #4, R1, -(SP)                           ;
                                    0000G CF             02   FB 00021  CALLS  #2, DCX$FREE_VM                   ;
                                16                  50   E9 00026  BLBC   STATUS, 1$                            ;
                                                    52   DD 00029  PUSHL  R2                                     ;  0797
                                                    62   DD 0002B  PUSHL  (R2)                                   ;
                                    0000G CF             02   FB 0002D  CALLS  #2, DCX$FREE_VM                   ;
```

```
                                 0A           50  E9 00032          BLBC    STATUS, 1$                                      ;
                                       04     BC  D4 00035          CLRL    @CONTEXT_ADDR                                   ; 0798
                                 50 00000000G 8F  D0 00038          MOVL    #DCX$_NORMAL, R0                                ; 0799
                                              04  0003F 1$:        RET                                                     ; 0801

; Routine Size:  64 bytes,    Routine Base:  $CODE$ + 024D
```

```
;  408          0802  1 END
;  409          0803  0 ELUDOM                                    ! Of module expand



;                                  PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $CODE$                         653  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                                Library Statistics
;
;                                          --------- Symbols ---------      Pages       Processing
;       File                               Total     Loaded    Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1        9776         11          0         581        00:01.0



;                                       COMMAND QUALIFIERS

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

; Size:         653 code + 0 data bytes
; Run Time:         00:15.9
; Elapsed Time:     00:49.8
; Lines/CPU Min:     3030
; Lexemes/CPU-Min: 24418
; Memory Used:  130 pages
; Compilation Complete
```

PREFIX
REQ

STACLINT
LIS

STATUS
LIS

ANALYZE
LIS

DCXMSG
LIS

STASTUB
LIS

DCXDEF
MDL

EXPAND
LIS

SYSOUTPUT
LIS

DCXPRVDEF
MDL

STATEMENT
LIS

DCX

COMPRESS
LIS

TRANSFER
LIS

SYMBOL
LIS

DCXSHR
MAP

SUBS
LIS