L 7

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

```
**FILE**ID**COMPRESS                                        L 12




  CCCCCCCC    000000    MM      MM  PPPPPPPP    RRRRRRRR    EEEEEEEEEE    SSSSSSSS    SSSSSSSS
  CCCCCCCC    000000    MM      MM  PPPPPPPP    RRRRRRRR    EEEEEEEEEE    SSSSSSSS    SSSSSSSS
CC          00      00  MMMM  MMMM  PP      PP  RR      RR  EE          SS          SS
CC          00      00  MMMM  MMMM  PP      PP  RR      RR  EE          SS          SS
CC          00      00  MM  MM  MM  PP      PP  RR      RR  EE          SS          SS
CC          00      00  MM  MM  MM  PP      PP  RR      RR  EE          SS          SS
CC          00      00  MM      MM  PPPPPPPP    RRRRRRRR    EEEEEEEE      SSSSSS      SSSSSS
CC          00      00  MM      MM  PPPPPPPP    RRRRRRRR    EEEEEEEE      SSSSSS      SSSSSS
CC          00      00  MM      MM  PP          RR  RR      EE                  SS          SS
CC          00      00  MM      MM  PP          RR  RR      EE                  SS          SS
CC          00      00  MM      MM  PP          RR    RR    EE                  SS          SS      ....
CC          00      00  MM      MM  PP          RR    RR    EE                  SS          SS      ....
  CCCCCCCC    000000    MM      MM  PP          RR      RR  EEEEEEEEEE  SSSSSSSS    SSSSSSSS        ....
  CCCCCCCC    000000    MM      MM  PP          RR      RR  EEEEEEEEEE  SSSSSSSS    SSSSSSSS        ....

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

```
0 MODULE dcx_compress (                         . Data compression routines
0                   LANGUAGE (BLISS32),
0                   IDENT = 'V04-000'
0                   ) =
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 !
1 !   FACILITY:
1 !
1 !       DCX -- Data Compression / Expansion Facility
1 !
1 !   ABSTRACT:
1 !
1 !       The Data Compression / Expansion procedures provide a general
1 !       method for reducing the storage requirement for a arbitrary data.
1 !
1 ! ENVIRONMENT:
1 !
1 !       VAX native, user mode.
1 !
1 !--
1 !
1 !
1 ! AUTHOR:  David Thiel
1 !
1 ! CREATION DATE: July, 1981
1 !
1 ! MODIFIED BY:
1 !
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$_trunc,                 ! Resultant data truncated
:    80          0479   1     dcx$_invdata,               ! Invalid character for bounded compression
:    81          0480   1     lib$_strtru;
:    82          0481   1
:    83          0482   1 FORWARD ROUTINE
:    84          0483   1     fill_cmpseg,                ! fill in cmpseg structure
:    85          0484   1     dcx$compress_init,          ! initialize for data compression
:    86          0485   1     dcx$compress_data,          ! compress data record
:    87          0486   1     dcx$do_compression_0 :      ! internal compress routine -- type 0
:    88          0487   1         lkg_do NOVALUE,
:    89          0488   1     dcx$compress_done;          ! delete compression context
```

```
1 %SBTTL  'fill_cmpseg - Fill in cmpseg structure'
1
1 ROUTINE fill_cmpseg (cmpseg : REF BBLOCK, offset, pos, bits : REF BITVECTOR, ptrs : REF VECTOR [, LONG]) =
2 BEGIN
2 !++
2 !
2 ! Fill in cmpseg structure from map.
2 !
2 ! Inputs:
2 !
2 !       cmpseg                  Address of structure to fill in
2 !       offset                  Current offset in map substructures
2 !       pos                     Number of bits decoded so far
2 !       bits                    Address of bitvector holding encoding
2 !       ptrs                    dcxsbm offset to cmpseg map
2 !
2 ! Outputs:
2 !
2 !       cmpseg                  Encoding strings allocated and pointers
2 !                               stored
2 !
2 ! Return value:
2 !
2 !       dcx$_normal             All is well
2 !       lib$_insvirmem          Error allocating memory
2 !--
2
2 BIND
2     dcxsbm = cmpseg [cmpseg$l_dcxsbm] : REF BBLOCK,
2     flags = .dcxsbm + .dcxsbm [dcxsbm$w_flags] : BITVECTOR,
2     nodes = .dcxsbm + .dcxsbm [dcxsbm$w_nodes] : VECTOR [, BYTE];
2
2 IF .flags [.offset]
2 THEN
3     BEGIN
3
3     BIND
3         code = cmpseg [cmpseg$l_code] : VECTOR [, LONG],
3         cmpseg_next = cmpseg [cmpseg$l_next] : VECTOR [, LONG];
3
3     LOCAL
3         bitv : REF VECTOR [, LONG],
3         char;
3
3     char = .nodes [.offset];
3     IF .dcxsbm [dcxsbm$w_next] NEQ 0
3     THEN
4         BEGIN
4
4         BIND
4             dcxsbm_next = .dcxsbm + .dcxsbm [dcxsbm$w_next] : VECTOR [, WORD];
4
4         cmpseg_next [.char] = .ptrs [.dcxsbm_next [.char - .dcxsbm [dcxsbm$b_min_char]]]
4         END
3     ELSE
3         cmpseg_next [.char] = .ptrs [0];
3     perform (dcx$get_vm (4 + (.pos+7)/8, code [.char]));
```

```
;  148          0546  3          bitv = .code [.char];
;  149          0547  3          bitv [0] = .pos;
;  150          0548  3          CH$MOVE ((.pos+7)/8, .bits, bitv [1]);
;  151          0549  3          END
;  152          0550  2     ELSE IF .nodes [.offset] EQL 0
;  153          0551  2     THEN
;  154          0552  3          BEGIN
;  155          0553  3
;  156          0554  3          BIND
;  157          0555  3              code = cmpseg [cmpseg$l_code] : VECTOR [, LONG];
;  158          0556  3
;  159          0557  3          LOCAL
;  160          0558  3              bitv : REF VECTOR [, LONG];
;  161          0559  3
;  162          0560  3          perform (dcx$get_vm (4 + (.pos+7)/8, code [dcx$c_eor]));
;  163          0561  3          bitv = .code [dcx$c_eor];
;  164          0562  3          bitv [0] = .pos;
;  165          0563  3          CH$MOVE ((.pos+7)/8, .bits, bitv [1]);
;  166          0564  3          END
;  167          0565  2     ELSE
;  168          0566  3          BEGIN
;  169          0567  3          bits [.pos] = 1;
;  170          0568  3          perform (fill_cmpseg (.cmpseg, 2*.nodes [.offset] + 1, .pos+1, .bits, .ptrs));
;  171          0569  3          bits [.pos] = 0;
;  172          0570  3          perform (fill_cmpseg (.cmpseg, 2*.nodes [.offset] + 0, .pos+1, .bits, .ptrs));
;  173          0571  2          END;
;  174          0572  2     RETURN dcx$_normal;
;  175          0573  2
;  176          0574  1     END;                        ! of fill_cmpseg


                                                                  .TITLE  DCX_COMPRESS
                                                                  .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$_TRUNC
                                                                  .EXTRN  DCX$_INVDATA, LIB$_STRTRU

                                                                  .PSECT  $CODE$,NOWRT,2

                                                 01FC 00000 FILL_CMPSEG:
                                                                  .WORD   Save R2,R3,R4,R5,R6,R7,R8                      ; 0491
                       57       04   AC   D0 00002                MOVL    CMPSEG, R7                                     ; 0517
                       51       0C   A7   D0 00006                MOVL    12(R7), R1                                     ; 0518
                       52       06   A1   3C 0000A                MOVZWL  6(R1), R2                                      ;
                       50       08   A1   3C 0000E                MOVZWL  8(R1), R0                                      ; 0519
                       50            51   C0 00012                ADDL2   R1, R0                                         ;
                       56       0C   AC   D0 00015                MOVL    POS, R6                                        ; 0545
                       58       10   AC   D0 00019                MOVL    BITS, R8                                       ; 0548
         5'          6241       08   AC   E1 0001D                BBC     OFFSET, (R2)[R1], 3$                           ; 0521
                       50       08 BC40   9A 00023                MOVZBL  @OFFSET[R0], CHAR                              ; 0533
         52            50            02   78 00028                ASHL    #2, CHAR, R2                                   ; 0541
                       53       0A   A1   3C 0002C                MOVZWL  10(R1), R3                                     ; 0534
                                     19   13 00030                BEQL    1$                                             ;
```

```
                          53            51  C0 00032        ADDL2    R1, R3                              ; 0539
                          54       02   A1  9A 00035        MOVZBL   2(R1), R4                           ; 0541
                          50            54  C2 00039        SUBL2    R4, R0                              ;
                          50          6340  3C 0003C        MOVZWL   (R3)[R0], R0                        ;
                                   10 A742  9F 00040        PUSHAB   16(R7)[R2]                          ;
                          9E       14 BC40  D0 00044        MOVL     @PTRS[R0], @(SP)+                   ;
                                        08  11 00049        BRB      2$                                  ;
                                   10 A742  9F 0004B 1$:    PUSHAB   16(R7)[R2]                          ; 0544
                          9E       14   BC  D0 0004F        MOVL     @PTRS, @(SP)+                       ;
                                 0410 C742  9F 00053 2$:    PUSHAB   1040(R7)[R2]                        ; 0545
                          53       07   A6  9E 00058        MOVAB    7(R6), R3                           ;
                          53            08  C6 0005C        DIVL2    #8, R3                              ;
                                   04   A3  9F 0005F        PUSHAB   4(R3)                               ;
                    0000G CF            02  FB 00062        CALLS    #2, DCX$GET_VM                      ;
                          78            50  E9 00067        BLBC     STATUS, 9$                          ;
                                 0410 C742  9F 0006A        PUSHAB   1040(R7)[R2]                        ; 0546
                          50            9E  D0 0006F        MOVL     @(SP)+, BITV                        ;
                                        27  11 00072        BRB      4$                                  ; 0547
                          54       08 BC40  9A 00074 3$:    MOVZBL   @OFFSET[R0], R4                     ; 0550
                                        2A  12 00079        BNEQ     5$                                  ;
                          52     0410   C7  9E 0007B        MOVAB    1040(R7), R2                        ; 0560
                                 0400   C2  9F 00080        PUSHAB   1024(R2)                            ;
                          53       07   A6  9E 00084        MOVAB    7(R6), R3                           ;
                          53            08  C6 00088        DIVL2    #8, R3                              ;
                                   04   A3  9F 0008B        PUSHAB   4(R3)                               ;
                    0000G CF            02  FB 0008E        CALLS    #2, DCX$GET_VM                      ;
                          4C            50  E9 00093        BLBC     STATUS, 9$                          ;
                          50     0400   C2  D0 00096        MOVL     1024(R2), BITV                      ; 0561
                          60            56  D0 0009B 4$:    MOVL     R6, (BITV)                          ; 0562
          04   A0         68            53  28 0009E        MOVC3    R3, (R8), 4(BITV)                   ; 0563
                          36            11     000A3        BRB      8$                                  ; 0550
               00         68            56  E2 000A5 5$:    BBSS     R6, (R8), 6$                        ; 0567
                                   14   AC  DD 000A9 6$:    PUSHL    PTRS                                ; 0568
                                        58  DD 000AC        PUSHL    R8                                  ;
                                   01   A6  9F 000AE        PUSHAB   1(R6)                               ;
               7E         54            01  78 000B1        ASHL     #1, R4, -(SP)                       ;
                                        6E  D6 000B5        INCL     (SP)                                ;
                                        57  DD 000B7        PUSHL    R7                                  ;
                    FF42  CF            05  FB 000B9        CALLS    #5, FILL_CMPSEG                     ;
                          21            50  E9 000BE        BLBC     STATUS, 9$                          ;
               00         68            56  E5 000C1        BBCC     R6, (R8), 7$                        ; 0569
                                   14   AC  DD 000C5 7$:    PUSHL    PTRS                                ; 0570
                                        58  DD 000C8        PUSHL    R8                                  ;
                                   01   A6  9F 000CA        PUSHAB   1(R6)                               ;
               7E         54            01  78 000CD        ASHL     #1, R4, -(SP)                       ;
                                        57  DD 000D1        PUSHL    R7                                  ;
                    FF28  CF            05  FB 000D3        CALLS    #5, FILL_CMPSEG                     ;
                          07            50  E9 000D8        BLBC     STATUS, 9$                          ;
                          50 00000000G  8F  D0 000DB 8$:    MOVL     #DCX$_NORMAL, R0                    ; 0572
                                            04 000E2 9$:    RET                                          ; 0574

; Routine Size:  227 bytes,    Routine Base:  $CODE$ + 0000
```

```
1 %SBTTL  'dcx$compress_init - Initialization for data compression'
1
1 GLOBAL ROUTINE dcx$compress_init (context_addr, map_addr) =
2 BEGIN
2 !++
2 !
2 ! Initialization for data compression.
2 ! Allocate and initialize context area.
2 !
2 ! Inputs:
2 !
2 !      context_addr.mz.r       Address of context longword
2 !      map_addr.ra.r           Address of map
2 !
2 ! Outputs:
2 !
2 !      context_addr.mz.r       Address of context block is stored
2 !
2 ! Return value:
2 !
2 !      status.wlc.v
2 !
2 !              dcx$_normal     All is well
2 !              dcx$_invmap     Invalid map structure
2 !              lib$_insvirmem  Error allocating memory
2 !--
2
2 BIND
2     ctx = .context_addr : REF BBLOCK,   ! address of context block
2     dcxmap = .map_addr : REF BBLOCK;    ! address of map
2
2 LOCAL
2     ptrs : REF VECTOR [, LONG],
2     cmp : REF BBLOCK,
2     cmpseg : REF BBLOCK,
2     dcxsbm : REF BBLOCK,
2     status : LONG;                      ! return status
2
2 ctx = 0;                                ! assume failure
2 perform (dcx$map_check (.dcxmap));      ! validate map
2 perform (dcx$get_vm (ctx$k_fixed_len + cmp$k_length, ctx));
2 ctx [ctx$l_size] = ctx$k_fixed_len + cmp$k_length;
2 ctx [ctx$b_type] = ctx$c_cmprs;
2 ctx [ctx$w_version] = ctx$c_version;
2 ctx [ctx$l_sanity] = ctx$c_sanity;
2 ctx [ctx$l_map] = .dcxmap;
2 cmp = ctx [ctx$l_specific];
2 cmp [cmp$l_flink] = cmp [cmp$l_blink] = cmp [cmp$q_queue];
2 perform (dcx$get_vm (4*.dcxmap [dcxmap$w_nsubs], ptrs));
2 dcxsbm = .dcxmap + .dcxmap [dcxmap$w_sub0];
2 INCR i FROM 0 TO .dcxmap [dcxmap$w_nsubs]-1 DO
3     BEGIN
3     perform (dcx$get_vm (cmpseg$k_length, ptrs [.i]));
3     cmpseg = .ptrs [.i];
3     insque (.cmpseg, .cmp [cmp$l_blink]);
3     cmpseg [cmpseg$l_size] = cmpseg$k_length;
3     cmpseg [cmpseg$l_dcxsbm] = .dcxsbm;
```

```
:   235          0632  3      dcxsbm = .dcxsbm + .dcxsbm [dcxsbm$w_size];
:   236          0633  2      END;
:   237          0634  2  cmpseg = .cmp [cmp$l_flink];
:   238          0635  2  WHILE .cmpseg NEQA cmp [cmp$q_queue] DO
:   239          0636  3      BEGIN
:   240          0637  3
:   241          0638  3      LOCAL
:   242          0639  3          bits : VECTOR [(dcx$c_chars-1+7)/8, BYTE];
:   243          0640  3
:   244          0641  3      CH$FILL (0, %ALLOCATION (bits), bits);
:   245          0642  3      bits [0] = 1;
:   246          0643  3      perform (fill_cmpseg (.cmpseg, 1, 1, bits, .ptrs));
:   247          0644  3      bits [0] = 0;
:   248          0645  3      perform (fill_cmpseg (.cmpseg, 0, 1, bits, .ptrs));
:   249          0646  3      cmpseg = .cmpseg [cmpseg$l_flink];
:   250          0647  2      END;
:   251          0648  2  perform (dcx$free_vm (4*.dcxmap [dcxmap$w_nsubs], .ptrs));
:   252          0649  2
:   253          0650  2  RETURN dcx$_normal;
:   254          0651  2
:   255          0652  1  END;                                              ! of dcx$compress_init



                                      03FC 00000         .ENTRY  DCX$COMPRESS_INIT, Save R2,R3,R4,R5,R6,R7,-  ; 0577
                                                                 R8,R9                                        ;
                  59      0000G  CF   9E 00002           MOVAB   DCX$GET_VM, R9                               ;
                  5E             24   C2 00007           SUBL2   #36, SP                                      ;
                  58         08  AC   D0 0000A           MOVL    MAP_ADDR, R8                                 ; 0604
                             04  BC   D4 0000E           CLRL    @CONTEXT_ADDR                                ; 0613
                  52             68   D0 00011           MOVL    (R8), R2                                     ; 0614
                  50             52   D0 00014           MOVL    R2, R0                                       ;
                              0000G   30 00017           BSBW    DCX$MAP_CHECK                                ;
                  5A             50   E9 0001A           BLBC    STATUS, 2$                                   ;
                             04  AC   DD 0001D           PUSHL   CONTEXT_ADDR                                 ; 0615
                                 1C   DD 00020           PUSHL   #28                                          ;
                  69             02   FB 00022           CALLS   #2, DCX$GET_VM                               ;
                  4F             50   E9 00025           BLBC    STATUS, 2$                                   ;
                  56         04  BC   D0 00028           MOVL    @CONTEXT_ADDR, R6                            ; 0616
                  86             1C   D0 0002C           MOVL    #28, (R6)+                                   ;
                  86             01   90 0002F           MOVB    #1, (R6)+                                    ; 0617
                             03  A6   B4 00032           CLRW    3(R6)                                        ; 0618
             07   A6 4F317C65    8F   D0 00035           MOVL    #1328643173, 7(R6)                           ; 0619
             0B   A6             52   D0 0003D           MOVL    R2, 11(R6)                                   ; 0620
                  56             0F   C0 00041           ADDL2   #15, CMP                                     ; 0621
             04   A6             56   D0 00044           MOVL    CMP, 4(CMP)                                  ; 0622
                  66             56   D0 00048           MOVL    CMP, (CMP)                                   ;
                                 5E   DD 0004B           PUSHL   SP                                           ; 0623
                  50         10  A2   3C 0004D           MOVZWL  16(R2), R0                                   ;
        7E        50             02   78 00051           ASHL    #2, R0, -(SP)                                ;
                  69             02   FB 00055           CALLS   #2, DCX$GET_VM                               ;
                  75             50   E9 00058           BLBC    STATUS, 5$                                   ;
                  53         12  A2   3C 0005B           MOVZWL  18(R2), DCXSBM                               ; 0624
                  53             52   C0 0005F           ADDL2   R2, DCXSBM                                   ;
                  54         10  A2   3C 00062           MOVZWL  16(R2), R4                                   ; 0625
```

```
                         52            01  CE 00066          MNEGL   #1, I                                   :
                         28            11  00069             BRB     3$                                      :
                               00 BE42 DF  0006B 1$:         PUSHAL  @PTRS[I]                                : 0627
                         7E       0814 8F  3C 0006F          MOVZWL  #2068, -(SP)                            :
                         69            02  FB 00074          CALLS   #2, DCX$GET_VM                          :
                         7A            50  E9 00077 2$:      BLBC    STATUS, 7$                              :
                         57    00 BE42 D0  0007A             MOVL    @PTRS[I], CMPSEG                        : 0628
                    04   B6            67  0E 0007F          INSQUE  (CMPSEG), @4(CMP)                       : 0629
                    08   A7       0814 8F  3C 00083          MOVZWL  #2068, 8(CMPSEG)                        : 0630
                    0C   A7            53  D0 00089          MOVL    DCXSBM, 12(CMPSEG)                      : 0631
                         50            63  3C 0008D          MOVZWL  (DCXSBM), R0                            : 0632
                         53            50  C0 00090          ADDL2   R0, DCXSBM                              :
               D4        52            54  F2 00093 3$:      AOBLSS  R4, I, 1$                               : 0625
                         57            66  D0 00097          MOVL    (CMP), CMPSEG                           : 0634
                         56            57  D1 0009A 4$:      CMPL    CMPSEG, CMP                             : 0635
                         39            13  0009D             BEQL    6$                                      :
     20        00        6E            00  2C 0009F          MOVC5   #0, (SP), #0, #32, BITS                 : 0641
                                  04   AE     000A4                                                          :
                    04   AE            01  90 000A6          MOVB    #1, BITS                                : 0642
                                       6E  DD 000AA          PUSHL   PTRS                                    : 0643
                                  08   AE  9F 000AC          PUSHAB  BITS                                    :
                                       01  DD 000AF          PUSHL   #1                                      :
                                       01  DD 000B1          PUSHL   #1                                      :
                                       57  DD 000B3          PUSHL   CMPSEG                                  :
                  FE63   CF            05  FB 000B5          CALLS   #5, FILL_CMPSEG                         :
                         37            50  E9 000BA          BLBC    STATUS, 7$                              :
                                  04   AE  94 000BD          CLRB    BITS                                    : 0644
                                       6E  DD 000C0          PUSHL   PTRS                                    : 0645
                                  08   AE  9F 000C2          PUSHAB  BITS                                    :
                                       01  DD 000C5          PUSHL   #1                                      :
                                       7E  D4 000C7          CLRL    -(SP)                                   :
                                       57  DD 000C9          PUSHL   CMPSEG                                  :
                  FE4D   CF            05  FB 000CB          CALLS   #5, FILL_CMPSEG                         :
                         21            50  E9 000D0 5$:      BLBC    STATUS, 7$                              :
                         57            67  D0 000D3          MOVL    (CMPSEG), CMPSEG                        : 0646
                                       C2  11 000D6          BRB     4$                                      : 0635
                                       6E  DD 000D8 6$:      PUSHL   PTRS                                    : 0648
                         50            68  D0 000DA          MOVL    (R8), R0                                :
                         51       10   A0  3C 000DD          MOVZWL  16(R0), R1                              :
               7E        51            02  78 000E1          ASHL    #2, R1, -(SP)                           :
                  0000G  CF            02  FB 000E5          CALLS   #2, DCX$FREE_VM                         :
                         07            50  E9 000EA          BLBC    STATUS, 7$                              :
                         50 00000000G  8F  D0 000ED          MOVL    #DCX$_NORMAL, R0                        : 0650
                                       04  000F4 7$:         RET                                             : 0652

; Routine Size:  245 bytes,    Routine Base:  $CODE$ + 00E3
```

```
1 %SBTTL  'dcx$compress_data - Compress data record'
1
1 GLOBAL ROUTINE dcx$compress_data (context_addr, in_rec : REF BBLOCK, out_rec : REF BBLOCK, out_len) =
2 BEGIN
2 !++
2 !
2 ! Compress data record
2 !
2 ! Inp ts:
2 !
2 !       context_addr.mz.r        Address of context longword
2 !       in_rec.rt.dx             Descriptor for input (text) data record
2 !       out_rec.wt.dx            Descriptor for output (text) data buffer
2 !
2 ! Outputs:
2 !
2 !       context_addr.mz.r        Context block accumulates data
2 !       out_rec.wt.dx            Buffer is filled with output record
2 !       out_len.wwu.r            Word in which to store length of
2 !                                output record (optional)
2 !
2 ! Return value:
2 !
2 !       status.wlc.v
2 !
2 !               dcx$_normal      All is well
2 !               dcx$_invctx      Invalid context block
2 !               dcx$_invmap      Invalid map
2 !--
2
2 BIND
2     ctx = .context_addr : REF BBLOCK,     ! context block
2     dcxmap = ctx [ctx$l_map] : REF BBLOCK,      ! map address
2     res_len = .out_len : WORD;             ! result length
2
2 LOCAL
2     in_len,                                ! input length (bytes)
2     in_addr,                               ! input data address
2     status2,
2     status;                                ! return status
2
2 BUILTIN
2     NULLPARAMETER;
2
2 perform (dcx$ctx_check (.ctx, ctx$c_cmprs));
2 perform (dcx$map_check (.dcxmap));
2 perform (lib$analyze_sdesc_r2 (.in_rec; status2, in_len, in_addr); .status2);
2
2 CASE .out_rec [dsc$b_class]
2     FROM MIN (dsc$k_class_z, dsc$k_class_s)
2     TO MAX (dsc$k_class_z, dsc$k_class_s)
2     OF
2 SET
2 [dsc$k_class_z, dsc$k_class_s]:
3     BEGIN
3
3     LOCAL
```

```
;  314         0710  3              result : LONG;
;  315         0711  3
;  316         0712  3          dcx$do_compression_0 (
;  317         0713  3              .ctx, .in_addr, .in_len, .out_rec [dsc$a_pointer], .out_rec [dsc$w_length]
;  318         0714  3              ; status, result);
;  319         0715  3          CH$FILL (%C' ', .out_rec [dsc$w_length] - .result, .out_rec [dsc$a_pointer]+.result);
;  320         0716  3          IF NOT NULLPARAMETER (4)
;  321         0717  3          THEN
;  322         0718  3              res_len = .result;
;  323         0719  2          END;
;  324         0720  2      [inrange, outrange]:
;  325         0721  3          BEGIN
;  326         0722  3
;  327         0723  3          LOCAL
;  328         0724  3              result : LONG,
;  329         0725  3              status1 : LONG,
;  330         0726  3              res_buf : VECTOR [65535, BYTE]; ! result buffer
;  331         0727  3
;  332         0728  3          dcx$do_compression_0 (
;  333         0729  3              .ctx, .in_addr, .in_len, res_buf, %ALLOCATION (res_buf)
;  334         0730  3              ; status, result);
;  335         0731  3          status1 = lib$scopy_r_dx (result, res_buf, .out_rec);
;  336         0732  3          IF .status1 EQL lib$_strtru
;  337         0733  3          THEN
;  338         0734  4              BEGIN
;  339         0735  4              IF NOT NULLPARAMETER (4)
;  340         0736  4              THEN
;  341         0737  4                  lib$analyze_sdesc_r2 (.out_rec; status, res_len);
;  342         0738  4              status1 = dcx$_trunc;
;  343         0739  4              END
;  344         0740  3          ELSE IF NOT NULLPARAMETER (4)
;  345         0741  3          THEN
;  346         0742  3              res_len = .result;
;  347         0743  3          IF NOT .status1 AND .status
;  348         0744  3          THEN
;  349         0745  3              status = .status1;
;  350         0746  2          END;
;  351         0747  2      TES;
;  352         0748  2      RETURN .status;
;  353         0749  2
;  354         0750  1      END;                                              ! Of dcx$compress_data



                                          01FC 00000          .ENTRY  DCX$COMPRESS_DATA, Save R2,R3,R4,R5,R6,R7,-  ; 0655
                                                                      R8                                           ;
                       58 00000000G   00   9E 00002           MOVAB   LIB$ANALYZE_SDESC_R2, R8                     ;
                       5E FFFEFFFC    EE   9E 00009           MOVAB   -65540(SP), SP                               ;
          52        04 BC             10   C1 00010           ADDL3   #16, @CONTEXT_ADDR, R2                       ; 0685
                       51             01   D0 00015           MOVL    #1, R1                                       ; 0697
                       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                                     ; 0698
                                  0000G 30 00025              BSBW    DCX$MAP_CHECK                                ;
```

```
                          06          50 E9 00028       BLBC    STATUS, 1$                      ;
                          50    08    AC D0 0002B       MOVL    IN_REC, R0                      ; 0699
                                      68 16 0002F       JSB     LIB$ANALYZE_SDESC_R2            ;
                          01          50 E8 00031  1$:  BLBS    STATUS, 2$                      ;
                                         04 00034       RET                                     ;
                          53    0C    AC D0 00035  2$:  MOVL    OUT_REC, R3                     ; 0701
                01        00    03    A3 8F 00039       CASEB   3(R3), #0, #1                   ;
                        0071        0071    0003E  3$:  .WORD   7$-3$,-                         ;
                                                                7$-3$
                          7E  FFFF    8F 3C 00042       MOVZWL  #65535, -(SP)                   ; 0728
                                08    AE 9F 00047       PUSHAB  RES_BUF                         ;
                                      51 DD 0004A       PUSHL   IN_LEN                          ; 0729
                                      52 DD 0004C       PUSHL   IN_ADDR                         ;
                                04    BC DD 0004E       PUSHL   @CONTEXT_ADDR                   ;
                     0000V CF         05 FB 00051       CALLS   #5, DCX$DO_COMPRESSION_0        ;
                          57          50 D0 00056       MOVL    R0, R7                          ;
                          6E          51 D0 00059       MOVL    R1, RESULT                      ; 0728
                                      53 DD 0005C       PUSHL   R3                              ; 0731
                                08    AE 9F 0005E       PUSHAB  RES_BUF                         ;
                                08    AE 9F 00061       PUSHAB  RESULT                          ;
                000000000G 00         03 FB 00064       CALLS   #3, LIB$SCOPY_R_DX              ;
                          54          50 D0 0006B       MOVL    R0, STATUS1                     ;
                000000000G 8F         54 D1 0006E       CMPL    STATUS1, #LIB$_STRTRU           ; 0732
                                      1F 12 00075       BNEQ    5$                              ;
                          04          6C 91 00077       CMPB    (AP), #4                        ; 0735
                                      11 1F 0007A       BLSSU   4$                              ;
                                10    AC D5 0007C       TSTL    16(AP)                          ;
                                      0C 13 0007F       BEQL    4$                              ;
                          50          53 D0 00081       MOVL    R3, R0                          ; 0737
                                      68 16 00084       JSB     LIB$ANALYZE_SDESC_R2            ;
                          57          50 D0 00086       MOVL    R0, R7                          ;
                       10 BC          51 D0 00089       MOVL    R1, @OUT_LEN                    ;
                          54 00000000G 8F D0 0008D  4$:  MOVL    #DCX$_TRUNC, STATUS1            ; 0738
                                      0E 11 00094       BRB     6$                              ; 0732
                          04          6C 91 00096  5$:  CMPB    (AP), #4                        ; 0740
                                      09 1F 00099       BLSSU   6$                              ;
                                10    AC D5 0009B       TSTL    16(AP)                          ;
                                      04 13 0009E       BEQL    6$                              ;
                       10 BC          6E B0 000A0       MOVW    RESULT, @OUT_LEN                ; 0742
                          3C          54 E8 000A4  6$:  BLBS    STATUS1, 8$                     ; 0743
                          39          57 E9 000A7       BLBC    STATUS, 8$                      ;
                          57          54 D0 000AA       MOVL    STATUS1, STATUS                 ; 0745
                                      34 11 000AD       BRB     8$                              ; 0701
                          7E          63 3C 000AF  7$:  MOVZWL  (R3), -(SP)                     ; 0713
                                04    A3 DD 000B2       PUSHL   4(R3)                           ;
                                      51 DD 000B5       PUSHL   IN_LEN                          ;
                                      52 DD 000B7       PUSHL   IN_ADDR                         ;
                                04    BC DD 000B9       PUSHL   @CONTEXT_ADDR                   ;
                     0000V CF         05 FB 000BC       CALLS   #5, DCX$DO_COMPRESSION_0        ;
                          57          50 D0 000C1       MOVL    R0, R7                          ;
                          56          51 D0 000C4       MOVL    R1, R6                          ;
                          50          63 3C 000C7       MOVZWL  (R3), R0                        ; 0715
                          50          56 C2 000CA       SUBL2   RESULT, R0                      ;
       50       20        6E          00 2C 000CD       MOVC5   #0, (SP), #32, R0, @4(R3)[RESULT] ;
                                04  B346    000D2                                               
                          04          6C 91 000D5       CMPB    (AP), #4                        ; 0716
                                      09 1F 000D8       BLSSU   8$                              ;
```

```
                                       10   AC  D5 000DA          TSTL    16(AP)                                              ;
                                            04  13 000DD          BEQL    8$                                                  ;
                             10   BC        56  B0 000DF          MOVW    RESULT, @OUT_LEN                                    ; 0718
                                  50        57  D0 000E3 8$:      MOVL    STATUS, R0                                          ; 0748
                                                04 000E6          RET                                                         ; 0750

; Routine Size:  231 bytes,    Routine Base:  $CODE$ + 01D8
```

```
1 %SBTTL  'dcx$do_compression_0 - Type 0 compression'
1
1 ROUTINE dcx$do_compression_0 (
1         ctx : REF BBLOCK, in_addr : REF VECTOR [, BYTE], in_bytes, out_addr, out_bytes
1         ; status, res_len) : lkg_do NOVALUE =
2 BEGIN
2 !++
2 !
2 ! Compress data record using type 0 compression.
2 !
2 ! Inputs:
2 !
2 !       ctx                     Address of context block
2 !       in_addr                 Address of input record
2 !       in_bytes                Length of input record
2 !       out_addr                Address of output buffer
2 !       out_bytes               Length of output buffer
2 !
2 ! Outputs:
2 !
2 !       ctx                     Context block accumulates data
2 !       status                  Status of operation
2 !       res_len                 Result length
2 !
2 ! Status value:
2 !
2 !       dcx$_normal             All is well
2 !       dcx$_trunc              Result buffer too small - output truncated
2 !--
2
2 BIND
2     dcxmap = ctx [ctx$l_map] : REF BBLOCK,        ! map address
2     cmp = ctx [ctx$l_specific] : BBLOCK;
2
2 IF .dcxmap [dcxmap$w_nsubs] EQL 0
2 THEN
3     BEGIN
3     dcx$long_move ((res_len = MINU (.in_bytes, .out_bytes)), .in_addr, .out_addr);
3     IF .in_bytes GTRU .out_bytes
3     THEN
3         RETURN status = dcx$_trunc
3     ELSE
3         RETURN status = dcx$_normal;
3     END
2 ELSE
3     BEGIN
3
3     LOCAL
3         cmpseg : REF BBLOCK,
3         outbits : LONG,
3         outptr : LONG;
3
3     outbits = .out_bytes * 8;
3     outptr = 0;
3     cmpseg = .cmp [cmp$l_flink];
3     DECR i FROM .in_bytes-1 TO 0 DO
4         BEGIN
```

```
;  413          0808  4
;  414          0809  4           BIND
;  415          0810  4               next = cmpseg [cmpseg$l_next] : VECTOR [, LONG],
;  416          0811  4               code = cmpseg [cmpseg$l_code] : VECTOR [, LONG],
;  417          0812  4               code_ptr = code [CH$RCHAR (.in_addr)] : REF VECTOR [, LONG];
;  418          0813  4
;  419          0814  4           LOCAL
;  420          0815  4               bit_src : REF VECTOR [, LONG],
;  421          0816  4               bit_count;
;  422          0817  4
;  423          0818  4           IF .code_ptr EQLA 0
;  424          0819  4           THEN
;  425          0820  5               BEGIN
;  426          0821  5               res_len = 0;
;  427          0822  5               RETURN status = dcx$_invdata;
;  428          0823  4               END;
;  429          0824  4           bit_count = .code_ptr [0];
;  430          0825  4           bit_src = code_ptr [1];
;  431          0826  4           IF (outbits = .outbits - .bit_count) LSS 0
;  432          0827  4           THEN
;  433          0828  5               BEGIN
;  434          0829  5               res_len = (.outbits+7)/8;
;  435          0830  5               RETURN status = dcx$_trunc;
;  436          0831  4               END;
;  437          0832  4           WHILE .bit_count GTRU 32 DO
;  438          0833  5               BEGIN
;  439          0834  5               (.out_addr) <.outptr, 32, 0> = .bit_src [0];
;  440          0835  5               bit_src = .bit_src + 4;
;  441          0836  5               outptr = .outptr + 32;
;  442          0837  5               bit_count = .bit_count - 32;
;  443          0838  4               END;
;  444          0839  4           (.out_addr) <.outptr, .bit_count, 0> = .bit_src [0];
;  445          0840  4           outptr = .outptr + .bit_count;
;  446          0841  4           cmpseg = .next [CH$RCHAR_A (in_addr)];
;  447          0842  4           IF .cmpseg EQLA 0
;  448          0843  4           THEN
;  449          0844  5               BEGIN
;  450          0845  5               res_len = 0;
;  451          0846  5               RETURN status = dcx$_invdata;
;  452          0847  4               END;
;  453          0848  3           END;
;  454          0849  3       IF true
;  455          0850  3       THEN
;  456          0851  4           BEGIN
;  457          0852  4
;  458          0853  4           BIND
;  459          0854  4               code = cmpseg [cmpseg$l_code] : VECTOR [, LONG],
;  460          0855  4               code_ptr = code [dcx$c_eor] : REF VECTOR [, LONG];
;  461          0856  4
;  462          0857  4           LOCAL
;  463          0858  4               bit_src : REF VECTOR [, LONG],
;  464          0859  4               bit_count;
;  465          0860  4
   466          0861  4           bit_count = .code_ptr [0];
;  467          0862  4           bit_src = code_ptr [1];
;  468          0863  4           IF (outbits = .outbits - .bit_count) LSS 0
;  469          0864  4           THEN
;  470        0865 5             BEGIN
;  471        0866 5             res_len = (.outbits+7)/8;
;  472        0867 5             RETURN status = dcx$_trunc;
;  473        0868 4             END;
;  474        0869 4          WHILE .bit_count GTRU 32 DO
;  475        0870 5             BEGIN
;  476        0871 5             (.out_addr) <.outptr, 32, 0> = .bit_src [0];
;  477        0872 5             bit_src = .bit_src + 4;
;  478        0873 5             outptr = .outptr + 32;
;  479        0874 5             bit_count = .bit_count - 32;
;  480        0875 4             END;
;  481        0876 4          (.out_addr) <.outptr, .bit_count, 0> = .bit_src [0];
;  482        0877 4          outptr = .outptr + .bit_count;
;  483        0878 3          END;
;  484        0879 3       res_len = (.outptr + 7)/8;
;  485        0880 3       RETURN status = dcx$_normal;
;  486        0881 2       END;
;  487        0882 2
;  488        0883 1 END;                                      ! Of dcx$do_compression_0



                                        03FC 00000 DCX$DO_COMPRESSION_0:
                                                              .WORD    Save R2,P3,R4,R5,R6,R7,R8,R9                        ; 0753
         50      04   AC          10  C1 00002                ADDL3    #16, CTX, RC                                        ; 0782
         54      04   AC          14  C1 00007                ADDL3    #20, CTX, R4                                        ; 0783
                      58    10    AC  D0 0000C                MOVL     OUT_ADDR, R8                                        ; 0788
                      50          60  DC 00010                MOVL     (R0), R0                                            ; 0785
                            10    A0  B5 00013                TSTW     16(R0)                                              ;
                                  25  12 00016                BNEQ     2$                                                  ;
                      50    0C    AC  D0 00018                MOVL     IN_BYTES, R0                                        ; 0788
                 14   AC          50  D1 0001C                CMPL     R0, OUT_BYTES                                       ;
                                  04  1B 00020                BLEQU    1$                                                  ;
                      50    14    AC  D0 00022                MOVL     OUT_BYTES, R0                                       ;
                      57          50  D0 00026 1$:            MOVL     R0, RES_LEN                                         ;
                      52          58  D0 00029                MOVL     R8, R2                                              ;
                      51    08    AC  D0 0002C                MOVL     IN_ADDR, R1                                         ;
                               0000G  30 00030                BSBW     DCX$LONG_MOVE                                       ;
                 14   AC    0C    AC  D1 00033                CMPL     IN_BYTES, OUT_BYTES                                 ; 0789
                                  7E  1A 00038                BGTRU    8$                                                  ;
                                00A5  31 0003A                BRW      11$                                                 ; 0793
         53      14   AC          03  78 0003D 2$:            ASHL     #3, OUT_BYTES, OUTBITS                              ; 0803
                                  52  D4 00042                CLRL     OUTPTR                                              ; 0804
                      54          64  D0 00044                MOVL     (R4), CMPSEG                                        ; 0805
                      59    0C    AC  D0 00047                MOVL     IN_BYTES, I                                         ; 0806
                                  51  11 0004B                BRB      7$                                                  ;
                      51    08    BC  9A 0004D 3$:            MOVZBL   @IN_ADDR, R1                                        ; 0812
                      51  0410  C441  D0 00051                MOVL     1040(CMPSEG)[R1], R1                                ; 0818
                                  3A  13 00057                BEQL     6$                                                  ;
                      56          81  D0 00059                MOVL     (R1)+, BIT_COUNT                                    ; 0824
                      53          56  C2 0005C                SUBL2    BIT_COUNT, OUTBITS                                  ; 0826
                                  0A  18 0005F                BGEQ     4$                                                  ;
                      55    07    A3  9E 00061                MOVAB    7(R3), R5                                           ; 0829
         57           55          08  C7 00065                DIVL3    #8, R5, RES_LEN                                     ;
                                  4D  11 00069                BRB      8$                                                  ; 0830
                               20               56  D1 0006B 4$:      CMPL     BIT_COUNT, #32                              ; 0832
                                                0D  1B 0006E          BLEQU    5$                                          ;
         68          20        52               81  F0 00070          INSV     (BIT_SRC)+, OUTPTR, #32, (R8)               ; 0834
                               52               20  C0 00075          ADDL2    #32, OUTPTR                                 ; 0836
                               56               20  C2 00078          SUBL2    #32, BIT_COUNT                              ; 0837
                                                EE  11 0007B          BRB      4$                                          ; 0832
         68          56        52               61  F0 0007D 5$:      INSV     (BIT_SRC), OUTPTR, BIT_COUNT, (R8)          ; 0839
                               52               56  C0 00082          ADDL2    BIT_COUNT, OUTPTR                           ; 0840
                               51          08   BC  9A 00085          MOVZBL   @IN_ADDR, R1                                ; 0841
                                           08   AC  D6 00089          INCL     IN_ADDR                                     ;
                               54          10 A441  D0 0008C          MOVL     16(CMPSEG)[R1], CMPSEG                      ;
                                                0B  12 00091          BNEQ     7$                                          ; 0842
                                                57  D4 00093 6$:      CLRL     RES_LEN                                     ; 0845
                               50 00000000G     8F  D0 00095          MOVL     #DCX$_INVDATA, STATUS                       ; 0846
                                                4B  11 0009C          BRB      12$                                         ;
                               AC               59  F4 0009E 7$:      SOBGEQ   I, 3$                                       ; 0806
                               51        0810   D4  D0 000A1          MOVL     @2064(CMPSEG), BIT_COUNT                    ; 0861
                     54      0810  C4           04  C1 000A6          ADDL3    #4, 2064(CMPSEG), BIT_SRC                   ; 0862
                               53               51  C2 000AC          SUBL2    BIT_COUNT, OUTBITS                          ; 0863
                                                10  18 000AF          BGEQ     9$                                          ;
                               53               07  C0 000B1          ADDL2    #7, R3                                      ; 0866
                     57        53               08  C7 000B4          DIVL3    #8, R3, RES_LEN                             ;
                               50 00000000G     8F  D0 000B8 8$:      MOVL     #DCX$_TRUNC, STATUS                         ; 0867
                                                28  11 000BF          BRB      12$                                         ;
                               20               51  D1 000C1 9$:      CMPL     BIT_COUNT, #32                              ; 0869
                                                0D  1B 000C4          BLEQU    10$                                         ;
         68          20        52               84  F0 000C6          INSV     (BIT_SRC)+, OUTPTR, #32, (R8)               ; 0871
                               52               20  C0 000CB          ADDL2    #32, OUTPTR                                 ; 0873
                               51               20  C2 000CE          SUBL2    #32, BIT_COUNT                              ; 0874
                                                EE  11 000D1          BRB      9$                                          ; 0869
         68          51        52               64  F0 000D3 10$:     INSV     (BIT_SRC), OUTPTR, BIT_COUNT, (R8)          ; 0876
                               52               51  C0 000D8          ADDL2    BIT_COUNT, OUTPTR                           ; 0877
                               52               07  C0 000DB          ADDL2    #7, R2                                      ; 0879
                     57        52               08  C7 000DE          DIVL3    #8, R2, RES_LEN                             ;
                               50 00000000G     8F  D0 000E2 11$:     MOVL     #DCX$_NORMAL, STATUS                        ; 0880
                               51               57  D0 000E9 12$:     MOVL     R7, R1                                      ; 0883
                                                    04 000EC          RET                                                  ;

; Routine Size:  237 bytes,    Routine Base:  $CODE$ + 02BF
```

```
1 %SBTTL  'dcx$compress_done -- Release data compression context'
1
1 GLOBAL ROUTINE dcx$compress_done (context_addr) =
2 BEGIN
2 !++
2 !
2 ! Release data compression context data record
2 !
2 ! Inputs:
2 !
2 !       context_addr.mz.r       Address of context longword
2 !
2 ! Outputs:
2 !
2 !       context_addr.mz.r       Context block accumulates data
2 !
2 ! Return value:
2 !
2 !       status.wlc.v
2 !
2 !               dcx$_normal     All is well
2 !               dcx$_invctx     Invalid context block
2 !               dcx$_invmap     Invalid map structure
2 !--
2
2 BIND
2     ctx = .context_addr : REF BBLOCK,   ! address of context block
2     cmp = ctx [ctx$l_specific] : BBLOCK;
2
2 LOCAL
2     cmpseg : REF BBLOCK;
2
2 perform (dcx$ctx_check (.ctx, ctx$c_cmprs));
2 WHILE NOT remque (.cmp [cmp$l_flink], cmpseg) DO
3     BEGIN
3
3     BIND
3         code = cmpseg [cmpseg$l_code] : VECTOR [, LONG];
3
3     DECR i FROM cmpseg$c_code-1 TO 0 DO
4         BEGIN
4
4         BIND
4             bits = code [.i] : REF VECTOR [, LONG];
4
4         IF .bits NEQA 0
4         THEN
5             BEGIN
5             perform (dcx$free_vm (4 + (.bits [0]+7)/8, bits [0]));
5             bits = 0;
4             END;
3         END;
3     perform (dcx$free_vm (.cmpseg [cmpseg$l_size], .cmpseg));
2     END;
2 perform (dcx$free_vm (.ctx [ctx$l_size], .ctx));
2 ctx = 0;                                ! mark context as gone
2 RETURN dcx$_normal;
```

```
;  547         0941  2
;  548         0942  1 END;                                        . Of dcx$compress_done



                                         007C 00000          .ENTRY  DCX$COMPRESS_DONE, Save R2,R3,R4,R5,R6          ; 0886
                         56     0000G  CF   9E 00002          MOVAB   DCX$FREE_VM, R6                                 ;
                         54        04  AC   D0 00007          MOVL    CONTEXT_ADDR, R4                                ; 0910
              55         64            14   C1 0000B          ADDL3   #20, (R4), R5                                   ; 0911
                         51            01   D0 0000F          MOVL    #1, R1                                          ; 0916
                         50            64   D0 00012          MOVL    (R4), R0                                        ;
                                    0000G   30 00015          BSBW    DCX$CTX_CHECK                                   ;
                                       35   11 00018          BRB     4$                                              ;
                         53        00  B5   0F 0001A 1$:      REMQUE  @0(R5), CMPSEG                                  ; 0917
                                       33   1D 0001E          BVS     5$                                              ;
                         52      0100  8F   3C 00020          MOVZWL  #256, I                                         ; 0923
                         50      0410 C342  D0 00025 2$:      MOVL    1040(CMPSEG)[I], R0                             ; 0929
                                       17   13 0002B          BEQL    3$                                              ;
                                       50   DD 0002D          PUSHL   R0                                              ; 0932
              50         60            07   C1 0002F          ADDL3   #7, (R0), R0                                    ;
                         50            08   C6 00033          DIVL2   #8, R0                                          ;
                                   04  A0   9F 00036          PUSHAB  4(R0)                                           ;
                         66            02   FB 00039          CALLS   #2, DCX$FREE_VM                                 ;
                         28            50   E9 0003C          BLBC    STATUS, 6$                                      ;
                                 0410 C342  D4 0003F          CLRL    1040(CMPSEG)[I]                                 ; 0933
                         DE            52   F4 00044 3$:      SOBGEQ  I, 2$                                           ; 0923
                                       53   DD 00047          PUSHL   CMPSEG                                          ; 0936
                                   08  A3   DD 00049          PUSHL   8(CMPSEG)                                       ;
                         66            02   FB 0004C          CALLS   #2, DCX$FREE_VM                                 ;
                         C8            50   E8 0004F 4$:      BLBS    STATUS, 1$                                      ;
                                            04 00052          RET                                                     ;
                                       64   DD 00053 5$:      PUSHL   (R4)                                            ; 0938
                                   00  B4   DD 00055          PUSHL   @0(R4)                                          ;
                         66            02   FB 00058          CALLS   #2, DCX$FREE_VM                                 ;
                         09            50   E9 0005B          BLBC    STATUS, 6$                                      ;
                                       64   D4 0005E          CLRL    (R4)                                            ; 0939
                         50 00000000G  8F   D0 00060          MOVL    #DCX$_NORMAL, R0                                ; 0940
                                            04 00067 6$:      RET                                                     ; 0942

; Routine Size:  104 bytes,    Routine Base:  $CODE$ + 03AC
;  550          0943  1 END                                       ! Of module compress
;  551          0944  0 ELUDOM



;                                  PSECT SUMMARY
;
;      Name                       Bytes                          Attributes
;
;  $CODE$                            1044  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          00:00.9



;                                   COMMAND QUALIFIERS

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

; Size:          1044 code + 0 data bytes
; Run Time:         00:22.6
; Elapsed Time:     01:08.4
; Lines/CPU Min:      2509
; Lexemes/CPU-Min: 23734
; Memory Used:  151 pages
; Compilation Complete
```

0074 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

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