

DDDDDDDDDDDDDD		CCCCCCCCCCCC	XXX		XXX
DDDDDDDDDDDDDD		CCCCCCCCCCCC	XXX		XXX
DDDDDDDDDDDDDD		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		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		XXX
DDDDDDDDDDDDDD		CCCCCCCCCCCC	XXX		XXX
DDDDDDDDDDDDDD		CCCCCCCCCCCC	XXX		XXX
DDDDDDDDDDDDDD		CCCCCCCCCCCC	XXX		XXX

```

CCCCCCCCC 000000 MM MM PPPPPPPP RRRRRRRR EEEEEEEEE SSSSSSSS SSSSSSSS
CCCCCCCCC 000000 MM MM PPPPPPPP RRRRRRRR EEEEEEEEE SSSSSSSS SSSSSSSS
CC          00          00 MMMM MMMM PP PP RR RR RR RR EE EE SS SS
CC          00          00 MMMM MMMM PP PP RR RR RR RR EE EE SS SS
CC          00          00 MM MM MM PP PP RR RR RR RR EE EE SS SS
CC          00          00 MM MM MM PP PP RR RR RR RR EE EE SS SS
CC          00          00 MM MM MM PPPPPPPP RRRRRRRR EEEEEEEEE SSSSSS SSSSSS
CC          00          00 MM MM MM PPPPPPPP RRRRRRRR EEEEEEEEE SSSSSS SSSSSS
CC          00          00 MM MM MM PP PP RR RR RR RR EE EE SS SS
CC          00          00 MM MM MM PP PP RR RR RR RR EE EE SS SS
CC          00          00 MM MM MM PP PP RR RR RR RR EE EE SS SS
CC          00          00 MM MM MM PP PP RR RR RR RR EE EE SS SS
CCCCCCCCC 000000 MM MM PP PP RR RR RR RR EEEEEEEEE SSSSSSSS SSSSSSSS
CCCCCCCCC 000000 MM MM PP PP RR RR RR RR EEEEEEEEE SSSSSSSS SSSSSSSS
...
...
...
...

LL          IIIIIII SSSSSSSS
LL          IIIIIII 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
LLLLLLLLLLL IIIIIII SSSSSSSS
LLLLLLLLLLL IIIIIII SSSSSSSS

```

```
1 0001 0 MODULE dcx_compress (
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 --
```

Declarations

```
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$copy_r_dx : ! General string copy
75 0474 1 ADDRESSING_MODE (GENERAL);
76 0475 1
77 0476 1 EXTERNAL LITERAL
78 0477 1 dcx$_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
```

fill_cmpseg - Fill in cmpseg structure

```

91 0489 1 %SBTTL 'fill_cmpseg - Fill in cmpseg structure'
92 0490 1
93 0491 1 ROUTINE fill_cmpseg (cmpseg : REF BBLOCK, offset, pos, bits : REF BITVECTOR, ptrs : REF VECTOR [, LONG]) =
94 0492 2 BEGIN
95 0493 2 ++
96 0494 2
97 0495 2 Fill in cmpseg structure from map.
98 0496 2
99 0497 2 Inputs:
100 0498 2
101 0499 2 cmpseg Address of structure to fill in
102 0500 2 offset Current offset in map substructures
103 0501 2 pos Number of bits decoded so far
104 0502 2 bits Address of bitvector holding encoding
105 0503 2 ptrs dcxsbm offset to cmpseg map
106 0504 2
107 0505 2 Outputs:
108 0506 2
109 0507 2 cmpseg Encoding strings allocated and pointers
110 0508 2 stored
111 0509 2
112 0510 2 Return value:
113 0511 2
114 0512 2 dcx$_normal All is well
115 0513 2 lib$_insvirmem Error allocating memory
116 0514 2 --
117 0515 2
118 0516 2 BIND
119 0517 2 dcxsbm = cmpseg [cmpseg$l_dcxsbm] : REF BBLOCK,
120 0518 2 flags = .dcxsbm + .dcxsbm [dcxsbm$w_flags] : BITVECTOR,
121 0519 2 nodes = .dcxsbm + .dcxsbm [dcxsbm$w_nodes] : VECTOR [, BYTE];
122 0520 2
123 0521 2 IF .flags [.offset]
124 0522 2 THEN
125 0523 2 BEGIN
126 0524 2
127 0525 2 BIND
128 0526 2 code = cmpseg [cmpseg$l_code] : VECTOR [, LONG],
129 0527 2 cmpseg_next = cmpseg [cmpseg$l_next] : VECTOR [, LONG];
130 0528 2
131 0529 2 LOCAL
132 0530 2 bitv : REF VECTOR [, LONG],
133 0531 2 char;
134 0532 2
135 0533 2 char = .nodes [.offset];
136 0534 2 IF .dcxsbm [dcxsbm$w_next] NEQ 0
137 0535 2 THEN
138 0536 2 BEGIN
139 0537 2
140 0538 2 BIND
141 0539 2 dcxsbm_next = .dcxsbm + .dcxsbm [dcxsbm$w_next] : VECTOR [, WORD];
142 0540 2
143 0541 2 cmpseg_next [.char] = .ptrs [.dcxsbm_next [.char] - .dcxsbm [dcxsbm$b_min_char]]
144 0542 2 END
145 0543 2 ELSE
146 0544 2 cmpseg_next [.char] = .ptrs [0];
147 0545 2 perform (dcx$get_vm (4 + (.pos+7)/8, code [.char]));
```

fill_cmpseg - Fill in cmpseg structure

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

.PSECT $CODE$, NOWRT, 2
```

01FC 00000 FILL_CMPSEG:

57	04	AC	D0	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8	: 0491
51	0C	A7	D0	00006	MOVL	CMPSEG, R7	: 0517
52	06	A1	3C	0000A	MOVL	12(R7), R1	: 0518
50	08	A1	3C	0000E	MOVZWL	6(R1), R2	
50		51	C0	00012	MOVZWL	8(R1), R0	: 0519
56	0C	AC	D0	00015	ADDL2	R1, R0	
58	10	AC	D0	00019	MOVL	POS, R6	: 0545
51	08	AC	E1	0001D	MOVL	BITS, R8	: 0548
50	08	BC40	9A	00023	BBC	OFFSET, (R2)[R1], 3\$	: 0521
52	50	02	78	00028	MOVZBL	@OFFSET[R0], CHAR	: 0533
53	0A	A1	3C	0002C	ASHL	#2, CHAR, R2	: 0541
		19	13	00030	MOVZWL	10(R1), R3	: 0534
					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], a(SP)+	:	
		08	11	00049	BRB	2\$	:	
	10	A742	9F	0004B	PUSHAB	16(R7)[R2]	: 0544	
9E	14	BC	D0	0004F	MOVL	@PTRS, a(SP)+	:	
	0410	C742	9F	00053	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	a(SP)+, BITV	:	
		27	11	00072	BRB	4\$	: 0547	
54	08	BC40	9A	00074	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	MOVL	R6, (BITV)	: 0562	
68		53	28	0009E	MOVC3	R3, (R8), 4(BITV)	: 0563	
		36	11	000A3	BRB	8\$	: 0550	
	68	56	E2	000A5	BBSS	R6, (R8), 6\$	: 0567	
		14	AC	DD	000A9	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	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	MOVL	#DCX\$_NORMAL, R0	: 0572
			04	000E2	RET		: 0574	

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

```
dcx$compress_init - Initialization for data com

0575 1 %SBTTL 'dcx$compress_init - Initialization for data compression'
0576 1
0577 1 GLOBAL ROUTINE dcx$compress_init (context_addr, map_addr) =
0578 2 BEGIN
0579 2 ++
0580 2
0581 2 Initialization for data compression.
0582 2 Allocate and initialize context area.
0583 2
0584 2 Inputs:
0585 2
0586 2 context_addr.mz.r Address of context longword
0587 2 map_addr.ra.r Address of map
0588 2
0589 2 Outputs:
0590 2
0591 2 context_addr.mz.r Address of context block is stored
0592 2
0593 2 Return value:
0594 2
0595 2 status.wlc.v
0596 2
0597 2 dcx$_normal All is well
0598 2 dcx$_invmap Invalid map structure
0599 2 lib$_insvirmem Error allocating memory
0600 2 --
0601 2
0602 2 BIND
0603 2 ctx = .context_addr : REF BBLOCK, ! address of context block
0604 2 dcxmap = .map_addr : REF BBLOCK; ! address of map
0605 2
0606 2 LOCAL
0607 2 ptrs : REF VECTOR [, LONG],
0608 2 cmp : REF BBLOCK,
0609 2 cmpseg : REF BBLOCK,
0610 2 dcxsbm : REF BBLOCK,
0611 2 status : LONG; ! return status
0612 2
0613 2 ctx = 0; ! assume failure
0614 2 perform (dcx$map_check (.dcxmap)); ! validate map
0615 2 perform (dcx$get_vm (ctx$fixed_len + cmp$length, ctx));
0616 2 ctx [ctx$size] = ctx$fixed_len + cmp$length;
0617 2 ctx [ctx$b_type] = ctx$c_cmprs;
0618 2 ctx [ctx$w_version] = ctx$c_version;
0619 2 ctx [ctx$l_sanity] = ctx$c_sanity;
0620 2 ctx [ctx$l_map] = .dcxmap;
0621 2 cmp = ctx [ctx$l_specific];
0622 2 cmp [cmp$l_flink] = cmp [cmp$l_blink] = cmp [cmp$q_queue];
0623 2 perform (dcx$get_vm (4*.dcxmap [dcxmap$w_nsubs], ptrs));
0624 2 dcxsbm = .dcxmap + .dcxmap [dcxmap$w_sub0];
0625 2 INCR i FROM 0 TO .dcxmap [dcxmap$w_nsubs]-1 DO
0626 2 BEGIN
0627 2 perform (dcx$get_vm (cmpseg$length, ptrs [.i]));
0628 2 cmpseg = .ptrs [.i];
0629 2 insque (.cmpseg, .cmp [cmp$l_blink]);
0630 2 cmpseg [cmpseg$l_size] = cmpseg$length;
0631 2 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$q_queue] DO
239 0636 2 BEGIN
240 0637 2
241 0638 2 LOCAL
242 0639 2 bits : VECTOR [(dcx$c_chars-1+7)/8, BYTE];
243 0640 2
244 0641 2 CH$FILL (0, %ALLOCATION (bits), bits);
245 0642 2 bits [0] = 1;
246 0643 2 perform (fill_cmpseg (.cmpseg, 1, 1, bits, .ptrs));
247 0644 2 bits [0] = 0;
248 0645 2 perform (fill_cmpseg (.cmpseg, 0, 1, bits, .ptrs));
249 0646 2 cmpseg = .cmpseg [cmpseg$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
	59	0000G	CF 9E 00002	MOVAB	R8,R9	
	5E		24 C2 00007	SUBL2	DCX\$GET_VM, R9	
	58	08	AC D0 0000A	MOVL	#36, SP	
		04	BC D4 0000E	MOVL	MAP_ADDR, R8	0604
	52		68 D0 00011	CLRL	@CONTEXT_ADDR	0613
	50		52 D0 00014	MOVL	(R8), R2	0614
		0000G	30 00017	MOVL	R2, R0	
	5A		50 E9 0001A	BSBW	DCX\$MAP_CHECK	
		04	AC DD 0001D	BLBC	STATUS, 2\$	
			1C DD 00020	PUSHL	CONTEXT_ADDR	0615
	69		02 FB 00022	PUSHL	#28	
	4F		50 E9 00025	CALLS	#2, DCX\$GET_VM	
	56	04	BC D0 00028	BLBC	STATUS, 2\$	
	86		1C D0 0002C	MOVL	@CONTEXT_ADDR, R6	0616
	86		01 90 0002F	MOVL	#28, (R6)+	
		03	A6 B4 00032	MOVB	#1, (R6)+	0617
			8F D0 00035	CLRW	3(R6)	0618
07	A6	4F317C65	52 D0 0003D	MOVL	#1328643173, 7(R6)	0619
08	A6		0F C0 00041	MOVL	R2, 11(R6)	0620
	56		56 D0 00044	ADDL2	#15, CMP	0621
04	A6		56 D0 00048	MOVL	CMP, 4(CMP)	0622
	66		5E DD 0004B	MOVL	CMP, (CMP)	
			A2 3C 0004D	PUSHL	SP	0623
	50	10	02 78 00051	MOVZWL	16(R2), R0	
7E	50		02 FB 00055	ASHL	#2, R0, -(SP)	
	69		50 E9 00058	CALLS	#2, DCX\$GET_VM	
	75		A2 3C 0005B	BLBC	STATUS, 5\$	
	53	12	52 C0 0005F	MOVZWL	18(R2), DCXSBM	0624
	53		A2 3C 00062	ADDL2	R2, DCXSBM	
	54	10		MOVZWL	16(R2), R4	0625

		52		01	CE	00066	MNEGL	#1, I		
				28	11	00069	BRB	3\$		
			00	BE	42	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	BE	42	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\$: AOBLS	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

```
257 0653 1 %SBTTL 'dcx$compress_data - Compress data record'
258 0654 1
259 0655 1 GLOBAL ROUTINE dcx$compress_data (context_addr, in_rec : REF BBLOCK, out_rec : REF BBLOCK, out_len) =
260 0656 2 BEGIN
261 0657 2 ++
262 0658 2
263 0659 2 Compress data record
264 0660 2
265 0661 2 Inputs:
266 0662 2
267 0663 2 context_addr.mz.r Address of context longword
268 0664 2 in_rec.rt.dx Descriptor for input (text) data record
269 0665 2 out_rec.wt.dx Descriptor for output (text) data buffer
270 0666 2
271 0667 2 Outputs:
272 0668 2
273 0669 2 context_addr.mz.r Context block accumulates data
274 0670 2 out_rec.wt.dx Buffer is filled with output record
275 0671 2 out_len.wwu.r Word in which to store length of
276 0672 2 output record (optional)
277 0673 2
278 0674 2 Return value:
279 0675 2
280 0676 2 status.wlc.v
281 0677 2
282 0678 2 dcx$normal All is well
283 0679 2 dcx$invctx Invalid context block
284 0680 2 dcx$invmap Invalid map
285 0681 2 --
286 0682 2
287 0683 2 BIND
288 0684 2 ctx = .context_addr : REF BBLOCK, ! context block
289 0685 2 dcxmap = ctx [ctx$l_map] : REF BBLOCK, ! map address
290 0686 2 res_len = .out_len : WORD; ! result length
291 0687 2
292 0688 2 LOCAL
293 0689 2 in_len, ! input length (bytes)
294 0690 2 in_addr, ! input data address
295 0691 2 status2,
296 0692 2 status; ! return status
297 0693 2
298 0694 2 BUILTIN
299 0695 2 NULLPARAMETER;
300 0696 2
301 0697 2 perform (dcx$ctx_check (.ctx, ctx$c_mprs));
302 0698 2 perform (dcx$map_check (.dcxmap));
303 0699 2 perform (lib$analyze_sdesc_r2 (.in_rec; status2, in_len, in_addr); .status2);
304 0700 2
305 0701 2 CASE .out_rec [dsc$b_class]
306 0702 2 FROM MIN (dsc$k_class_z, dsc$k_class_s)
307 0703 2 TO MAX (dsc$k_class_z, dsc$k_class_s)
308 0704 2 OF
309 0705 2 SET
310 0706 2 [dsc$k_class_z, dsc$k_class_s]:
311 0707 2 BEGIN
312 0708 2
313 0709 2 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 3 END;
324 0720 3 [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$_strtru
337 0733 3 THEN
338 0734 3 BEGIN
339 0735 3 IF NOT NULLPARAMETER (4)
340 0736 3 THEN
341 0737 3 lib$analyze_sdesc_r2 (.out_rec; status, res_len);
342 0738 3 status1 = dcx$_trunc;
343 0739 3 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 3 END;
351 0747 3 TES;
352 0748 3 RETURN .status;
353 0749 3
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	MOVAB	LIB\$ANALYZE_SDESC_R2, R8	
		5E	FFFEFFFC	MOVAB	-65540(SP), SP	
52	04	BC		ADDL3	#16, @CONTEXT_ADDR, R2	0685
		51		MOVL	#1, R1	0697
		50	04	MOVL	@CONTEXT_ADDR, R0	
			0000G	BSBW	DCX\$CTX_CHECK	
		0F	50	BLBC	STATUS, -1\$	
		50	62	MOVL	(R2), R0	0698
			0000G	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	BLBS	STATUS, 2\$		
				04	00034	RET			
01	53	0C	AC	D0	00035	MOVL	OUT_REC, R3		0701
	00	03	A3	8F	00039	CASEB	3(R3), #0, #1		
	0071		0071		0003E	.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		
00000000G	00		03	FB	00064	CALLS	#3, LIB\$SCOPY_R_DX		
	54		50	D0	0006B	MOVL	R0, STATUS1		
00000000G	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	MOVL	#DCX\$_TRUNC, STATUS1		0738
			0E	11	00094	BRB	6\$		0732
	04		6C	91	00096	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	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	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		
	6E		00	2C	000CD	MOVC5	#0, (SP), #32, R0, @4(R3)[RESULT]		
50		20	04	B346	000D2				
			6C	91	000D5	CMPB	(AP), #4		0716
	04		09	1F	000D8	BLSSU	8\$		

dcx\$compress_data - Compress data record

K 13
15-Sep-1984 23:41:25
14-Sep-1984 12:15:56

VAX-11 Bliss-32 V4.0-742 Page 12
DISK\$VMSMASTER:[DCX.SRC]COMPRESS.B32:1 (5)

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

0718
0748
0750

; Routine Size: 231 bytes, Routine Base: \$CODES + 01D8

dcx\$do_compression_0 - Type 0 compression

```

356 0751 1 %SBTTL 'dcx$do_compression_0 - Type 0 compression'
357 0752 1
358 0753 1 ROUTINE dcx$do_compression_0 (
359 0754 1   ctx : REF BBLOCK, in_addr : REF VECTOR [, BYTE], in_bytes, out_addr, out_bytes
360 0755 1   ; status, res_len) : lkg_do NOVALUE =
361 0756 2 BEGIN
362 0757 2 ++
363 0758 2
364 0759 2   Compress data record using type 0 compression.
365 0760 2
366 0761 2   Inputs:
367 0762 2
368 0763 2       ctx                Address of context block
369 0764 2       in_addr           Address of input record
370 0765 2       in_bytes         Length of input record
371 0766 2       out_addr        Address of output buffer
372 0767 2       out_bytes       Length of output buffer
373 0768 2
374 0769 2   Outputs:
375 0770 2
376 0771 2       ctx                Context block accumulates data
377 0772 2       status             Status of operation
378 0773 2       res_len          Result length
379 0774 2
380 0775 2   Status value:
381 0776 2
382 0777 2       dcx$_normal        All is well
383 0778 2       dcx$_trunc         Result buffer too small - output truncated
384 0779 2   --
385 0780 2
386 0781 2 BIND
387 0782 2   dcxmap = ctx [ctx$l_map] : REF BBLOCK,      ! map address
388 0783 2   cmp = ctx [ctx$l_specific] : BBLOCK;
389 0784 2
390 0785 2 IF .dcxmap [dcxmap$w_nsubs] EQL 0
391 0786 2 THEN
392 0787 2   BEGIN
393 0788 2     dcx$long_move ((res_len = MINU (.in_bytes, .out_bytes)), .in_addr, .out_addr);
394 0789 2     IF .in_bytes GTRU .out_bytes
395 0790 2     THEN
396 0791 2       RETURN status = dcx$_trunc
397 0792 2     ELSE
398 0793 2       RETURN status = dcx$_normal;
399 0794 2     END
400 0795 2 ELSE
401 0796 2   BEGIN
402 0797 2
403 0798 2   LOCAL
404 0799 2     cmpseg : REF BBLOCK,
405 0800 2     outbits : LONG,
406 0801 2     outptr : LONG;
407 0802 2
408 0803 2   outbits = .out_bytes * 8;
409 0804 2   outptr = 0;
410 0805 2   cmpseg = .cmp [cmp$l_flink];
411 0806 2   DECR 1 FROM .in_bytes-1 TO 0 DO
412 0807 2     BEGIN
```

```

413 0808 4
414 0809 4
415 0810 4
416 0811 4
417 0812 4
418 0813 4
419 0814 4
420 0815 4
421 0816 4
422 0817 4
423 0818 4
424 0819 4
425 0820 5
426 0821 5
427 0822 5
428 0823 4
429 0824 4
430 0825 4
431 0826 4
432 0827 4
433 0828 5
434 0829 5
435 0830 5
436 0831 4
437 0832 4
438 0833 5
439 0834 5
440 0835 5
441 0836 5
442 0837 5
443 0838 4
444 0839 4
445 0840 4
446 0841 4
447 0842 4
448 0843 4
449 0844 5
450 0845 5
451 0846 5
452 0847 4
453 0848 3
454 0849 3
455 0850 3
456 0851 4
457 0852 4
458 0853 4
459 0854 4
460 0855 4
461 0856 4
462 0857 4
463 0858 4
464 0859 4
465 0860 4
466 0861 4
467 0862 4
468 0863 4
469 0864 4

      BIND
      next = cmpseg [cmpseg$l_next] : VECTOR [, LONG],
      code = cmpseg [cmpseg$l_code] : VECTOR [, LONG],
      code_ptr = code [CH$RCHAR (.in_addr)] : REF VECTOR [, LONG];

      LOCAL
      bit_src : REF VECTOR [, LONG],
      bit_count;

      IF .code_ptr EQLA 0
      THEN
      BEGIN
      res_len = 0;
      RETURN status = dcx$_invdata;
      END;
      bit_count = .code_ptr [0];
      bit_src = code_ptr [1];
      IF (outbits = .outbits - .bit_count) LSS 0
      THEN
      BEGIN
      res_len = (.outbits+7)/8;
      RETURN status = dcx$_trunc;
      END;
      WHILE .bit_count GTRU 32 DO
      BEGIN
      (.out_addr) <.outptr, 32, 0> = .bit_src [0];
      bit_src = .bit_src + 4;
      outptr = .outptr + 32;
      bit_count = .bit_count - 32;
      END;
      (.out_addr) <.outptr, .bit_count, 0> = .bit_src [0];
      outptr = .outptr + .bit_count;
      cmpseg = .next [CH$RCHAR_A (in_addr)];
      IF .cmpseg EQLA 0
      THEN
      BEGIN
      res_len = 0;
      RETURN status = dcx$_invdata;
      END;
      END;
      IF true
      THEN
      BEGIN
      BIND
      code = cmpseg [cmpseg$l_code] : VECTOR [, LONG],
      code_ptr = code [dcx$_c_eor] : REF VECTOR [, LONG];

      LOCAL
      bit_src : REF VECTOR [, LONG],
      bit_count;

      bit_count = .code_ptr [0];
      bit_src = code_ptr [1];
      IF (outbits = .outbits - .bit_count) LSS 0
      THEN
```

```
dcx$do_compression_0 - Type 0 compression

: 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$DO_COMPRESSION_0:
50      04 AC      10 C1 00002      .WORD      Save R2,R3,R4,R5,R6,R7,R8,R9
54      04 AC      14 C1 00007      ADDL3      #16, CTX, RC
58      10 AC      10 AC D0 0000C      ADDL3      #20, CTX, R4
50      50      60 D0 00010      MOVL      OUT_ADDR, R8
10      A0 B5 00013      MOVL      (R0), R0
25      12 00016      TSTW      16(R0)
14      50      0C AC D0 00018      BNEQ      2$
14      AC      50 D1 0001C      MOVL      IN_BYTES, R0
50      04 1B 00020      CMPL      R0, OUT_BYTES
57      14      04 AC D0 00022      BLEQU     1$
52      50      50 D0 00026 1$:      MOVL      OUT_BYTES, R0
51      08 AC      58 D0 00029      MOVL      R0, RES_LEN
51      0000G 30 00030      MOVL      R8, R2
14      AC      0C AC D1 00033      MOVL      IN_ADDR, R1
7E      1A 00038      BSBW      DCX$LONG_MOVE
53      14 AC      7E AC D1 00033      CMPL      IN_BYTES, OUT_BYTES
54      00A5 31 0003A      BGTRU     8$
59      0C      03 78 0003D 2$:      BRW       11$
51      54      52 D4 00042      ASHL      #3, OUT_BYTES, OUTBITS
51      0C AC      64 D0 00044      CLRL      OUTPTR
51      08 BC      64 D0 00047      MOVL      (R4), CMPSEG
51      0410 C441 D0 00051      MOVL      IN_BYTES, 1
56      3A 13 00057      BRB       7$
53      81 D0 00059      MOVZBL   @IN_ADDR, R1
55      07 A3 9E 00061      MOVL      1040(CMPSEG)[R1], R1
55      08 C7 00065      BEQL      6$
57      55      0A 18 0005F      MOVL      (R1)+, BIT_COUNT
55      08 C7 00065      SUBL2     BIT_COUNT, OUTBITS
55      08 C7 00065      BGEQ      4$
55      08 C7 00065      MOVAB     7(R3), R5
55      08 C7 00065      DIVL3     #8, R5, RES_LEN
55      4D 11 00069      BRB       8$
```

DCX_COMPRESS
V04=000

dcx\$do_compression_0 - Type 0 compression

B 14
15-Sep-1984 23:41:25
14-Sep-1984 12:15:56

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCX.SRC]COMPRESS.B32;1
Page 16
(6)

		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	A4	41 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	1, 3\$	: 0806
		51	0810	D4	D0 000A1		MOVL	@2064(CMPSEG), BIT_COUNT	: 0861
	54	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, RT	: 0883
				04	000EC		RET		

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

```

: 490      0884 1 %SBTTL 'dcx$compress_done -- Release data compression context'
: 491      0885 1
: 492      0886 1 GLOBAL ROUTINE dcx$compress_done (context_addr) =
: 493      0887 2 BEGIN
: 494      0888 2 ++
: 495      0889 2
: 496      0890 2 Release data compression context data record
: 497      0891 2
: 498      0892 2 Inputs:
: 499      0893 2
: 500      0894 2 context_addr.mz.r      Address of context longword
: 501      0895 2
: 502      0896 2 Outputs:
: 503      0897 2
: 504      0898 2 context_addr.mz.r      Context block accumulates data
: 505      0899 2
: 506      0900 2 Return value:
: 507      0901 2
: 508      0902 2 status.wlc.v
: 509      0903 2
: 510      0904 2 dcx$_normal      All is well
: 511      0905 2 dcx$_invctx      Invalid context block
: 512      0906 2 dcx$_invmap      Invalid map structure
: 513      0907 2 --
: 514      0908 2
: 515      0909 2 BIND
: 516      0910 2   ctx = .context_addr : REF BBLOCK, ! address of context block
: 517      0911 2   cmp = ctx [ctx$l_specific] : BBLOCK;
: 518      0912 2
: 519      0913 2 LOCAL
: 520      0914 2   cmpseg : REF BBLOCK;
: 521      0915 2
: 522      0916 2 perform (dcx$ctx_check (.ctx, ctx$c_cmprs));
: 523      0917 2 WHILE NOT remque (.cmp [cmp$l_flink], cmpseg) DO
: 524      0918 2   BEGIN
: 525      0919 2
: 526      0920 2   BIND
: 527      0921 2     code = cmpseg [cmpseg$l_code] : VECTOR [, LONG];
: 528      0922 2
: 529      0923 2   DECR i FROM cmpseg$c_code-1 TO 0 DO
: 530      0924 2     BEGIN
: 531      0925 2
: 532      0926 2     BIND
: 533      0927 2       bits = code [.i] : REF VECTOR [, LONG];
: 534      0928 2
: 535      0929 2     IF .bits NEQA 0
: 536      0930 2     THEN
: 537      0931 2       BEGIN
: 538      0932 2         perform (dcx$free_vm (4 + (.bits [0]+7)/8, bits [0]));
: 539      0933 2         bits = 0;
: 540      0934 2       END;
: 541      0935 2     END;
: 542      0936 2   perform (dcx$free_vm (.cmpseg [cmpseg$l_size], .cmpseg));
: 543      0937 2   END;
: 544      0938 2 perform (dcx$free_vm (.ctx [ctx$l_size], .ctx));
: 545      0939 2 ctx = 0; ! mark context as gone
: 546      0940 2 RETURN dcx$_normal;
```

DCX_COMPRESS
V04=000

dcx\$compress_done -- Release data compression c

D 14
15-Sep-1984 23:41:25
14-Sep-1984 12:15:56

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCX.SRC]COMPRESS.B32;1 Page 18
(7)

: 547
: 548

0941 2
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
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		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

DCX COMPRESS
V04=000

dcx\$compress_done -- Release data compression c

E 14
15-Sep-1984 23:41:25
14-Sep-1984 12:15:56
! Of module compress

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCX.SRC]COMPRESS.B32;1 Page 19 (8)

: 550
: 551
0943 1 END
0944 0 ELUDOM

PSECT SUMMARY

:
: Name Bytes Attributes
: \$CODE\$ 1044 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)

Library Statistics

:
: File Total Symbols Loaded Percent Pages Mapped Processing 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