

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

! Data analysis routines

```
0001 0 MODULE dcx_analyze (  
0002 0     LANGUAGE (BLISS32),  
0003 0     IDENT = 'V04-000'  
0004 0 ) =  
0005 1 BEGIN  
0006 1  
0007 1  
0008 1 *****  
0009 1 *  
0010 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY  
0011 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.  
0012 1 *  ALL RIGHTS RESERVED.  
0013 1 *  
0014 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  
0015 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  
0016 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  
0017 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  
0018 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  
0019 1 *  TRANSFERRED.  
0020 1 *  
0021 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  
0022 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT  
0023 1 *  CORPORATION.  
0024 1 *  
0025 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  
0026 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.  
0027 1 *  
0028 1 *  
0029 1 *****  
0030 1  
0031 1 ++  
0032 1  
0033 1 FACILITY:  
0034 1  
0035 1     DCX -- Data Compression / Expansion Facility  
0036 1  
0037 1 ABSTRACT:  
0038 1  
0039 1     The Data Compression / Expansion procedures provide a general  
0040 1     method for reducing the storage requirement for a arbitrary data.  
0041 1  
0042 1 ENVIRONMENT:  
0043 1  
0044 1     VAX native, user mode.  
0045 1  
0046 1 --  
0047 1  
0048 1  
0049 1 AUTHOR: David Thiel  
0050 1  
0051 1 CREATION DATE: July, 1981  
0052 1  
0053 1 MODIFIED BY:  
0054 1  
0055 1     V03-001 DWT0078      David W. Thiel      22-Feb-1983  
0056 1     Add support for estimated size of data to be compressed.  
0057 1
```

DCX_ANALYZE
V04=000

H 9
15-Sep-1984 23:38:18
14-Sep-1984 12:15:55

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCX.SRC]ANALYZE.B32;1 Page 2 (1)

: 58 0058 1 !--

Declarations

```
60 0059 1 %SBTTL 'Declarations';
61 0060 1
62 0061 1 LIBRARY
63 0062 1 'sys$library:starlet'; ! System macros
64 0063 1 REQUIRE
65 0064 1 'prefix'; ! DCX macros
66 0207 1 REQUIRE
67 0208 1 'dcxdef'; ! DCX public structure definitions
68 0302 1 REQUIRE
69 0303 1 'dcxprvdef'; ! DCX private structure definitions
70 0469 1
71 0470 1 ! random tree insertion macro
72 0471 1
73 0472 1 ! format of tree entry:
74 0473 1
75 0474 1
76 0475 1
77 0476 1
78 0477 1
79 0478 1
80 0479 1
81 0480 1
82 0481 1 MACRO
83 M 0482 1 tree_insert (header_address, item_address) =
84 M 0483 1 BEGIN
85 M 0484 1
86 M 0485 1 BIND
87 M 0486 1 _it = (item_address) : VECTOR [, LONG];
88 M 0487 1
89 M 0488 1 LOCAL
90 M 0489 1 _h : LONG;
91 M 0490 1
92 M 0491 1 _it [0] = 0;
93 M 0492 1 _it [1] = 0;
94 M 0493 1 _it [2] = 0;
95 M 0494 1 _h = (header_address);
96 M 0495 1 WHILE _h NEQA 0 DO
97 M 0496 1 BEGIN
98 M 0497 1 IF _it [3] LSSU .VECTOR [.._h, 3]
99 M 0498 1 THEN
100 M 0499 1 _h = VECTOR [.._h, 0]
101 M 0500 1 ELSE-IF _it [3] GTRU .VECTOR [.._h, 3]
102 M 0501 1 THEN
103 M 0502 1 _h = VECTOR [.._h, 1]
104 M 0503 1 ELSE
105 M 0504 1 BEGIN
106 M 0505 1 _it [2] = .VECTOR [.._h, 2];
107 M 0506 1 _h = VECTOR [.._h, 2];
108 M 0507 1 EXITLOOP;
109 M 0508 1 END;
110 M 0509 1 END;
111 M 0510 1 _h = _it [0];
112 M 0511 1 ENDX,
113 M 0512 1 tree_least (header_address) =
114 M 0513 1 BEGIN
115 M 0514 1
116 M 0515 1 LOCAL
```

```

: 117 M 0516 1      _q : REF VECTOR [, LONG],
: 118 M 0517 1      _h : LONG;
: 119 M 0518 1
: 120 M 0519 1      h = (header_address);
: 121 M 0520 1      IF _h EQLA 0
: 122 M 0521 1      THEN
: 123 M 0522 1          0
: 124 M 0523 1      ELSE
: 125 M 0524 1          BEGIN
: 126 M 0525 1              WHILE .VECTOR [.._h, 0] NEQA 0 DO
: 127 M 0526 1                  _h = VECTOR [.._h, 0];
: 128 M 0527 1                  IF (_q = .VECTOR [.._h, 2]) NEQA 0
: 129 M 0528 1                      THEN
: 130 M 0529 1                          BEGIN
: 131 M 0530 1                              VECTOR [..._h, 2] = ._q [2];
: 132 M 0531 1                              _q [0]
: 133 M 0532 1                          END
: 134 M 0533 1                      ELSE
: 135 M 0534 1                          BEGIN
: 136 M 0535 1                              _q = VECTOR [..._h, 0];
: 137 M 0536 1                              _h = ._q [1];
: 138 M 0537 1                              _q [0]
: 139 M 0538 1                          END
: 140 M 0539 1                      END
: 141 M 0540 1          ENDX;
: 142 M 0541 1
: 143 M 0542 1      EXTERNAL ROUTINE
: 144 M 0543 1          dcx$ctx_check : lkg_ctx_check,      ! Check context block
: 145 M 0544 1          dcx$map_check : lkg_map_check,      ! Check map
: 146 M 0545 1          dcx$get_vm,      ! Allocate memory
: 147 M 0546 1          dcx$free_vm,      ! Deallocate memory
: 148 M 0547 1          lib$scopy_r_dx : ! General string copy
: 149 M 0548 1          ADDRESSING_MODE (GENERAL);
: 150 M 0549 1
: 151 M 0550 1      EXTERNAL LITERAL
: 152 M 0551 1          dcx$_invarg,
: 153 M 0552 1          dcx$_invctx,
: 154 M 0553 1          dcx$_invitem,
: 155 M 0554 1          dcx$_normal;
: 156 M 0555 1
: 157 M 0556 1      FORWARD ROUTINE
: 158 M 0557 1          process_item,      ! process item list
: 159 M 0558 1          make_seg,      ! make a segment
: 160 M 0559 1          dcx$analyze_init, ! initialize for data analysis
: 161 M 0560 1          dcx$analyze_data, ! process data record
: 162 M 0561 1          huffman_size,      ! compute size of Huffman encoded data
: 163 M 0562 1          remove_seg,      ! remove one segment
: 164 M 0563 1          eliminate_seg,      ! eliminate unprofitable segment
: 165 M 0564 1          dcx$make_map,      ! compute comp / exp function
: 166 M 0565 1          dcx$analyze_done; ! delete analysis context
```


process_item - Process Item List

```
168 0566 1 %SBTTL 'process_item - Process Item List'
169 0567 1
170 0568 1 ROUTINE process_item (anl : REF BBLOCK, list : REF VECTOR [, LONG]) =
171 0569 2 BEGIN
172 0570 2 ++
173 0571 2
174 0572 2 Compute size of Huffman Encoded Data
175 0573 2
176 0574 2 Inputs:
177 0575 2
178 0576 2 anl Address of anl structure
179 0577 2 list Address of item list
180 0578 2
181 0579 2 Outputs:
182 0580 2
183 0581 2 NONE
184 0582 2
185 0583 2 Return value:
186 0584 2
187 0585 2 status code
188 0586 2
189 0587 2 --
190 0588 2
191 0589 2 IF NOT .list [0]
192 0590 2 THEN
193 0591 2 RETURN dcx$ invitem; ! even length list
194 0592 2 INCR i FROM 1 TO .list [0] BY 2 DO
195 0593 2 BEGIN
196 0594 2 SELECTONE .list [.i] OF
197 0595 2 SET
198 0596 2
199 0597 2 [dcx$ bounded]:
200 0598 2 anl [anl$ v_bounded] = ..list [.i + 1];
201 0599 2
202 0600 2 [dcx$ one_pass]:
203 0601 2 anl [anl$ v_one_pass] = ..list [.i + 1];
204 0602 2
205 0603 2 [dcx$ est_records]:
206 0604 2 BEGIN
207 0605 2 anl [anl$ v_est_recs] = true;
208 0606 2 anl [anl$ l_est_d_recs] = ..list [.i + 1];
209 0607 2 END;
210 0608 2
211 0609 2 [dcx$ est_bytes]:
212 0610 2 BEGIN
213 0611 2 anl [anl$ v_est_bytes] = true;
214 0612 2 anl [anl$ l_est_d_bytes] = ..list [.i + 1];
215 0613 2 END;
216 0614 2
217 0615 2 [dcx$ list]:
218 0616 2 perform (process_item (.anl, .list [.i + 1]));
219 0617 2
220 0618 2 TES;
221 0619 2 END;
222 0620 2 RETURN dcx$ _normal;
223 0621 2
224 0622 1 END; ! Of process_item
```

.TITLE DCX_ANALYZE
.IDENT \V04-000\

.EXTRN LIB\$ANALYZE_SDESC_R2
.EXTRN DCX\$CTX_CHECK, DCX\$MAP_CHECK
.EXTRN DCX\$GET_VM, DCX\$FREE_VM
.EXTRN LIB\$SCOPY_R_DX, DCX\$-INVARG
.EXTRN DCX\$-INVCTX, DCX\$-INVITEM
.EXTRN DCX\$-NORMAL

.PSECT \$CODE\$,NOWRT,2

		000C 00000		PROCESS_ITEM:					
		53	08	AC	DO	00002	.WORD	Save R2,R3	0568
		08		63	EB	00006	MOVL	LIST, R3	0589
		50	00000000G	8F	DO	00009	BLBS	(R3), 1\$	
					04	00010	MOVL	#DCX\$-INVITEM, R0	0591
		52		01	CE	00011	RET		
				7A	11	00014	MNEGL	#1, I	0592
		50		6342	DO	00016	BRB	7\$	
	00000101	8F		50	D1	0001A	MOVL	(R3)[I], R0	0594
				0D	12	00021	CMPL	R0, #257	0597
04	BC	50	04	A342	DO	00023	BNEQ	3\$	
		00		60	FO	00028	MOVL	4(R3)[I], R0	0598
				60	11	0002E	INSV	(R0), #0, #1, @ANL	
	00000102	8F		50	D1	00030	BRB	7\$	
				0D	12	00037	CMPL	R0, #258	0600
		50	04	A342	DO	00039	BNEQ	4\$	
04	BC	01		60	FO	0003E	MOVL	4(R3)[I], R0	0601
				4A	11	00044	INSV	(R0), #1, #1, @ANL	
	00000201	8F		50	D1	00046	BRB	7\$	
				12	12	0004D	CMPL	R0, #513	0603
		51	04	AC	DO	0004F	BNEQ	5\$	
		61		08	88	00053	MOVL	ANL, R1	0605
		50	04	A342	DO	00056	BISB2	#8, (R1)	
10		A1		60	DO	0005B	MOVL	4(R3)[I], R0	0606
				2F	11	0005F	MOVL	(R0), 16(R1)	
	00000202	8F		50	D1	00061	BRB	7\$	0594
				12	12	00068	CMPL	R0, #514	0609
		50	04	AC	DO	0006A	BNEQ	6\$	
		60		04	88	0006E	MOVL	ANL, R0	0611
		51	04	A342	DO	00071	BISB2	#4, (R0)	
0C		A0		61	DO	00076	MOVL	4(R3)[I], R1	0612
				14	11	0007A	MOVL	(R1), 12(R0)	
		01		50	D1	0007C	BRB	7\$	0594
				0F	12	0007F	CMPL	R0, #1	0615
			04	A342	DD	00081	BNEQ	7\$	
			04	AC	DD	00085	PUSHL	4(R3)[I]	0616
	FF73	CF		02	FB	00088	PUSHL	ANL	
		0D		50	E9	0008D	CALLS	#2, PROCESS_ITEM	
FF80	52	02		63	F1	00090	BLBC	STATUS, 8\$	
		50	00000000G	8F	DO	00096	ACBL	(R3), #2, I, 2\$	0592
					04	0009D	MOVL	#DCX\$-NORMAL, R0	0620
							RET		0622

DCX_ANALYZE
V04=000

process_item - Process Item List

; Routine Size: 158 bytes, Routine Base: \$CODE\$ + 0000

M 9
15-Sep-1984 23:38:18
14-Sep-1984 12:15:55

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCX.SRC]ANALYZE.B32;1 Page 7 (3)

make_seg - Make a tree segment

```
226 0623 1 %SBTTL 'make_seg - Make a tree segment'
227 0624 1
228 0625 1 ROUTINE make_seg (anl : REF BBLOCK, parent_seg : REF BBLOCK, char : LONG) =
229 0626 2 BEGIN
230 0627 2 ++
231 0628 2
232 0629 2 Compute size of Huffman Encoded Data
233 0630 2
234 0631 2 Inputs:
235 0632 2
236 0633 2 anl Address of anl structure
237 0634 2 parent_seg Address of parent anlseg structure or 0
238 0635 2 char Transition character into segment
239 0636 2
240 0637 2 Outputs:
241 0638 2
242 0639 2 NONE
243 0640 2
244 0641 2 Return value:
245 0642 2
246 0643 2 status code
247 0644 2
248 0645 2 --
249 0646 2
250 0647 2 LOCAL
251 0648 2 anlseg : REF BBLOCK;
252 0649 2
253 0650 2 perform (dcx$get_vm (anlseg$k_length, anlseg));
254 0651 2 anl [anlseg$w_nsegs] = .anl [anlseg$w_nsegs] + 1;
255 0652 2 IF .parent_seg EQLA 0
256 0653 2 THEN
257 0654 2 anl [anlseg$b_depth] = 1
258 0655 2 ELSE IF .parent_seg [anlseg$b_depth] EQL .anl [anlseg$b_depth]
259 0656 2 THEN
260 0657 2 anl [anlseg$b_depth] = .anl [anlseg$b_depth] + 1;
261 0658 2 insque (.anlseg, .anl [anlseg$b_link]);
262 0659 2 IF .parent_seg NEQA 0
263 0660 2 THEN
264 0661 2 parent_seg [anlseg$w_sons] = .parent_seg [anlseg$w_sons] + 1;
265 0662 2 anlseg [anlseg$l_size] = anlseg$k_length;
266 0663 2 anlseg [anlseg$w_id] = 0;
267 0664 2 anlseg [anlseg$w_char] = .char;
268 0665 2 anlseg [anlseg$w_active] = 0;
269 0666 2 anlseg [anlseg$w_active_r] = 0;
270 0667 2 IF .parent_seg EQLA 0
271 0668 2 THEN
272 0669 2 anlseg [anlseg$b_depth] = 1
273 0670 2 ELSE
274 0671 2 anlseg [anlseg$b_depth] = 1 + .parent_seg [anlseg$b_depth];
275 0672 2 anlseg [anlseg$b_max_char] = %X'00';
276 0673 2 anlseg [anlseg$b_min_char] = %X'FF';
277 0674 2 anlseg [anlseg$b_escape] = 0;
278 0675 2 anlseg [anlseg$w_tent] = true;
279 0676 2 anlseg [anlseg$w_solid] = false;
280 0677 2 anlseg [anlseg$w_escape] = false;
281 0678 2 anlseg [anlseg$w_base] = true;
282 0679 2 anlseg [anlseg$w_unbounded] = NOT .anl [anlseg$w_bounded];
```


make_seg - Make a tree segment

```
283 0680 2 anlseg [anlseg$w_sons] = 0;
284 0681 2 anlseg [anlseg$w_max_code] = 0;
285 0682 2 anlseg [anlseg$w_mapseg_size] = 0;
286 0683 2 anlseg [anlseg$l_prev] = .parent_seg;
287 0684 2 anlseg [anlseg$l_comp_bits] = 0;
288 0685 2 anlseg [anlseg$l_adj_bits] = 0;
289 0686 2 anlseg [anlseg$l_chars] = 0;
290 0687 2 IF .parent_seg NEQA 0
291 0688 2 THEN
292 0689 2     CH$COPY (
293 0690 2         .parent_seg [anlseg$b_depth] - 1, parent_seg [anlseg$t_string],
294 0691 2         1, char,
295 0692 2         0,
296 0693 2         anlseg$s_string, anlseg [anlseg$t_string]
297 0694 2     );
298 0695 2 IF .parent_seg EQLA 0
299 0696 2 THEN
300 0697 2     BEGIN
301 0698 2         DECR index FROM anlseg$c_next - 1 TO 0 DO
302 0699 2             VECTOR [anlseg [anlseg$l_next], .index] = .anlseg;
303 0700 2     END
304 0701 2 ELSE
305 0702 2     BEGIN
306 0703 2         LOCAL
307 0704 2             seg : REF BBLOCK;
308 0705 2
309 0706 2         BIND
310 0707 2             pnext = parent_seg [anlseg$l_next] : VECTOR [, LONG],
311 0708 2             ptrs = anlseg [anlseg$l_next] : VECTOR [, LONG],
312 0709 2             ca = anlseg [anlseg$t_string] : VECTOR [, BYTE];
313 0710 2
314 0711 2         pnext [.char] = .anlseg;
315 0712 2         seg = .anl [anl$l_flink];
316 0713 2         WHILE .seg NEQA anl [anl$q_queue] DO
317 0714 2             BEGIN
318 0715 2                 BIND
319 0716 2                     sptrs = seg [anlseg$l_next] : VECTOR [, LONG],
320 0717 2                     snext = sptrs [.char] : REF BBLOCK;
321 0718 2
322 0719 2                     IF .seg EQLA .anlseg
323 0720 2                         THEN
324 0721 2                             0
325 0722 2                         ELSE IF .anlseg [anlseg$b_depth] EQL 2
326 0723 2                             THEN
327 0724 2                                 sptrs [.char] = .anlseg
328 0725 2                             ELSE IF .seg [anlseg$b_depth] LSS .anlseg [anlseg$b_depth]
329 0726 2                                 THEN
330 0727 2                                     0
331 0728 2                                 ELSE IF .snext [anlseg$b_depth] GEQ .anlseg [anlseg$b_depth]
332 0729 2                                     THEN
333 0730 2                                         0
334 0731 2                                     ELSE IF CH$EQL (
335 0732 2                                         .anlseg [anlseg$b_depth] - 2,
336 0733 2                                         anlseg [anlseg$t_string] + 1,
337 0734 2                                         .anlseg [anlseg$b_depth] - 2,
338 0735 2
339 0736 2
```


C 10
15-Sep-1984 23:38:18 VAX-11 Bliss-32 V4.0-742 Page 10
14-Sep-1984 12:15:55 DISK\$VMSMASTER:[DCX.SRC]ANALYZE.B32;1 (4)

! Of make_seg

OFFC 00000 MAKE_SEG:				WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	
	5E		0C C2 00002	SUBL2	#12, SP	0625
		08	AE 9F 00005	PUSHAB	ANLSEG	0650
0000G	7E	0840	8F 3C 00008	MOVZWL	#2112, -(SP)	
	CF		02 FB 0000D	CALLS	#2, DCX\$GET_VM	
	01		50 E8 00012	BLBS	STATUS, 1\$	
			04 00015	RET		
	50	04	AC 7D 00016 1\$:	MOVQ	ANL, R0	0651
		16	A0 B6 0001A	INCW	22(R0)	
			51 D5 0001D	TSTL	R1	0652
			06 12 0001F	BNEQ	2\$	
14	A0		01 90 00021	MOVB	#1, 20(R0)	0654
			0A 11 00025	BRB	3\$	
14	A0	14	A1 91 00027 2\$:	CMPB	20(R1), 20(R0)	0655
			03 12 0002C	BNEQ	3\$	
		14	A0 96 0002E	INCB	20(R0)	0657
1C	B0	08	BE 0E 00031 3\$:	INSQUE	@ANLSEG, @28(R0)	0658
	57	08	AC D0 00036	MOVL	PARENT_SEG, R7	0659
			51 D4 0003A	CLRL	R1	
			57 D5 0003C	TSTL	R7	
			05 13 0003E	BEQL	4\$	
			51 D6 00040	INCL	R1	
		1C	A7 B6 00042	INCW	28(R7)	0661
	56	08	AE D0 00045 4\$:	MOVL	ANLSEG, R6	0662
08	A6	0840	8F 3C 00049	MOVZWL	#2112, 8(R6)	
		0C	A6 B4 0004F	CLRW	12(R6)	0663
	59	0C	AC D0 00052	MOVL	CHAR, R9	0664
0E	A6		59 3C 00056	MOVZWL	R9, 14(R6)	
		12	A6 B4 0005A	CLRW	18(R6)	0666
		04	AE D4 0005D	CLRL	4(SP)	0667
			57 D5 00060	TSTL	R7	
			09 12 00062	BNEQ	5\$	
		04	AE D6 00064	INCL	4(SP)	
14	A6		01 90 00067	MOVB	#1, 20(R6)	0669
			06 11 0006B	BRB	6\$	
14	A7		01 81 0006D 5\$:	ADDB3	#1, 20(R7), 20(R6)	0671

52	68	15	A6	FF	8F	9B	00073	6\$:	MOVZBW	#255, 21(R6)	0673
				17	A6	94	00078		CLRB	23(R6)	0674
			50	18	A6	9E	0007B		MOVAB	24(R6), R0	0675
			60		01	88	0007F		BISB2	#1, (R0)	
			60		0A	8A	00082		BICB2	#10, (R0)	0677
			60		10	88	00085		BISB2	#16, (R0)	0678
			58	04	AC	D0	00088		MOVL	ANL, R8	0679
			01		00	EF	0008C		EXTZV	#0, #1, (R8), R2	
			52		52	92	00091		MCOMB	R2, R2	
60	01		05		52	F0	00094		INSV	R2, #5, #1, (R0)	
				1A	A6	D4	00099		CLRL	26(R6)	0681
				1E	A6	B4	0009C		CLRW	30(R6)	0682
		20	A6		57	D0	0009F		MOVL	R7, 32(R6)	0683
				24	A6	7C	000A3		CLRQ	36(R6)	0684
			23	2C	A6	D4	000A6		CLRL	44(R6)	0686
			6E		51	E9	000A9		BLBC	R1, 7\$	0687
				14	A7	9A	000AC		MOVZBL	20(R7), (SP)	0690
			5B		6E	D7	000B0		DECL	(SP)	
			5A		08	D0	000B2		MOVL	#8, R11	
5B	00	30	A7	30	A6	9E	000B5		MOVAB	48(R6), R10	0693
					6E	2C	000B9		MOVC5	(SP), 48(R7), #0, R11, (R10)	
					6A		000BF				
					0D	18	000C0		BGEQ	7\$	
			5A		6E	C0	000C2		ADDL2	(SP), R10	
5B	00	0C	5B		6E	C2	000C5		SUBL2	(SP), R11	
			AC		01	2C	000C8		MOVC5	#1, CHAR, #0, R11, (R10)	
					6A		000CE				
			5A	043C	C6	9E	000CF	7\$:	MOVAB	1084(R6), R10	0699
			0E	04	AE	E9	000D4		BLBC	4(SP), 9\$	
			50	0100	8F	3C	000D8		MOVZWL	#256, INDEX	0698
		6A40	F9		56	D0	000DD	8\$:	MOVL	R6, (R10)[INDEX]	0699
					50	F4	000E1		SOBGEQ	INDEX, 8\$	
					78	11	000E4		BRB	16\$	0695
		043C	C749		56	D0	000E6	9\$:	MOVL	R6, 1084(R7)[R9]	0712
			54	18	A8	D0	000EC		MOVL	24(R8), SEG	0713
			50	18	A8	9E	000F0	10\$:	MOVAB	24(R8), R0	0714
			50		54	D1	000F4		CMPL	SEG, R0	
					3F	13	000F7		BEQL	13\$	
			55	043C	C449	DE	000F9		MOVAL	1084(SEG)[R9], R5	0719
			56		54	D1	000FF		CMPL	SEG, R6	0721
					2F	13	00102		BEQL	12\$	
			51	14	A6	9A	00104		MOVZBL	20(R6), R1	0724
			02		51	91	00108		CMPB	R1, #2	
					23	13	0010B		BEQL	11\$	
			51	14	A4	91	0010D		CMPB	20(SEG), R1	0727
					20	1F	00111		BLSSU	12\$	
			50		65	D0	00113		MOVL	(R5), R0	0730
			51	14	A0	91	00116		CMPB	20(R0), R1	
					17	1E	0011A		BGEQU	12\$	
			52	FE	A1	9E	0011C		MOVAB	-2(R1), R2	0734
			50	14	A4	9A	00120		MOVZBL	20(SEG), R0	0737
			50		51	C2	00124		SUBL2	R1, R0	
31 A044		31	A6		52	29	00127		CMPC3	R2, 49(R6), 49(R0)[SEG]	0733
					03	12	0012E		BNEQ	12\$	
			65		56	D0	00130	11\$:	MOVL	R6, (R5)	0740
			54		64	D0	00133	12\$:	MOVL	(SEG), SEG	0741
					B8	11	00136		BRB	10\$	0714

DCX_ANALYZE
V04=000

make_seg - Make a tree segment

E 10
15-Sep-1984 23:38:18
14-Sep-1984 12:15:55

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCX.SPC]ANALYZE.B32;1

Page 12
(4)

		54	18	A8	D0	00138	13\$:	MOVL	24(R8), SEG	:	0744
		52	14	A6	9A	0013C		MOVZBL	20(R6), R2	:	0745
		52		02	C2	00140		SUBL2	#2, R2	:	
				51	D4	00143		CLRL	INDEX	:	
				0B	11	00145		BRB	15\$	:	
		50	30	A641	9A	00147	14\$:	MOVZBL	48(R6)[INDEX], R0	:	0746
		54	043C	C440	D0	0014C		MOVL	1084(SEG)[R0], SEG	:	
F1		51		52	F3	00152	15\$:	AOBLEQ	R2, INDEX, 14\$	:	
6A	043C	C4	0404	8F	28	00156		MOVCL	#1028, 1084(SEG), (R10)	:	0747
		50	00000000G	8F	D0	0015E	16\$:	MOVL	#DCX\$_NORMAL, R0	:	0749
					04	00165		RET		:	0751

; Routine Size: 358 bytes, Routine Base: \$CODE\$ + 009E


```
0752 1 %SBTTL 'dcx$analyze_init - Initialization for data analysis'
0753 1
0754 1 GLOBAL ROUTINE dcx$analyze_init (context_addr, item, value) =
0755 2 BEGIN
0756 2 ++
0757 2
0758 2 Initialization for data analysis.
0759 2 Allocate and initialize context area.
0760 2
0761 2 Inputs:
0762 2
0763 2 context_addr.wz.r Address of context longword
0764 2 item.rl.r Item code (optional)
0765 2 value.rl.r Value associated with item (optional)
0766 2
0767 2 Outputs:
0768 2
0769 2 ccontext_addr Address of context block is stored
0770 2
0771 2 Return value:
0772 2
0773 2 status.wlc.v
0774 2
0775 2 dcx$_normal All is well
0776 2 dcx$_invitem Invalid item code or missing item value
0777 2 lib$_insvirmem Error allocating memory
0778 2 --
0779 2
0780 2 BIND
0781 2 ctx = .context_addr : REF BBLOCK; ! address of context block
0782 2
0783 2 LOCAL
0784 2 anl : REF BBLOCK, ! address of anl block
0785 2 anlseg : REF BBLOCK; ! address of anlseg block
0786 2
0787 2 BUILTIN
0788 2 ACTUALPARAMETER,
0789 2 ACTUALCOUNT,
0790 2 NULLPARAMETER;
0791 2
0792 2 perform (dcx$get_vm (ctx$k_fixed_len + anl$k_length, ctx));
0793 2 ctx [ctx$l_size] = ctx$k_fixed_len + anl$k_length;
0794 2 ctx [ctx$b_type] = ctx$c_anlyz;
0795 2 ctx [ctx$w_version] = ctx$c_version;
0796 2 ctx [ctx$l_sanity] = ctx$c_sanity;
0797 2 anl = ctx [ctx$l_specific];
0798 2 anl [anl$v_bounded] = false;
0799 2 anl [anl$v_one_pass] = false;
0800 2 anl [anl$v_est_bytes] = false;
0801 2 anl [anl$v_est_recs] = false;
0802 2 anl [anl$l_est_d_bytes] = 0;
0803 2 anl [anl$l_est_d_recs] = 0;
0804 2 anl [anl$l_d_bytes] = 0;
0805 2 anl [anl$l_d_recs] = 0;
0806 2 anl [anl$b_depth] = 0;
0807 2 anl [anl$w_nsegs] = 0;
0808 2 anl [anl$l_flink] = anl [anl$l_blink] = anl [anl$q_queue];
```

dcx\$analyze_init - Initialization for data anal

```

: 413 0809 2 IF NOT ACTUALCOUNT ( )
: 414 0810 THEN
: 415 0811 RETURN dcx$ invitem; ! even number of arguments
: 416 0812 INCR i FROM 2 TO ACTUALCOUNT ( ) BY 2 DO
: 417 0813 BEGIN
: 418 0814
: 419 0815 LOCAL
: 420 0816 list : VECTOR [3, LONG];
: 421 0817
: 422 0818 list [0] = 3;
: 423 0819 list [1] = .ACTUALPARAMETER (.i);
: 424 0820 list [2] = ACTUALPARAMETER (.i + 1);
: 425 0821 perform (process_item (.anl, list));
: 426 0822 END;
: 427 0823 perform (make_seg (.anl, 0, -1));
: 428 0824 DECR char FROM anlseg$c_next - 1 TO 0 DO
: 429 0825 perform (make_seg (.anl, .anl [anl$l_flink], .char));
: 430 0826 RETURN dcx$_normal;
: 431 0827
: 432 0828 1 END; ! of dcx$analyze_init
```

			001C 00000	.ENTRY	DCX\$ANALYZE_INIT, Save R2,R3,R4	: 0754
	5E		0C C2 00002	SUBL2	#12, SP	
		04	AC DD 00005	PUSHL	CONTEXT_ADDR	: 0792
			3C DD 00008	PUSHL	#60	
0000G	CF		02 FB 0000A	CALLS	#2, DCX\$GET_VM	
	73		50 E9 0000F	BLBC	STATUS, 4\$	
	52	04	BC D0 00012	MOVL	@CONTEXT_ADDR, R2	: 0793
	82		3C D0 00016	MOVL	#60, (R2)+	
			82 94 00019	CLRB	(R2)+	: 0794
		03	A2 B4 0001B	CLRW	3(R2)	: 0795
07	A2 4F317C65		8F D0 0001E	MOVL	#1328643173, 7(R2)	: 0796
	52		0F C0 00026	ADDL2	#15, ANL	: 0797
	62		0F 8A 00029	BICB2	#15, (ANL)	: 0801
		0C	A2 7C 0002C	CLRQ	12(ANL)	: 0802
		04	A2 7C 0002F	CLRQ	4(ANL)	: 0804
		14	A2 94 00032	CLRB	20(ANL)	: 0806
		16	A2 B4 00035	CLRW	22(ANL)	: 0807
	50	18	A2 9E 00038	MOVAB	24(ANL), R0	: 0808
1C	A2		50 D0 0003C	MOVL	R0, 28(ANL)	
18	A2		50 D0 00040	MOVL	R0, 24(ANL)	
	08		6C E8 00044	BLBS	(AP), 1\$	: 0809
	50 00000000G		8F D0 00047	MOVL	#DCX\$_INVITEM, R0	: 0811
			04 0004E	RET		
	54		6C 9A 0004F	MOVZBL	(AP), R4	: 0812
			53 D4 00052	CLRL	I	
			1D 11 00054	BRB	3\$	
	6E		03 D0 00056	MOVL	#3, LIST	: 0818
	50	6C43	D0 00059	MOVL	(AP)[1], R0	: 0819
04	AE		60 D0 0005D	MOVL	(R0), LIST+4	
08	AE	04 AC43	D0 00061	MOVL	4(AP)[1], LIST+8	: 0820
		4004	8F BB 00067	PUSHR	#*M<R2,SP>	: 0821
FD8C	CF		02 FB 0006B	CALLS	#2, PROCESS_ITEM	

; Routine Size: 167 bytes Routine Base: \$CODES + 0204

dcx\$analyze_data - Analyze data record

```

434 0829 1 %SBTTL 'dcx$analyze_data - Analyze data record'
435 0830 1
436 0831 1 GLOBAL ROUTINE dcx$analyze_data (context_addr, rec : REF BBLOCK) =
437 0832 2 BEGIN
438 0833 2 ++
439 0834 2
440 0835 2 Analyze data record
441 0836 2
442 0837 2 Inputs:
443 0838 2
444 0839 2 context_addr.mz.r Address of context longword
445 0840 2 rec.rt.dx Descriptor for (text) data record
446 0841 2
447 0842 2 Outputs:
448 0843 2
449 0844 2 context_addr.mz.r Context block accumulates data
450 0845 2
451 0846 2 Return value:
452 0847 2
453 0848 2 status.wlc.v
454 0849 2
455 0850 2 dcx$_normal All is well
456 0851 2 dcx$_invctx Invalid context block
457 0852 2 --
458 0853 2
459 0854 2 BIND
460 0855 2 ctx = .context_addr : REF BBLOCK;
461 0856 2
462 0857 2 LOCAL
463 0858 2 addr : REF VECTOR [, BYTE], ! address of data
464 0859 2 len, ! length of data
465 0860 2 anl : REF BBLOCK,
466 0861 2 anlseg : REF BBLOCK,
467 0862 2 status; ! return status
468 0863 2
469 0864 2 ! check context block
470 0865 2
471 0866 2 perform (dcx$ctx_check (.ctx, ctx$_anlyz));
472 0867 2
473 0868 2 ! get address of data record
474 0869 2
475 0870 2 perform (lib$analyze_sdesc_r2 (.rec; status, len, addr); .status);
476 0871 2
477 0872 2 ! accumulate statistical information
478 0873 2
479 0874 2 anl = ctx [ctx$_specific];
480 0875 2 anl [anl$_d_bytes] = .anl [anl$_d_bytes] + .len;
481 0876 2 anl [anl$_d_recs] = .anl [anl$_d_recs] + 1;
482 0877 2 anlseg = .anl [anl$_flink];
483 0878 2 DECR 1 FROM .len-1 TO 0 DO
484 0879 2 BEGIN
485 0880 2
486 0881 2 BIND
487 0882 2 count = anlseg [anlseg$_count] : VECTOR [, LONG],
488 0883 2 next = anlseg [anlseg$_next] : VECTOR [, LONG];
489 0884 2
490 0885 2 anlseg [anlseg$_chars] = .anlseg [anlseg$_chars] + 1;
```



```
dcx$analyze_data - Analyze data record

: 491      0886      3      count [.addr [0]] = .count [.addr [0]] + 1;
: 492      0887      3      anlseg = .next [.addr [0]];
: 493      0888      3      addr = .addr + 1;
: 494      0889      3      END;
: 495      0890      3      IF true
: 496      0891      3      THEN
: 497      0892      3      BEGIN
: 498      0893      3
: 499      0894      3      BIND
: 500      0895      3      count = anlseg [anlseg$l_count] : VECTOR [, LONG];
: 501      0896      3
: 502      0897      3      anlseg [anlseg$l_chars] = .anlseg [anlseg$l_chars] + 1;
: 503      0898      3      count [dcx$c_eor] = .count [dcx$c_eor] + 1;
: 504      0899      3      END;
: 505      0900      3      RETURN dcx$_normal;
: 506      0901      3
: 507      0902      1      END;

! Of dcx$analyze_data
```

			000C 00000	.ENTRY DCX\$ANALYZE_DATA, Save R2,R3	: 0831
		51	D4 00002	CLRL R1	: 0866
50	04	BC	D0 00004	MOVL @CONTEXT_ADDR, R0	
		0000G	30 00008	BSBW DCX\$CTX_CHECK	
44		50	E9 0000B	BLBC STATUS, 3\$	
50	08	AC	D0 0000E	MOVL REC, R0	: 0870
		00000000G	00 16 00012	JSB LIB\$ANALYZE_SDESC_R2	
37		50	E9 00018	BLBC STATUS, 3\$	
50	04	BC	14 C1 0001B	ADDL3 #20, @CONTEXT_ADDR, ANL	: 0874
		51	C0 00020	ADDL2 LEN, 4(ANL)	: 0875
	08	A0	D6 00024	INCL 8(ANL)	: 0876
53	18	A0	D0 00027	MOVL 24(ANL), ANLSEG	: 0877
		10	11 0002B	BRB 2\$	: 0878
	2C	A3	D6 0002D	INCL 44(ANLSEG)	: 0885
50		82	9A 00030	MOVZBL (ADDR)+, R0	: 0886
	38	A340	D6 00033	INCL 56(ANLSEG)[R0]	
53	043C	C340	D0 00037	MOVL 1084(ANLSEG)[R0], ANLSEG	: 0887
ED		51	F4 0003D	SOBGEQ I, 1\$	: 0878
50	38	A3	9E 00040	MOVAB 56(ANLSEG), R0	: 0895
	2C	A3	D6 00044	INCL 44(ANLSEG)	: 0897
	0400	C0	D6 00047	INCL 1024(R0)	: 0898
50	00000000G	8F	D0 0004B	MOVL #DCX\$_NORMAL, R0	: 0900
		04	00052	RET	: 0902

; Routine Size: 83 bytes, Routine Base: \$CODE\$ + 02AB

```
huffman_size - Compute size of Huffman Encoded Data'
1 %SBTTL 'huffman_size - Compute size of Huffman Encoded Data'
1
1 ROUTINE huffman_size (anl : REF BBLOCK, anlseg : REF BBLOCK) =
1 BEGIN
1 ++
1
1 Compute size of Huffman Encoded Data in bits
1 Store size in anlseg$l_comp_bits field
1 Also fill in anlseg$w_active and anlseg$w_active_r fields
1 Also fill in anlseg$b_min_char and anlseg$b_max_char fields
1
1 Inputs:
1
1 anl Address of anl structure
1 anlseg Address of anlseg structure
1
1 Outputs:
1
1 NONE
1
1 Return value:
1
1 Status code
1
1 --
1
1 BIND
1 count = anlseg [anlseg$l_count] : VECTOR [, LONG];
1
1 IF NOT .anl [anl$v_bounded]
1 THEN
1 BEGIN
1 anlseg [anlseg$w_active] = dcx$c_chars;
1 anlseg [anlseg$w_active_r] = dcx$c_chars;
1 anlseg [anlseg$b_min_char] = %X'00';
1 anlseg [anlseg$b_max_char] = %X'FF';
1 END
1 ELSE IF .anlseg [anlseg$l_chars] EQL 0
1 THEN
1 BEGIN
1 anlseg [anlseg$b_min_char] = %X'00';
1 anlseg [anlseg$b_max_char] = %X'00';
1 anlseg [anlseg$w_active] = 0;
1 anlseg [anlseg$w_active_r] = 0;
1 END
1 ELSE
1 BEGIN
1 anlseg [anlseg$b_min_char] = %X'00';
1 anlseg [anlseg$b_max_char] = %X'00';
1 anlseg [anlseg$w_active] = 0;
1 DECR 1 FROM anlseg$c_count-2 TO 0 DO
1 IF .count [.i] NEQ 0
1 THEN
1 BEGIN
1 anlseg [anlseg$w_active] = .anlseg [anlseg$w_active] + 1;
1 anlseg [anlseg$b_min_char] = .i;
1 END;
1
```



```
566 0960 3 IF .count [dcx$e_or] NEQ 0
567 0961 3 THEN
568 0962 3     anlseg [anlseg$w_active] = .anlseg [anlseg$w_active] + 1;
569 0963 3 IF .anlseg [anlseg$w_active] NEQ 1
570 0964 3 THEN
571 0965 3     anlseg [anlseg$w_active_r] = .anlseg [anlseg$w_active]
572 0966 3 ELSE
573 0967 3     anlseg [anlseg$w_active_r] = 2;
574 0968 3     DECR i FROM anlseg$e_count-2 TO 0 DO
575 0969 3         IF .count [.i] NEQ 0
576 0970 3             THEN
577 0971 4                 BEGIN
578 0972 4                     anlseg [anlseg$b_max_char] = .i;
579 0973 4                     EXITLOOP;
580 0974 3                 END;
581 0975 3     END;
582 0976 3 IF .anlseg [anlseg$l_chars] EQL 0
583 0977 3 THEN
584 0978 3     anlseg [anlseg$l_comp_bits] = 0
585 0979 3 ELSE IF .anlseg [anlseg$w_active] EQL 0
586 0980 3 THEN
587 0981 3     anlseg [anlseg$l_comp_bits] = 0
588 0982 3 ELSE IF .anlseg [anlseg$w_active] EQL 1
589 0983 3 THEN
590 0984 3     anlseg [anlseg$l_comp_bits] = .anlseg [anlseg$l_chars]
591 0985 3 ELSE
592 0986 3     BEGIN
593 0987 3         LOCAL
594 0988 3         p1 : REF VECTOR [4, LONG],
595 0989 3         p2 : REF VECTOR [4, LONG],
596 0990 3         ptr : LONG,
597 0991 3         list : VECTOR [4 * anlseg$e_count, LONG],
598 0992 3         zero : LONG;
599 0993 3         ! pointer into list
600 0994 3         ! storage list
601 0995 3     anlseg [anlseg$l_comp_bits] = 0;
602 0996 3     zero = false;
603 0997 3     ptr = 0;
604 0998 3     p1 = list [0];
605 0999 3     DECR i FROM anlseg$e_count-1 TO 0 DO
606 1000 4         BEGIN
607 1001 4             p1 [3] = .count [.i];
608 1002 4             IF .p1 [3] NEQ 0
609 1003 4                 THEN
610 1004 5                 BEGIN
611 1005 5                     tree_insert (ptr, p1 [0]);
612 1006 5                     p1 = p1 [4];
613 1007 5                     ! bump to next cell
614 1008 5                 END
615 1009 4             ELSE IF (NOT .anl [anl$v_bounded]) AND (NOT .zero)
616 1010 4                 THEN
617 1011 5                 BEGIN
618 1012 5                     ! all zeroes logically consolidate to one zero
619 1013 5                     zero = true;
620 1014 5                     tree_insert (ptr, p1 [0]);
621 1015 5                     p1 = p1 [4];
622 1016 5                     ! bump to next cell
623 1016 3                 END;
624 1016 3     END;
```

huffman_size - Compute size of Huffman Encoded

14-Sep-1984 12:15:55

DISK\$VMSMASTER:[DCX.SRC]ANALYZE.B32;1 (7)

```
1017 3  WHILE (p1 = tree_least (ptr); (p2 = tree_least (ptr)) NEQA 0) DO
1018 4      BEGIN
1019 4      p1 [3] = .p1 [3] + .p2 [3];
1020 4      anlseg [anlseg$l_comp_bits] = .anlseg [anlseg$l_comp_bits] + .p1 [3];
1021 4      tree_insert (ptr, p1 [0]);
1022 3      END;
1023 2  END;
1024 2  anlseg [anlseg$l_adj_bits] = .anlseg [anlseg$l_comp_bits];
1025 2  RETURN dcx$normal;
1026 2
1027 1  END;

! Of huffman_size
```

00FC 00000 HUFFMAN_SIZE:				.WORD		Save R2,R3,R4,R5,R6,R7	
5E	EFEC	CE	9E	00C02	MOVAB	-4116(SP), SP	0905
53	08	AC	D0	00007	MOVL	ANLSEG, R3	0930
55	38	A3	9E	0000B	MOVAB	56(R3), R5	
51	10	A3	9E	0000F	MOVAB	16(R3), R1	0935
13	04	BC	E8	00013	BLBS	@ANL, 1\$	0932
61	0101	8F	B0	00017	MOVW	#257, (R1)	0935
12	A3	0101	8F	0001C	MOVW	#257, 18(R3)	0936
15	A3	FF00	8F	00022	MOVW	#65280, 21(R3)	0937
			4F	11	BRB	10\$	0932
	2C	A3	D5	0002A	TSTL	44(R3)	0940
		0A	12	0002D	BNEQ	2\$	
	15	A3	B4	0002F	CLRW	21(R3)	0943
		61	B4	00032	CLRW	(R1)	0945
	12	A3	B4	00034	CLRW	18(R3)	0946
		40	11	00037	BRB	10\$	0940
	15	A3	B4	00039	CLRW	21(R3)	0950
		61	B4	0003C	CLRW	(R1)	0952
50	FF	8F	9A	0003E	MOVZBL	#255, 1	0953
		6540	D5	00042	TSTL	(R5)[1]	0954
		06	13	00045	BEQL	4\$	
		61	B6	00047	INCW	(R1)	0957
15	A3	50	90	00049	MOVB	1, 21(R3)	0958
	F2	50	F4	0004D	SOBGEQ	1, 3\$	0954
		0400	C5	00050	TSTL	1024(R5)	0960
			02	13	BEQL	5\$	
		61	B6	00056	INCW	(R1)	0962
01		61	B1	00058	CMPW	(R1), #1	0963
		06	13	0005B	BEQL	6\$	
12	A3	61	B0	0005D	MOVW	(R1), 18(R3)	0965
		04	11	00061	BRB	7\$	
12	A3	02	B0	00063	MOVW	#2, 18(R3)	0967
	50	FF	8F	00067	MOVZBL	#255, 1	0968
		6540	D5	0006B	TSTL	(R5)[1]	0969
		06	13	0006E	BEQL	9\$	
16	A3	50	90	00070	MOVB	1, 22(R3)	0972
		03	11	00074	BRB	10\$	0971
	F2	50	F4	00076	SOBGEQ	1, 8\$	0969
	56	24	A3	9E	MOVAB	36(R3), R6	0978
		2C	A3	D5	TSTL	44(R3)	0976

		04	13	00080	BEQL	11\$	
		61	B5	00082	TSTW	(R1)	0979
		04	12	00084	BNEQ	12\$	
		66	D4	00086	CLRL	(R6)	0981
		09	11	00088	BRB	13\$	
01		61	B1	0008A	CMPW	(R1), #1	0982
		07	12	0008D	BNEQ	14\$	
66	2C	A3	D0	0008F	MOVL	44(R3), (R6)	0984
		010D	31	00093	BRW	39\$	
		66	D4	00096	CLRL	(R6)	0995
		57	D4	00098	CLRL	ZERO	0996
		6E	D4	0009A	CLRL	PTR	0997
50	04	AE	9E	0009C	MOVAB	LIST, P1	0998
52	0100	8F	3C	000A0	MOVZWL	#256, I	0999
OC	A0	6542	D0	000A5	MOVL	(R5)[I], 12(P1)	1001
		21	13	000AA	BEQL	18\$	1002
		60	7C	000AC	CLRQ	(P1)	1005
	08	A0	D4	000AE	CLRL	8(P1)	
54		6E	9E	000B1	MOVAB	PTR, -H	
51		64	D0	000B4	MOVL	(H), -R1	
		48	13	000B7	BEQL	22\$	
OC	A1	OC	A0	D1	000B9	CPL	12(P1), 12(R1)
			05	1E	000BE	BGEQU	17\$
54			51	D0	000C0	MOVL	R1, -H
			EF	11	000C3	BRB	16\$
			31	1B	000C5	BLEQU	21\$
54	04	A1	9E	000C7	MOVAB	4(R1), -H	
		E7	11	000CB	BRB	16\$	
36	04	BC	E8	000CD	BLBS	23\$	1008
33		57	E8	000D1	BLBS	ZERO, 23\$	
57		01	D0	000D4	MOVL	#1, ZERO	1012
		60	7C	000D7	CLRQ	(P1)	1013
	08	A0	D4	000D9	CLRL	8(P1)	
54		6E	9E	000DC	MOVAB	PTR, -H	
51		64	D0	000DF	MOVL	(H), -R1	
		1D	13	000E2	BEQL	22\$	
OC	A1	OC	A0	D1	000E4	CPL	12(P1), 12(R1)
			05	1E	000E9	BGEQU	20\$
54			51	D0	000EB	MOVL	R1, -H
			EF	11	000EE	BRB	19\$
			06	1B	000F0	BLEQU	21\$
54	04	A1	9E	000F2	MOVAB	4(R1), -H	
		E7	11	000F6	BRB	19\$	
08	A0	08	A1	D0	000F8	MOVL	8(R1), 8(P1)
	54	08	A1	9E	000FD	MOVAB	8(R1), -H
	64		80	7E	00101	MOVAQ	(P1)+, -H
	50		08	C0	00104	ADDL2	#8, P1
	9B		52	F4	00107	SOBGEQ	I, 15\$
	55		6E	9E	0010A	MOVAB	PTR, -H
			65	D5	0010D	TSTL	(H), -H
			04	12	0010F	BNEQ	25\$
			50	D4	00111	CLRL	P1
			24	11	00113	BRB	29\$
		00	B5	D5	00115	TSTL	20(-H)
			05	13	00118	BEQL	26\$
55			65	D0	0011A	MOVL	(H), -H
			F6	11	0011D	BRB	25\$

54		65	D0	0011F	26\$:	MOVL	(H), R4		
51	08	A4	D0	00122		MOVL	8(R4), -Q		
		07	13	00126		BEQL	27\$		
08	A4	08	A1	D0	00128	MOVL	8(Q), 8(R4)		
		07	11	0012D		BRB	28\$		
51		54	D0	0012F	27\$:	MOVL	R4, Q		
65	04	A1	D0	00132		MOVL	4(Q), (-H)		
50		51	D0	00136	28\$:	MOVL	R1, P1		
55		6E	9E	00139	29\$:	MOVAB	PTR, -H		
		65	D5	0013C		TSTL	(H)		
		04	12	0013E		BNEQ	30\$		
		52	D4	00140		CLRL	P2		
		24	11	00142		BRB	34\$		
	00	B5	D5	00144	30\$:	TSTL	20(-H)		
		05	13	00147		BEQL	31\$		
55		65	D0	00149		MOVL	(H), -H		
		F6	11	0014C		BRB	30\$		
54		65	D0	0014E	31\$:	MOVL	(H), R4		
51	08	A4	D0	00151		MOVL	8(R4), -Q		
		07	13	00155		BEQL	32\$		
08	A4	08	A1	D0	00157	MOVL	8(Q), 8(R4)		
		07	11	0015C		BRB	33\$		
51		54	D0	0015E	32\$:	MOVL	R4, Q		
65	04	A1	D0	00161		MOVL	4(Q), (-H)		
52		51	D0	00165	33\$:	MOVL	R1, P2		
		39	13	00168	34\$:	BEQL	39\$		
0C	A0	0C	A2	C0	0016A	ADDL2	12(P2), 12(P1)	1019	
	66	0C	A0	C0	0016F	ADDL2	12(P1), (R6)	1020	
		08	60	7C	00173	CLRQ	(P1)	1021	
			A0	D4	00175	CLRL	8(P1)		
54		6E	9E	00178		MOVAB	PTR, -H		
51		64	D0	0017B	35\$:	MOVL	(H), -R1		
		1D	13	0017E		BEQL	38\$		
0C	A1	0C	A0	D1	00180	CMPL	12(P1), 12(R1)		
		05	1E	00185		BGEQU	36\$		
54		51	D0	00187		MOVL	R1, -H		
		EF	11	0018A		BRB	35\$		
		06	1B	0018C	36\$:	BLEQU	37\$		
54	04	A1	9E	0018E		MOVAB	4(R1), -H		
		E7	11	00192		BRB	35\$		
08	A0	08	A1	D0	00194	37\$:	MOVL	8(R1), 8(P1)	
	54	08	A1	9E	00199	MOVAB	8(R1), -H		
	64		50	D0	0019D	38\$:	MOVL	P1, (-H)	
		FF67	31	001A0		BRW	24\$	1017	
28	A3		66	D0	001A3	39\$:	MOVL	(R6), 40(R3)	1024
	50	00000000G	8F	D0	001A7	MOVL	#DCX\$_NORMAL, R0	1025	
			04	001AE		RET		1027	

; Routine Size: 431 bytes, Routine Base: \$CODE\$ + 02FE

remove_seg - Remove one segment

```

: 635      1028 1 %SBTTL 'remove_seg - Remove one segment'
: 636      1029 1
: 637      1030 1 ROUTINE remove_seg (anl : REF BBLOCK, anlseg : REF BBLOCK) =
: 638      1031 2 BEGIN
: 639      1032 2 ++
: 640      1033 2
: 641      1034 2 Remove one segment fixing pointers and freeing storage
: 642      1035 2
: 643      1036 2 Inputs:
: 644      1037 2
: 645      1038 2         anl                Address of anl structure
: 646      1039 2         anlseg            Address of anlseg structure to be removed
: 647      1040 2
: 648      1041 2 Outputs:
: 649      1042 2
: 650      1043 2         NONE
: 651      1044 2
: 652      1045 2 Return value:
: 653      1046 2
: 654      1047 2         dcx$normal
: 655      1048 2         memory deallocation error
: 656      1049 2
: 657      1050 2 --
: 658      1051 2
: 659      1052 2 ! this segment is unprofitable, eliminate it
: 660      1053 2
: 661      1054 2 LOCAL
: 662      1055 2     seg : REF BBLOCK, ! replacement
: 663      1056 2     anl_p : REF BBLOCK;
: 664      1057 2
: 665      1058 2 remque (anlseg [anlseg$q_queue], anl_p);
: 666      1059 2 IF .anlseg [anlseg$l_prev] NEQA 0
: 667      1060 2 THEN
: 668      1061 2 BEGIN
: 669      1062 2
: 670      1063 2 BIND
: 671      1064 2     p_seg = anlseg [anlseg$l_prev] : REF BBLOCK;
: 672      1065 2
: 673      1066 2     p_seg [anlseg$w_sons] = .p_seg [anlseg$w_sons] - 1;
: 674      1067 2     seg = .anl [anl$l_flink];
: 675      1068 2     INCR index FROM 1 TO .anlseg [anlseg$b_depth] - 2 DO
: 676      1069 2     BEGIN
: 677      1070 2     4
: 678      1071 2     4 BIND
: 679      1072 2     4     seg_next = seg [anlseg$l_next] : VECTOR [, LONG],
: 680      1073 2     4     anlseg_string = anlseg [anlseg$t_string] : VECTOR [, BYTE];
: 681      1074 2     4
: 682      1075 2     4     seg = .seg_next [.anlseg_string [.index]];
: 683      1076 2     3 END;
: 684      1077 2     3 anl_p = .anl [anl$l_flink];
: 685      1078 2     3 WHILE .anl_p NEQA anl [anl$q_queue] DO
: 686      1079 2     4 BEGIN
: 687      1080 2     4
: 688      1081 2     4 BIND
: 689      1082 2     4     next_p = anl_p [anlseg$l_next] : VECTOR [, LONG];
: 690      1083 2     4
: 691      1084 2     4 IF .next_p [.anlseg [anlseg$w_char]] EQLA .anlseg
```

remove_seg - Remove one segment

```
: 692      1085 4      THEN
: 693      1086 4      next_p [.anlseg [anlseg$w_char]] = .seg;
: 694      1087 4      anl_p = .anl_p [anlseg$l_link];
: 695      1088 3      END;
: 696      1089 2      END;
: 697      1090 2      perform (dcx$free_vm (.anlseg [anlseg$l_size], .anlseg));
: 698      1091 2      anl [anl$w_nsegs] = .anl [anl$w_nsegs] = 1;
: 699      1092 2      RETURN dcx$_normal;
: 700      1093 2
: 701      1094 1 END;                                ! Of remove_seg
```

00FC 00000 REMOVE_SEG:						
	53	08	BC 0F 00002	.WORD	Save R2,R3,R4,R5,R6,R7	1030
	54	08	AC D0 00006	REMQUE	@ANLSEG, ANL_P	1058
		20	A4 D5 0000A	MOVL	ANLSEG, R4	1059
			53 13 0000D	TSTL	32(R4)	
	50	20	A4 D0 0000F	BEQL	5\$	
		1C	A0 B7 00013	MOVL	32(R4), R0	1066
	52	04	AC D0 00016	DECW	28(R0)	
	56	18	A2 D0 0001A	MOVL	ANL, R2	1067
	57	14	A4 9A 0001E	MOVL	24(R2), SEG	
	57		02 C2 00022	MOVZBL	20(R4), R7	1068
	51	30	A4 9E 00025	SUBL2	#2, R7	
			55 D4 00029	MOVAB	48(R4), R1	1073
			0A 11 0002B	CLRL	INDEX	
	50		6541 9A 0002D 1\$:	BRB	2\$	
	56	043C	C640 D0 00031	MOVZBL	(INDEX)[R1], R0	1075
F2	55		57 F3 00037 2\$:	MOVL	1084(SEG)[R0], SEG	
	53	18	A2 D0 0003B	AOBLEQ	R7, INDEX, 1\$	1068
	50	0E	A4 9E 0003F	MOVL	24(R2), ANL_P	1077
	51	18	A2 9E 00043 3\$:	MOVAB	14(R4), R0	1084
	51		53 D1 00047	MOVAB	24(R2), R1	1078
			16 13 0004A	CMPL	ANL_P, R1	
	51		60 3C 0004C	BEQL	5\$	
	54	043C	C341 D1 0004F	MOVZWL	(R0), R1	1084
			06 12 00055	CMPL	1084(ANL_P)[R1], R4	
043C	C341		56 D0 00057	BNEQ	4\$	
	53		63 D0 0005D 4\$:	MOVL	SEG, 1084(ANL_P)[R1]	1086
			E1 11 00060	MOVL	(ANL_P), ANL_P	1087
			54 DD 00062 5\$:	BRB	3\$	1078
		08	A4 DD 00064	PUSHL	R4	1090
0000G	CF		02 FB 00067	PUSHL	8(R4)	
	0E		50 E9 0006C	CALLS	#2, DCX\$FREE_VM	
	50	04	AC D0 0006F	BLBC	STATUS, 6\$	
		16	A0 B7 00073	MOVL	ANL, R0	1091
	50	00000000G	8F D0 00076	DECW	22(R0)	
			04 0007D 6\$:	MOVL	#DCX\$_NORMAL, R0	1092
				RET		1094

; Routine Size: 126 bytes, Routine Base: \$CODE\$ + 04AD

Eliminate_seg - Remove unprofitable segments

```
: 703      1095 1 %SBTTL 'Eliminate_seg - Remove unprofitable segments'
: 704      1096 1
: 705      1097 1 ROUTINE eliminate_seg (anl : REF BBLOCK, this : REF BBLOCK) =
: 706      1098 2 BEGIN
: 707      1099 2 ++
: 708      1100 2
: 709      1101 2 Determine if a segment is unprofitable and ifso, remove it
: 710      1102 2
: 711      1103 2 Inputs:
: 712      1104 2
: 713      1105 2 anl Address of anl structure
: 714      1106 2 this Address of anlseg structure in question
: 715      1107 2
: 716      1108 2 Outputs:
: 717      1109 2
: 718      1110 2 NONE
: 719      1111 2
: 720      1112 2 Return value:
: 721      1113 2
: 722      1114 2 dcx$_normal
: 723      1115 2 memory deallocation error
: 724      1116 2
: 725      1117 2 --
: 726      1118 2
: 727      1119 2 Macro to scale a size based on the ratio of observed to estimated
: 728      1120 2 data.
: 729      1121 2
: 730      M 1122 2 MACRO scale_map (size) =
: 731      M 1123 2 BEGIN
: 732      M 1124 2
: 733      M 1125 2 LOCAL
: 734      M 1126 2 scaled_size : LONG, ! result
: 735      M 1127 2 remainder : LONG, ! throw away
: 736      M 1128 2 ext_prod : VECTOR [2, LONG]; ! extended intermediate value
: 737      M 1129 2
: 738      M 1130 2 BUILTIN
: 739      M 1131 2 emul,
: 740      M 1132 2 ediv;
: 741      M 1133 2
: 742      M 1134 2 emul (%REF (size), anl [anl$l_ratio_num], %REF (0), ext_prod);
: 743      M 1135 2 ediv (anl [anl$l_ratio_denom], ext_prod, scaled_size, remainder);
: 744      M 1136 2 .scaled_size
: 745      M 1137 2
: 746      1138 2 END%;
: 747      1139 2
: 748      1140 2 BIND
: 749      1141 2 aff = this [anlseg$l_prev] : REF BBLOCK,
: 750      1142 2 this_count = this [anlseg$l_count] : VECTOR [, LONG],
: 751      1143 2 next = this [anlseg$l_next] : VECTOR [, LONG];
: 752      1144 2
: 753      1145 2 ! If we have no true successors then we can
: 754      1146 2 ! consider eliminating the current anlseg block
: 755      1147 2
: 756      1148 2 IF .this [anlseg$w_sons] NEQ 0
: 757      1149 2 THEN
: 758      1150 2 0
: 759      1151 2 ELSE IF .this [anlseg$l_chars] EQL 0
```

```

: 760      1152 2 THEN
: 761      1153 2   remove_seg (.anl, .this)      ! Basic map is bigger than data
: 762      1154 2 ELSE IF .this [anlseg$w_active] EQL 0
: 763      1155 2 THEN
: 764      1156 2   remove_seg (.anl, .this)
: 765      1157 2 ELSE IF .this [anlseg$l_prev] EQL 0
: 766      1158 2 THEN
: 767      1159 2   BEGIN
: 768      1160 2       ! check profitability of eliminating the last block
: 769      1161 2       !
: 770      1162 2       IF (
: 771      1163 2           (
: 772      1164 2               (.this [anlseg$l_comp_bits])
: 773      1165 2               - ((.this [anlseg$l_chars] - .this_count [dcx$c_eor]) * 8)
: 774      1166 2               )
: 775      1167 2           + scale_map (
: 776      1168 2               (dcxsbm$k_length*8
: 777      1169 2               + (.this [anlseg$w_active_r] - 1) * (2 + 16))
: 778      1170 2               - (0)
: 779      1171 2           )
: 780      1172 2           ) GEQ 0
: 781      1173 2           ! true if current total size is greater
: 782      1174 2       THEN
: 783      1175 2           remove_seg (.anl, .this);
: 784      1176 2       END
: 785      1177 2 ELSE
: 786      1178 2   BEGIN
: 787      1179 2       ! check profitability of eliminating the block
: 788      1180 2       !
: 789      1181 2       LOCAL
: 790      1182 2       aff2 : BBLOCK [anlseg$k_length];
: 791      1183 2       aff2 : BBLOCK [anlseg$k_length];
: 792      1184 2       BIND
: 793      1185 2       aff2_count = aff2 [anlseg$l_count] : VECTOR [, LONG];
: 794      1186 2       CHSMOVE (.aff [anlseg$l_size], .aff, aff2);
: 795      1187 2       aff2 [anlseg$l_chars] = .aff2 [anlseg$l_chars] + .this [anlseg$l_chars];
: 796      1188 2       DECR i FROM anlseg$c_count - 1 TO 0 DO
: 797      1189 2       aff2_count [.i] = .aff2_count [.i] + .this_count [.i];
: 798      1190 2       perform thuffman_size (.anl, aff2);
: 799      1191 2       IF (
: 800      1192 2           (
: 801      1193 2               (.this [anlseg$l_adj_bits])
: 802      1194 2               + (.aff [anlseg$l_adj_bits])
: 803      1195 2               - (.aff2 [anlseg$l_adj_bits])
: 804      1196 2               )
: 805      1197 2           + scale_map (
: 806      1198 2               (dcxsbm$k_length*8 + (.this [anlseg$w_active_r] - 1) * (2 + 16) +
: 807      1199 2               (.this [anlseg$b_max_char] - .this [anlseg$b_min_char] + 1) * 16)
: 808      1200 2               + (dcxsbm$k_length*8 + (.aff [anlseg$w_active_r] - 1) * (2 + 16) +
: 809      1201 2               (.aff [anlseg$b_max_char] - .aff [anlseg$b_min_char] + 1) * 16)
: 810      1202 2               - (dcxsbm$k_length*8 + (.aff2 [anlseg$w_active_r] - 1) * (2 + 16) +
: 811      1203 2               (.aff2 [anlseg$b_max_char] - .aff2 [anlseg$b_min_char] + 1) * 16)
: 812      1204 2               (IF .anl [anlseg$w_nsegs] GTR 2
: 813      1205 2                   THEN ((.aff2 [anlseg$b_max_char] - .aff2 [anlseg$b_min_char] + 1) * 16)
: 814      1206 2                   ELSE 0))
: 815      1207 2           )
: 816      1208 2       )

```



```
! Of eliminate_seg
```

	CPU	PC	OPCODE	EFFECTIVE ADDRESS	INSTRUMENTED INSTRUCTIONS	ADDRESS OF FIRST INSTRUCTION	ADDRESS OF LAST INSTRUCTION
SE	F7B8	CE	9E 00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10	: 1097	:
56	08	AC	D0 00007	MOVAB	-2120(SP), SP	: 1141	:
SA	38	A6	9E 0000B	MOVL	THIS, R6	: 1142	:
	1C	A6	B5 0000F	MOVAB	56(R6), R10	: 1148	:
		O3	13 00012	TSTW	28(R6)	:	:
		00F3	31 00014	BEQL	2\$	:	:
58	04	AC	D0 00017	BRW	10\$	: 1153	:
59	2C	A6	D0 0001B	MOVL	ANL, R8	: 1151	:
		33	13 0001F	MOVL	44(R6), R9	:	:
	10	A6	B5 00021	BEQL	3\$	: 1154	:
		2E	13 00024	TSTW	16(R6)	:	:
	20	A6	D5 00026	BEQL	3\$	: 1157	:
		2C	12 00029	TSTL	32(R6)	:	:
CA		S9	C3 0002B	BNEQ	4\$	: 1166	:
51	24 B640	7E	00031	SUBL3	R9, 1024(R10), RO	: 1164	:
50	12	A6	3C 00036	MovAQ	@36(R6)[R0], R1	: 1172	:
50		12	C4 0003A	MOVZWL	18(R6), RO	:	:
50	4E	A0	9E 0003D	MULL2	#18, RO	:	:
A8		50	7A 00041	MOVAB	78(RO), RO	:	:
AD	24	A8	7B 00048	EMUL	RO, 32(R8), #O, EXT_PROD	:	:
51		50	C0 0004F	EDIV	36(R8), EXT_PROD, SCALED_SIZE, REMAINDER	:	:
		C0	19 00052	ADDL2	SCALED_SIZE, R1	: 1173	:
		00AA	31 00054	BLSS	1\$	: 1175	:
57	20	A6	D0 00057	BRW	9\$	: 1188	:
67	08	A7	28 0005B	MOVL	32(R6), R7	:	:
AE		59	C0 00061	MOVCL	8(R7), (R7), AFF2	: 1189	:
50	0100	8F	3C 00065	ADDL2	R9, AFF2+44	: 1190	:
E40		6A40	C0 0006A	MOVZWL	#256, I	: 1191	:
F7		50	F4 00070	ADDL2	(R10)[I], AFF2_COUNT[I]	: 1192	:
	08	AE	9F 00073	SOBGEO	I, 5\$	:	:
		58	DD 00076	PUSHAB	AFF2	:	:
CF		02	FB 00078	PUSHL	R8	:	:
01		50	E8 0007D	CALLS	#2, HUFFMAN_SIZE	:	:
		04	00080	BLBS	STATUS, 6\$	: 1196	:
A6	28	A7	C1 00081	RET		: 1197	:
53	30	AE	C2 00087	ADDL3	40(R7), 40(R6), R3	: 1208	:
50	12	A6	3C 0008B	SUBL2	AFF2+40, R3	:	:
50		12	C4 0008F	MOVZWL	18(R6), RO	:	:
51	16	A6	9A 00092	MULL2	#18, RO	:	:
52	15	A6	9A 00096	MOVZBL	22(R6), R1	:	:
				MOVZBL	21(R6), R2	:	:

51				50	000A	SUBL2	R2, R1	
51				50	000A	MULL2	#16, R1	
51				50	000A	ADDL2	R0, R1	
50	12	A7	50	000A3	MOVZWL	18(R7), R0		
50		12	C4	000A7	MULL2	#18, R0		
52	16	A7	9A	000AA	MOVZBL	22(R7), R2		
54	15	A7	9A	000AE	MOVZBL	21(R7), R4		
52		54	C2	000B2	SUBL2	R4, R2		
52		10	C4	000B5	MULL2	#16, R2		
50		52	C0	000B8	ADDL2	R2, R0		
50		51	C0	000BB	ADDL2	R1, R0		
52	1A	AE	3C	000BE	MOVZWL	AFF2+18, R2		
52		12	C4	000C2	MULL2	#18, R2		
02	16	A8	B1	000C5	CMPW	22(R8), #2		
		13	1B	000C9	BLEQU	7\$		
51	1E	AE	9A	000CB	MOVZBL	AFF2+22, R1		
54	1D	AE	9A	000CF	MOVZBL	AFF2+21, R4		
51		54	C2	000D3	SUBL2	R4, R1		
51		10	C4	000D6	MULL2	#16, R1		
51		10	C0	000D9	ADDL2	#16, R1		
		02	11	000DC	BRB	8\$		
		51	D4	000DE	CLRL	R1		
51		52	C0	000E0	ADDL2	R2, R1		
50		51	C2	000E3	SUBL2	R1, R0		
50	6E	A0	9E	000E6	MOVAB	110(R0), R0		
51		50	7A	000EA	EMUL	R0, 32(R8), #0, EXT_PROD		
6E		50	7B	000F0	EDIV	36(R8), EXT_PROD, SCALED_SIZE, REMAINDER		
51		50	C0	000F6	ADDL2	SCALED_SIZE, R3		
		0F	19	000F9	BLSS	10\$		
	67	08	AE	08	A7	28	000FB	
			56	DD	00101	9\$:		
			58	DD	00103	PUSHL	R6	
			02	FB	00105	PUSHL	R8	
			02	FB	00105	CALLS	#2, REMOVE_SEG	
			8F	DD	0010A	10\$:		
			04	00111	MOVL	#DCX\$_NORMAL, R0		
					RET			

1209
1212
1213
1216
1218

; Routine Size: 274 bytes, Routine Base: \$CODE\$ + 052B

build_map_seg - Build a map segment

```

: 828 1219 1 %SBTTL 'build_map_seg - Build a map segment'
: 829 1220 1
: 830 1221 1 ROUTINE build_map_seg (anl : REF BBLOCK, anlseg : REF BBLOCK, dcxsbm : REF BBLOCK) =
: 831 1222 2 BEGIN
: 832 1223 2 ++
: 833 1224 2
: 834 1225 2 Build a map segment
: 835 1226 2
: 836 1227 2 Inputs:
: 837 1228 2
: 838 1229 2 anl Address of anl block
: 839 1230 2 anlseg Analysis segment structure address
: 840 1231 2 dcxsbm Analysis of allocated map segment
: 841 1232 2
: 842 1233 2 Outputs:
: 843 1234 2
: 844 1235 2 dcxsbm Filled in
: 845 1236 2
: 846 1237 2 Return value:
: 847 1238 2
: 848 1239 2 dcx$_normal All is well
: 849 1240 2 --
: 850 1241 2
: 851 1242 2 LOCAL
: 852 1243 2 flags : REF BITVECTOR,
: 853 1244 2 nodes : REF VECTOR [, BYTE],
: 854 1245 2 next : REF VECTOR [, WORD],
: 855 1246 2 p1 : REF VECTOR [6, LONG],
: 856 1247 2 p2 : REF VECTOR [6, LONG],
: 857 1248 2 ptr : LONG,
: 858 1249 2 list : VECTOR [6 * anlseg$c_count, LONG]; ! pointer into list
: 859 1250 2 ! storage list
: 860 1251 2 BIND
: 861 1252 2 count = anlseg [anlseg$l_count] : VECTOR [, LONG];
: 862 1253 2
: 863 1254 2 flags = .dcxsbm + .dcxsbm [dcxsbm$w_flags];
: 864 1255 2 nodes = .dcxsbm + .dcxsbm [dcxsbm$w_nodes];
: 865 1256 2 IF .dcxsbm [dcxsbm$w_next] NEQ 0
: 866 1257 2 THEN
: 867 1258 2 next = .dcxsbm + .dcxsbm [dcxsbm$w_next]
: 868 1259 2 ELSE
: 869 1260 2 next = 0;
: 870 1261 2 ptr = 0;
: 871 1262 2 p1 = list [0];
: 872 1263 2 DECR i FROM anlseg$c_count-1 TO 0 DO
: 873 1264 2 BEGIN
: 874 1265 2 p1 [3] = .count [i];
: 875 1266 2 IF .p1 [3] NEQ 0 OR NOT .anl [anl$v_bounded]
: 876 1267 2 THEN
: 877 1268 2 BEGIN
: 878 1269 2 p1 [4] = 0; ! node offset
: 879 1270 2 p1 [5] = .i; ! character code
: 880 1271 2 tree_insert (ptr, p1 [0]);
: 881 1272 2 p1 = p1 [6];
: 882 1273 2 END;
: 883 1274 2 END;
: 884 1275 2 IF .anlseg [anlseg$w_active] EQL 1
```

```
: 885      1276 2 THEN
: 886      1277 BEGIN
: 887      1278   p2 = ptr;
: 888      1279   p1 [3] = .p2 [3];
: 889      1280   p1 [4] = .p2 [4];
: 890      1281   p1 [5] = .p2 [5];
: 891      1282   tree_insert (ptr, p1 [0]);
: 892      1283 END;
: 893      1284   DECR high FROM .anlseg [anlseg$w_active_r] - 2 TO 0 DO
: 894      1285   BEGIN
: 895      1286   LOCAL
: 896      1287   p : VECTOR [2, LONG];
: 897      1288   BIND
: 898      1289   p2 = p [0] : REF VECTOR [6, LONG],
: 899      1290   p1 = p [1] : REF VECTOR [6, LONG];
: 900      1291   DECR i FROM 1 TO 0 DO
: 901      1292   BEGIN
: 902      1293   LOCAL
: 903      1294   q : REF VECTOR [6, LONG];
: 904      1295   p [.i] = q = tree_least (ptr);
: 905      1296   IF .q [5] EQL dcx$c_eor
: 906      1297   THEN
: 907      1298   BEGIN
: 908      1299   flags [2*.high + .i] = false;
: 909      1300   nodes [2*.high + .i] = 0;
: 910      1301   END
: 911      1302   ELSE IF .q [5] GEQ 0
: 912      1303   THEN
: 913      1304   BEGIN
: 914      1305   flags [2*.high + .i] = true;
: 915      1306   nodes [2*.high + .i] = .q [5];
: 916      1307   IF .next NEQA 0
: 917      1308   THEN
: 918      1309   BEGIN
: 919      1310   BIND
: 920      1311   next_seg_a = anlseg [anlseg$l_next] : VECTOR [, LONG],
: 921      1312   next_seg = next_seg_a [.q [5]] : REF BBLOCK;
: 922      1313   next [.q [5] - .anlseg [anlseg$b_min_char]] = .next_seg [anlseg$w_id];
: 923      1314   END;
: 924      1315   END
: 925      1316   ELSE
: 926      1317   BEGIN
: 927      1318   flags [2*.high + .i] = false;
: 928      1319   nodes [2*.high + .i] = .q [4];
: 929      1320   END;
: 930      1321   END
: 931      1322   p1 [3] = .p1 [3] + .p2 [3];
: 932      1323   p1 [4] = .high;
: 933      1324   p1 [5] = -1;
: 934      1325   tree_insert (ptr, p1 [0]);
: 935      1326   END;
: 936      1327   END;
: 937      1328   END;
: 938      1329   END;
: 939      1330   END;
: 940      1331   END;
: 941      1332   END;
```



```
: 942      1333 2      END;
: 943      1334 2
: 944      1335 2 RETURN dcx$_normal;
: 945      1336 2
: 946      1337 1 END;

! of build_map_seg
```

```
03FC 00000 BUILD_MAP_SEG:
      SE      E7DC  CE  9E 00002      .WORD      Save R2,R3,R4,R5,R6,R7,R8,R9      : 1221
      55      08  AC  D0 00007      MOVAB      -6180(SP), SP
      50      0C  AC  D0 0000B      MOVL      ANLSEG, R5      : 1252
      59      06  A0  3C 0000F      MOVL      DCXSBM, R0      : 1254
      59      50  C0 00013      MOVZWL     6(R0), FLAGS
      56      08  A0  3C 00016      ADDL2     R0, FLAGS
      56      50  C0 0001A      MOVZWL     8(R0), NODES
      0A      A0  B5 0001D      ADDL2     R0, NODES
      09      13 00020      TSTW      10(R0)
      58      0A  A0  3C 00022      BEQL      1$
      58      50  C0 00026      MOVZWL     10(R0), NEXT
      02      11 00029      ADDL2     R0, NEXT
      58      D4 0002B 1$:      BRB      2$
      6E      D4 0002D 2$:      CLRL      NEXT
      0C      AE  9E 0002F      CLRL      PTR
      52      8F  3C 00033      MOVAB      LIST, P1
      OC  A0      38 A542 D0 00038 3$:      MOVZWL     #256, I
      04      12 0003E      MOVL      56(R5)[I], 12(P1)
      37      04  BC  E8 00040      BNEQ      4$
      14  A0      10  A0  D4 00044 4$:      BLBS      @ANL, 9$
      52      D0 00047      CLRL      16(P1)
      60      7C 0004B      MOVL      I, 20(P1)
      08      A0  D4 0004D      CLRQ      (P1)
      53      6E  9E 00050      CLRL      8(P1)
      51      63  D0 00053 5$:      MOVAB      PTR, _H
      1D      13 00056      MOVL      (H), -R1
      OC  A1      0C  A0  D1 00058      BEQL      8$
      05      1E 0005D      CMPL      12(P1), 12(R1)
      53      51  D0 0005F      BGEQU     6$
      EF      11 00062      MOVL      R1, _H
      06      1B 00064 6$:      BRB      5$
      53      04  A1  9E 00066      BLEQU     7$
      E7      11 0006A      MOVAB      4(R1), _H
      08  A0      08  A1  DC 0006C 7$:      BRB      5$
      53      08  A1  9E 00071      MOVL      8(R1), 8(P1)
      63      80  7E 00075 8$:      MOVAB      8(R1), _H
      50      10  C0 00078      MOVAQ     (P1)+, T_H
      BA      52  F4 0007B 9$:      ADDL2     #16, P1
      01      1D  A5  B1 0007E      SOBGEQ     I, 3$
      3A      12 00082      CMPW      16(R5), #1
      6E      D0 00084      BNEQ      14$
      OC  A0      0C  A1  7D 00087      MOVL      PTR, P2
      14  A0      14  A1  D0 0008C      MOVQ      12(P2), 12(P1)
      60      7C 00091      MOVL      20(P2), 20(P1)
      A0      D4 00093      CLRQ      (P1)
      CLRL      8(P1)      : 1272
      : 1263
      : 1275
      : 1278
      : 1279
      : 1281
      : 1282
```

	52	6E	9E	00096	MOVAB	PTR, -H			
	51	62	D0	00099	10\$:	MOVL	(H), -R1		
		1D	13	0009C		BEQL	13\$		
	0C	A1	0C	A0	D1	0009E	CMPL	12(P1), 12(R1)	
		05	1E	000A3		BGEQU	11\$		
	52	51	D0	000A5		MOVL	R1, -H		
		EF	11	000A8		BRB	10\$ -H		
		06	1B	000AA	11\$:	BLEQU	12\$		
	52	A1	9E	000AC		MOVAB	4(R1), -H		
		E7	11	000B0		BRB	10\$		
	08	A0	08	A1	D0	000B2	12\$:	MOVL	8(R1), 8(P1)
		52	08	A1	9E	000B7		MOVAB	8(R1), -H
		62		50	D0	000BB	13\$:	MOVL	P1, (H)
		51	12	A5	3C	000BE	14\$:	MOVZWL	18(R5), HIGH
				51	D7	000C2		DECL	HIGH
				00C5	31	000C4		BRW	31\$
57		51		01	78	000C7	15\$:	ASHL	#1, HIGH, R7
		54		01	D0	000CB		MOVL	#1, I
		53		6E	9E	000CE	16\$:	MOVAB	PTR, -H
				63	D5	000D1		TSTL	(H) -H
				04	12	000D3		BNEQ	17\$
				50	D4	000D5		CLRL	Q
				21	11	000D7		BRB	20\$
			00	B3	D5	000D9	17\$:	TSTL	20(-H)
				05	13	000DC		BEQL	18\$ -H
		53		63	D0	000DE		MOVL	(H), -H
				F6	11	000E1		BRB	17\$
		52		63	D0	000E3	18\$:	MOVL	(H), R2
		50		A2	D0	000E6		MOVL	8(R2), -Q
				07	13	000EA		BEQL	19\$
	08	A2	08	A0	D0	000EC		MOVL	8(Q), 8(R2)
				07	11	000F1		BRB	20\$
		50		52	D0	000F3	19\$:	MOVL	R2, Q
		63		A0	D0	000F6		MOVL	4(Q), (-H)
	04	AE44	04	50	D0	000FA	20\$:	MOVL	Q, -P[1]
53		57		54	C1	000FF		ADDL3	I, R7, R3
		52		A0	D0	00103		MOVL	20(Q), R2
	00000100	8F	14	52	D1	00107		CMPL	R2, #256
				09	12	0010E		BNEQ	22\$
00		69		53	E5	00110		BBCC	R3, (FLAGS), 21\$
			6346	94	00114	21\$:	CLRB	(R3)[NODES]	
				2E	11	00117		BRB	26\$
				52	D5	00119	22\$:	TSTL	R2
				21	19	0011B		BLSS	24\$
00		69		53	E2	0011D		BBSS	R3, (FLAGS), 23\$
		6346		52	90	00121	23\$:	MOVB	R2, (R3)[NODES]
				58	D5	00125		TSTL	NEXT
				1E	13	00127		BEQL	26\$
		50	15	A5	9A	00129		MOVZBL	21(R5), R0
50		52		50	C3	0012D		SUBL3	R0, R2, R0
		52	043C	C542	D0	00131		MOVL	1084(R5)[R2], R2
		6840	0C	A2	B0	00137		MOVW	12(R2), (NEXT)[R0]
				09	11	0013C		BRB	26\$
00		69		53	E5	0013E	24\$:	BBCC	R3, (FLAGS), 25\$
		6346	10	A0	90	00142	25\$:	MOVB	16(Q), (R3)[NODES]
		84		54	F4	00147	26\$:	SOBGEQ	I, 16\$
		50	08	AE	D0	0014A		MOVL	P1, R0

	52	04	AE	D0	0014E	MOVL	P2, R2		
0C	A0	0C	A2	C0	00152	ADDL2	12(R2), 12(R0)		
10	A0		51	D0	00157	MOVL	HIGH, 16(R0)		1330
14	A0		01	CE	0015B	MNEGL	#1, 20(R0)		1331
			60	7C	0015F	CLRQ	(R0)		1332
		08	A0	D4	00161	CLRL	8(R0)		
	53		6E	9E	00164	MOVAB	PTR, -H		
	52		63	D0	00167	MOVL	(H), -R2		
			1D	13	0016A	BEQL	30\$		
0C	A2	0C	A0	D1	0016C	CMPL	12(R0), 12(R2)		
			05	1E	00171	BGEQU	28\$		
	53		52	D0	00173	MOVL	R2, -H		
			EF	11	00176	BRB	27\$		
			06	1B	00178	BLEQU	29\$		
	53	04	A2	9E	0017A	MOVAB	4(R2), -H		
			E7	11	0017E	BRB	27\$		
08	A0	08	A2	D0	00180	MOVL	8(R2), 8(R0)		
	53	08	A2	9E	00185	MOVAB	8(R2), -H		
	63		50	D0	00189	MOVL	R0, (-H)		
	02		51	F4	0018C	SOBGEQ	HIGH, -32\$		1284
			03	11	0018F	BRB	33\$		
			FF33	31	00191	BRW	15\$		
	50	00000000G	8F	D0	00194	MOVL	#DCX\$_NORMAL, R0		1335
			04	0019B	RET				1337

; Routine Size: 412 bytes, Routine Base: \$CODE\$ + 063D

dcx\$make_map - Compute mapping function

```

: 948 1338 1 %SBTTL 'dcx$make_map - Compute mapping function'
: 949 1339 1
: 950 1340 1 GLOBAL ROUTINE dcx$make_map (context_addr, map_addr, map_size) =
: 951 1341 2 BEGIN
: 952 1342 2 ++
: 953 1343 2
: 954 1344 2 Compute mapping function
: 955 1345 2
: 956 1346 2 Inputs:
: 957 1347 2
: 958 1348 2 context_addr.mz.r Address of context longword
: 959 1349 2
: 960 1350 2 Outputs:
: 961 1351 2
: 962 1352 2 context_addr.mz.r Context block accumulates data
: 963 1353 2 map_addr.wa.r Address of longword to receive map address
: 964 1354 2 map_size.wl.r Address of longword to receive map length
: 965 1355 2
: 966 1356 2 Return value:
: 967 1357 2
: 968 1358 2 status.wlc.v
: 969 1359 2
: 970 1360 2 dcx$_normal All is well
: 971 1361 2 dcx$_invctx Invalid context block
: 972 1362 2 lib$_insvirmem
: 973 1363 2 --
: 974 1364 2
: 975 1365 2 BUILTIN
: 976 1366 2 NULLPARAMETER;
: 977 1367 2
: 978 1368 2 BIND
: 979 1369 2 ctx = .context_addr : REF BBLOCK,
: 980 1370 2 anl = ctx [ctx$l_specific] : BBLOCK,
: 981 1371 2 dcxmap = ctx [ctx$l_map] : REF BBLOCK;
: 982 1372 2
: 983 1373 2 LOCAL
: 984 1374 2 id,
: 985 1375 2 seg : REF BBLOCK,
: 986 1376 2 size;
: 987 1377 2
: 988 1378 2 .map_addr = 0;
: 989 1379 2 IF NOT NULLPARAMETER (3)
: 990 1380 2 THEN
: 991 1381 2 .map_size = 0;
: 992 1382 2
: 993 1383 2 ! validate context block
: 994 1384 2
: 995 1385 2 perform (dcx$ctx_check (.ctx, ctx$c_anlyz));
: 996 1386 2
: 997 1387 2 ! Compute ratio of estimated to observed data size.
: 998 1388 2 ! Give priority first to an estimate of data size,
: 999 1389 2 ! then to an estimate of number of records, and if
: 1000 1390 2 ! neither of these is provided, assume that estimated
: 1001 1391 2 ! is equal to observed.
: 1002 1392 2
: 1003 1393 2 IF .anl [anl$v_est_bytes] AND .anl [anl$l_est_d_bytes] GTR 0
: 1004 1394 2 THEN
```



```

: 1005      1395 3 BEGIN
: 1006      1396 3 anl [anl$l_ratio_num] = .anl [anl$l_d_bytes];
: 1007      1397 3 anl [anl$l_ratio_denom] = .anl [anl$l_est_d_bytes];
: 1008      1398 3 END
: 1009      1399 2 ELSE IF .anl [anl$v_est_recs] AND .anl [anl$l_est_d_recs] GTR 0
: 1010      1400 2 THEN
: 1011      1401 2 BEGIN
: 1012      1402 2 anl [anl$l_ratio_num] = .anl [anl$l_d_recs];
: 1013      1403 2 anl [anl$l_ratio_denom] = .anl [anl$l_est_d_recs];
: 1014      1404 2 END
: 1015      1405 2 ELSE
: 1016      1406 2 BEGIN
: 1017      1407 2 anl [anl$l_ratio_num] = 1;
: 1018      1408 2 anl [anl$l_ratio_denom] = 1;
: 1019      1409 2 END;
: 1020      1410 2
: 1021      1411 2 ! pre-process data
: 1022      1412 2
: 1023      1413 2 DECR depth FROM .anl [anl$b_depth] TO 1 DO
: 1024      1414 2 BEGIN
: 1025      1415 2
: 1026      1416 2 LOCAL
: 1027      1417 2 seg : REF BBLOCK;
: 1028      1418 2
: 1029      1419 2 seg = .anl [anl$l_flink];
: 1030      1420 2 WHILE .seg NEQA anl [anl$q_queue] DO
: 1031      1421 2 BEGIN
: 1032      1422 2
: 1033      1423 2 LOCAL
: 1034      1424 2 next_seg : REF BBLOCK;
: 1035      1425 2
: 1036      1426 2 next_seg = .seg [anlseg$l_flink];
: 1037      1427 2 IF
: 1038      1428 2 .seg [anlseg$b_depth] EQL .depth
: 1039      1429 2 AND
: 1040      1430 2 .seg [anlseg$l_chars] EQL 0
: 1041      1431 2 AND
: 1042      1432 2 .seg [anlseg$w_sons] EQL 0
: 1043      1433 2 THEN
: 1044      1434 2 remove_seg (anl, .seg)
: 1045      1435 2 ELSE
: 1046      1436 2 perform (huffman_size (anl, .seg)); ! compute compressed size
: 1047      1437 2 seg = .next_seg;
: 1048      1438 2 END;
: 1049      1439 2 END;
: 1050      1440 2 DECR depth FROM .anl [anl$b_depth] TO 1 DO
: 1051      1441 2 BEGIN
: 1052      1442 2
: 1053      1443 2 LOCAL
: 1054      1444 2 ptr : LONG, ! random tree listhead
: 1055      1445 2 p1 : REF VECTOR [5, LONG],
: 1056      1446 2 list : VECTOR [5 * dcx$c_max_segs, LONG],
: 1057      1447 2 seg : REF BBLOCK;
: 1058      1448 2
: 1059      1449 2 ptr = 0;
: 1060      1450 2 p1 = list [0];
: 1061      1451 2 seg = .anl [anl$l_flink];
```


dcx\$make_map - Compute mapping function

```
1062 1452 3 WHILE .seg NEQA anl [anl$q_queue] DO
1063 1453 4 BEGIN
1064 1454 4 IF
1065 1455 4 .seg [anlseg$b_depth] EQL .depth
1066 1456 4 AND
1067 1457 4 .seg [anlseg$w_sors] EQL 0
1068 1458 4 THEN
1069 1459 5 BEGIN
1070 1460 5 p1 [3] = .seg [anlseg$l_chars];
1071 1461 5 p1 [4] = .seg;
1072 1462 5 tree_insert (ptr, p1 [0]);
1073 1463 5 p1 = p1 [5];
1074 1464 4 END;
1075 1465 4 seg = .seg [anlseg$l_flink];
1076 1466 4 END;
1077 1467 4 WHILE (p1 = tree_least (ptr)) NEQA 0 DO
1078 1468 4 perform (eliminate_seg (anl, .p1 [4]));
1079 1469 4 END;
1080 1470 2
1081 1471 2 ! Compute size of map
1082 1472 2 !
1083 1473 2 id = 0;
1084 1474 2 size = dcxmap$k_length;
1085 1475 2 seg = .anl [anl$l_flink];
1086 1476 2 WHILE .seg NEQA anl [anl$q_queue] DO
1087 1477 3 BEGIN
1088 1478 3 seg [anlseg$w_id] = .id;
1089 1479 3 id = .id + 1;
1090 1480 3 size = .size + dcxsbm$k_length;
1091 1481 3 size = .size + (2*(.seg [anlseg$w_active_r]-1) + 7) / 8;
1092 1482 3 size = .size + 2*(.seg [anlseg$w_active_r]-1);
1093 1483 3 IF .seg [anlseg$l_flink] NEQA .seg [anlseg$l_blink]
1094 1484 3 THEN
1095 1485 4 size = .size + 2 * (.seg [anlseg$b_max_char] - .seg [anlseg$b_min_char] + 1);
1096 1486 3 seg = .seg [anlseg$l_flink];
1097 1487 3 END;
1098 1488 2
1099 1489 2 ! Allocate map and populate
1100 1490 2 !
1101 1491 2 perform (dcx$get_vm (.size, dcxmap));
1102 1492 2 dcxmap [dcxmap$l_size] = .size;
1103 1493 2 dcxmap [dcxmap$w_version] = dcxmap$c_version;
1104 1494 2 dcxmap [dcxmap$l_sanity] = dcxmap$c_sanity;
1105 1495 2 dcxmap [dcxmap$l_flags] = 0;
1106 1496 2 dcxmap [dcxmap$w_nsubs] = .id;
1107 1497 2 IF .id NEQ 0
1108 1498 2 THEN
1109 1499 3 BEGIN
1110 1500 4
1111 1501 4 LOCAL
1112 1502 4 seg : REF BBLOCK,
1113 1503 4 dcxsbm : REF BBLOCK;
1114 1504 4
1115 1505 4 dcxmap [dcxmap$w_sub0] = dcxmap$k_length;
1116 1506 4 seg = .anl [anl$l_flink];
1117 1507 4 dcxsbm = .dcxmap + .dcxmap [dcxmap$w_sub0];
1118 1508 4 WHILE .seg NEQA anl [anl$q_queue] DO
```


dcx\$make_map - Compute mapping function

```
1119 1509 4 BEGIN
1120 1510 4 dcxsbm [dcxsbm$w_size] = dcxsbm$length;
1121 1511 4 dcxsbm [dcxsbm$b_min_char] = .seg [anlseg$b_min_char];
1122 1512 4 dcxsbm [dcxsbm$b_max_char] = .seg [anlseg$b_max_char];
1123 1513 4 dcxsbm [dcxsbm$b_escape] = 0;
1124 1514 4 dcxsbm [dcxsbm$v_escape] = false;
1125 1515 4 dcxsbm [dcxsbm$v_unbounded] = false;
1126 1516 4 dcxsbm [dcxsbm$w_flags] = .dcxsbm [dcxsbm$w_size];
1127 1517 4 dcxsbm [dcxsbm$w_size] = .dcxsbm [dcxsbm$w_size] + (2*(.seg [anlseg$w_active_r]-1) + 7) / 8;
1128 1518 4 dcxsbm [dcxsbm$w_nodes] = .dcxsbm [dcxsbm$w_size];
1129 1519 4 dcxsbm [dcxsbm$w_size] = .dcxsbm [dcxsbm$w_size] + 2*(.seg [anlseg$w_active_r]-1);
1130 1520 4 IF .seg [anlseg$l_flink] NEQA .seg [anlseg$l_blink]
1131 1521 4 THEN
1132 1522 5 BEGIN
1133 1523 5 dcxsbm [dcxsbm$w_next] = .dcxsbm [dcxsbm$w_size];
1134 1524 5 dcxsbm [dcxsbm$w_size] = .dcxsbm [dcxsbm$w_size] + 2 * (.seg [anlseg$b_max_char] - .seg [anlseg$
1135 1525 5 END
1136 1526 4 ELSE
1137 1527 4 dcxsbm [dcxsbm$w_next] = 0;
1138 1528 4 build_map_seg (anl, .seg, .dcxsbm);
1139 1529 4 dcxsbm = .dcxsbm + .dcxsbm [dcxsbm$w_size];
1140 1530 4 seg = .seg [anlseg$l_flink];
1141 1531 3 END;
1142 1532 4 IF .dcxsbm NEQA (.dcxmap + .dcxmap [dcxmap$l_size])
1143 1533 3 THEN
1144 1534 3 RETURN ss$_accvio;
1145 1535 2 END;
1146 1536 2 .map_addr = .dcxmap;
1147 1537 2 IF NOT NULLPARAMETER (3)
1148 1538 2 THEN
1149 1539 2 .map_size = .size;
1150 1540 2 RETURN dcx$_normal;
1151 1541 2
1152 1542 1 END;
```

! Of dcx\$make_map

				00FC 00000	.ENTRY	DCX\$MAKE MAP, Save R2,R3,R4,R5,R6,R7	: 1340
		5E	AFFC	CE 9E 00002	MOVAB	-20484(SP), SP	
54	04	BC		14 C1 00007	ADDL3	#20, @CONTEXT_ADDR, R4	: 1370
55	04	BC		10 C1 0000C	ADDL3	#16, @CONTEXT_ADDR, R5	: 1371
			08	BC D4 00011	CLRL	@MAP_ADDR	: 1378
		03		6C 91 00014	CMPB	(AP), #3	: 1379
				08 1F 00017	BLSSU	1\$	
			0C	AC D5 00019	TSTL	12(AP)	
				03 13 0001C	BEQL	1\$	
			0C	BC D4 0001E	CLRL	@MAP_SIZE	: 1381
				51 D4 00021	CLRL	R1	: 1385
		50	04	BC D0 00023	MOVL	@CONTEXT_ADDR, R0	
				0000G 30 00027	BSBW	DCX\$CTX_CHECK	
		70		50 E9 0002A	BLBC	STATUS, -8\$	
11		64		02 E1 0002D	BBC	#2, (R4), 2\$	: 1393
			0C	A4 D5 00031	TSTL	12(R4)	
				0C 15 00034	BLEQ	2\$	
	20	A4	04	A4 D0 00036	MOVL	4(R4), 32(R4)	: 1396

		24	A4	0C	A4	D0	0003B	MOVL	12(R4), 36(R4)	1397	
					1D	11	00040	BRB	4\$	1393	
11		64			03	E1	00042	BBC	#3, (R4), 3\$	1399	
				10	A4	D5	00046	TSTL	16(R4)		
					0C	15	00049	BLEQ	3\$		
		20	A4	08	A4	D0	0004B	MOVL	8(R4), 32(R4)	1402	
		24	A4	10	A4	D0	00050	MOVL	16(R4), 36(R4)	1403	
					08	11	00055	BRB	4\$	1399	
		20	A4		01	D0	00057	MOVL	#1, 32(R4)	1407	
		24	A4		01	D0	0005B	MOVL	#1, 36(R4)	1408	
			53	14	A4	9A	0005F	MOVZBL	20(R4), DEPTH	1413	
					53	D6	00063	INCL	DEPTH		
					3F	11	00065	BRB	10\$		
		52		18	A4	D0	00067	MOVL	24(R4), SEG	1419	
		50		18	A4	9E	0006B	MOVAB	24(R4), R0	1420	
		50			52	D1	0006F	CMPL	SEG, R0		
					32	13	00072	BEQL	10\$		
		56			62	D0	00074	MOVL	(SEG), NEXT_SEG	1426	
53	14	A2	08		00	ED	00077	CMPZV	#0, #8, 20(SEG), DEPTH	1428	
					15	12	0007D	BNEQ	7\$		
				2C	A2	D5	0007F	TSTL	44(SEG)	1430	
					10	12	00082	BNEQ	7\$		
				1C	A2	B5	00084	TSTW	28(SEG)	1432	
					0B	12	00087	BNEQ	7\$		
					52	DD	00089	PUSHL	SEG	1434	
					54	DD	0008B	PUSHL	R4		
		FC42	CF		02	FB	0008D	CALLS	#2, REMOVE_SEG		
					0D	11	00092	BRB	9\$		
					52	DD	00094	PUSHL	SEG	1436	
					54	DD	00096	PUSHL	R4		
		FA88	CF		02	FB	00098	CALLS	#2, HUFFMAN_SIZE		
		01			50	E8	0009D	BLBS	STATUS, 9\$		
					04	000A0	RET				
		52			56	D0	000A1	MOVL	NEXT_SEG, SEG	1437	
					C5	11	000A4	BRB	6\$	1420	
		BE			53	F5	000A6	SOBGTR	DEPTH, 5\$	1413	
		56		14	A4	9A	000A9	MOVZBL	20(R4), DEPTH	1440	
					56	D6	000AD	INCL	DEPTH		
					009D	31	000AF	BRW	24\$		
					6E	D4	000B2	CLRL	PTR	1449	
		52		04	AE	9E	000B4	MOVAB	LIST, P1	1450	
		51		18	A4	D0	000B8	MOVL	24(R4), SEG	1451	
		50		18	A4	9E	000BC	MOVAB	24(R4), R0	1452	
		50			51	D1	000C0	CMPL	SEG, R0		
					4B	13	000C3	BEQL	18\$		
					00	ED	000C5	CMPZV	#0, #8, 20(SEG), DEPTH	1455	
					3E	12	000CB	BNEQ	17\$		
				1C	A1	B5	000CD	TSTW	28(SEG)	1457	
					39	12	000D0	BNEQ	17\$		
				2C	A1	D0	000D2	MOVL	44(SEG), 12(P1)	1460	
		0C	A2		51	D0	000D7	MOVL	SEG, 16(P1)	1461	
		10	A2		62	7C	000DB	CLRQ	(P1)	1462	
				08	A2	D4	000DD	CLRL	8(P1)		
					6E	9E	000E0	MOVAB	PTR, -H		
		53			63	D0	000E3	MOVL	(-H), -R0		
		50			1D	13	000E6	BEQL	18\$		
					0C	A2	D1	000E8	CMPL	12(P1), 12(R0)	

		05	1E	000ED	BGEQU	14\$	
	53	50	D0	000EF	MOVL	R0, _H	
		EF	11	000F2	BRB	13\$	
		06	1B	000F4	14\$: BLEQU	15\$	
	53	04	A0	9E 000F6	MOVAB	4(R0), _H	
		E7	11	000FA	BRB	13\$	
08	A2	08	A0	D0 000FC	15\$: MOVL	8(R0), 8(P1)	
	53	08	A0	9E 00101	MOVAB	8(R0), _H	
	63		82	7E 00105	16\$: MOVAQ	(P1)+, 7_H)	
	52		0C	C0 00108	ADDL2	#12, P1	
	51		61	D0 0010B	17\$: MOVL	(SEG), SEG	
		AC	11	0010E	BRB	12\$	
	53		6E	9E 00110	18\$: MOVAB	PTR, _H	
		63	D5	00113	TSTL	(_H)	
		04	12	00115	BNEQ	19\$	
		52	D4	00117	CLRL	P1	
		24	11	00119	BRB	23\$	
		00	B3	D5 0011B	19\$: TSTL	20(_H)	
		05	13	0011E	BEQL	20\$	
	53		63	D0 00120	MOVL	(_H), _H	
		F6	11	00123	BRB	19\$	
	51		63	D0 00125	20\$: MOVL	(_H), R1	
	50	08	A1	D0 00128	MOVL	8(R1), _Q	
		07	13	0012C	BEQL	21\$	
08	A1	08	A0	D0 0012E	MOVL	8(_Q), 8(R1)	
		07	11	00133	BRB	22\$	
	50		51	D0 00135	21\$: MOVL	R1, _Q	
	63	04	A0	D0 00138	MOVL	4(_Q), (_H)	
	52		50	D0 0013C	22\$: MOVL	R0, P1	
		0E	13	0013F	23\$: BEQL	24\$	
		10	A2	DD 00141	PUSHL	16(P1)	
			54	DD 00144	PUSHL	R4	
FC07	CF		02	FB 00146	CALLS	#2, ELIMINATE_SEG	
	C2		50	E8 0014B	BLBS	STATUS, 18\$	
			04	0014E	RET		
	02		56	F5 0014F	24\$: SOBGTR	DEPTH, 25\$	
			03	11 00152	BRB	26\$	
		FF	5B	31 00154	25\$: BRW	11\$	
			53	D4 00157	26\$: CLRL	ID	
	56		14	D0 00159	MOVL	#20, SIZE	
	50	18	A4	D0 0015C	MOVL	24(R4), SEG	
	51	18	A4	9E 00160	27\$: MOVAB	24(R4), R1	
	51		50	D1 00164	CMPL	SEG, R1	
			3A	13 00167	BEQL	29\$	
0C	A0		53	B0 00169	MOVW	ID, 12(SEG)	
			53	D6 0016D	INCL	ID	
	56		0C	C0 0016F	ADDL2	#12, SIZE	
	51	12	A0	3C 00172	MOVZWL	18(SEG), R1	
	51		01	78 00176	ASHL	#1, R1, R2	
	52		05	C0 0017A	ADDL2	#5, R2	
	52		08	C6 0017D	DIVL2	#8, R2	
	56		52	C0 00180	ADDL2	R2, SIZE	
	56	FE	A6	41 3E 00183	MOVAW	-2(SIZE)[R1], SIZE	
			60	D1 00188	CMPL	(SEG), 4(SEG)	
04	A0		10	13 0018C	BEQL	28\$	
	51	16	A0	9A 0018E	MOVZBL	22(SEG), R1	
	52	15	A0	9A 00192	MOVZBL	21(SEG), R2	

	51		52	C2	00196	SUBL2	R2, R1		
	56	02	A641	3E	00199	MOVAW	2(SIZE)[R1], SIZE		
	50		60	D0	0019E	28\$:	MOVL	(SEG), SEG	1486
			BD	11	001A1		BRB	27\$	1476
			55	DD	001A3	29\$:	PUSHL	R5	1491
			56	DD	001A5		PUSHL	SIZE	
0000G	CF		02	FB	001A7		CALLS	#2, DCX\$GET_VM	
	01		50	E8	001AC		BLBS	STATUS, 30\$	
			04	001AF			RET		
	55		65	D0	001B0	30\$:	MOVL	(R5), R5	1492
	65		56	D0	001B3		MOVL	SIZE, (R5)	
		04	A5	B4	001B6		CLRW	4(R5)	1493
08	A5	5BF5A3A7	8F	D0	001B9		MOVL	#1542324871, 8(R5)	1494
		0C	A5	D4	001C1		CLRL	12(R5)	1495
10	A5		53	B0	001C4		MOVW	ID, 16(R5)	1496
			53	D5	001C8		TSTL	ID	1497
			03	12	001CA		BNEQ	31\$	
			0090	31	001CC		BRW	36\$	
12	A5		14	B0	001CF	31\$:	MOVW	#20, 18(R5)	1505
	52	18	A4	D0	001D3		MOVL	24(R4), SEG	1506
	53	12	A5	3C	001D7		MOVZWL	18(R5), DCXSBM	1507
	53		55	C0	001DB		ADDL2	R5, DCXSBM	
	50	18	A4	9E	001DE	32\$:	MOVAB	24(R4), R0	1508
	50		52	D1	001E2		CMPL	SEG, R0	
			6B	13	001E5		BEQL	35\$	
	63		0C	B0	001E7		MOVW	#12, (DCXSBM)	1510
02	A3	15	A2	B0	001EA		MOVW	21(SEG), 2(DCXSBM)	1511
04	A3	03FF	8F	AA	001EF		BICW2	#1023, 4(DCXSBM)	1515
06	A3		63	B0	001F5		MOVW	(DCXSBM), 6(DCXSBM)	1516
	50	12	A2	3C	001F9		MOVZWL	18(SEG), R0	1517
51	50		01	78	001FD		ASHL	#1, R0, R1	
	51		05	C0	00201		ADDL2	#5, R1	
	51		08	C6	00204		DIVL2	#8, R1	
	63		51	A0	00207		ADDW2	R1, (DCXSBM)	
08	A3		63	B0	0020A		MOVW	(DCXSBM), 8(DCXSBM)	1518
	51		63	3C	0020E		MOVZWL	(DCXSBM), R1	1519
	57	FE	A140	3E	00211		MOVAW	-2(R1)[R0], R7	
	63		57	B0	00216		MOVW	R7, (DCXSBM)	
04	A2		62	D1	00219		CMPL	(SEG), 4(SEG)	1520
			1C	13	0021D		BEQL	33\$	
0A	A3		63	B0	0021F		MOVW	(DCXSBM), 10(DCXSBM)	1523
	51		63	3C	00223		MOVZWL	(DCXSBM), R1	1524
	50	16	A2	9A	00226		MOVZBL	22(SEG), R0	
	57	15	A2	9A	0022A		MOVZBL	21(SEG), R7	
	50		57	C2	0022E		SUBL2	R7, R0	
	57	02	A140	3E	00231		MOVAW	2(R1)[R0], R7	
	63		57	B0	00236		MOVW	R7, (DCXSBM)	
			03	11	00239		BRB	34\$	1520
		0A	A3	B4	0023B	33\$:	CLRW	10(DCXSBM)	1527
			0C	BB	0023E	34\$:	PUSHR	#*M<R2,R3>	1528
			54	DD	00240		PUSHL	R4	
FC1D	CF		03	FB	00242		CALLS	#3, BUILD_MAP_SEG	
	50		63	3C	00247		MOVZWL	(DCXSBM), R0	1529
	53		50	C0	0024A		ADDL2	R0, DCXSBM	
	52		62	D0	0024D		MOVL	(SEG), SEG	1530
			8C	11	00250		BRB	32\$	1508
50	55		65	C1	00252	35\$:	ADDL3	(R5), R5, R0	1532

DCX_ANALYZE
V04=000

dcx\$make_map - Compute mapping function

H 12
15-Sep-1984 23:38:18
14-Sep-1984 12:15:55

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCX.SRC]ANALYZE.B32;1 Page 41
(11)

50		53	D1	00256	CMPL	DCXSBM, R0	:	
		04	13	00259	BEQL	36\$	:	
50		0C	D0	0025B	MOVL	#12, R0	: 1534	
			04	0025E	RET		:	
08	BC	55	D0	0025F	36\$: MOVL	R5, @MAP_ADDR	: 1536	
	03	6C	91	00263	CMPB	(AP), #3	: 1537	
		09	1F	00266	BLSSU	37\$	:	
		0C	AC	D5	00268	TSTL	12(AP)	:
		04	13	0026B	BEQL	37\$	:	
0C	BC	56	D0	0026D	MOVL	SIZE, @MAP_SIZE	: 1539	
	50	8F	D0	00271	37\$: MOVL	#DCX\$_NORMAL, R0	: 1540	
	00000000G	04	00278	RET			: 1542	

; Routine Size: 633 bytes, Routine Base: \$CODE\$ + 07D9

```
1154 1543 1 %SBTTL 'dcx$analyze_done -- release context and map'
1155 1544 1
1156 1545 1 GLOBAL ROUTINE dcx$analyze_done (context_addr) =
1157 1546 2 BEGIN
1158 1547 2 ++
1159 1548 2
1160 1549 2 Release context and map
1161 1550 2
1162 1551 2 Inputs:
1163 1552 2
1164 1553 2 context_addr.mz.r Address of context longword
1165 1554 2
1166 1555 2 Outputs:
1167 1556 2
1168 1557 2 context_addr.mz.r Context block accumulates data
1169 1558 2
1170 1559 2 Return value:
1171 1560 2
1172 1561 2 status.wlc.v
1173 1562 2
1174 1563 2 dcx$_normal All is well
1175 1564 2 --
1176 1565 2
1177 1566 2 BIND
1178 1567 2 ctx = .context_addr : REF BBLOCK, ! address of context block
1179 1568 2 anl = ctx [ctx$l_specific] : BBLOCK;
1180 1569 2
1181 1570 2 LOCAL
1182 1571 2 anlseg : REF BBLOCK;
1183 1572 2
1184 1573 2 perform (dcx$ctx_check (.ctx, ctx$c_anlyz));
1185 1574 2 WHILE NOT remque (.anl [anl$l_flink], anlseg) DO
1186 1575 2 perform (dcx$free_vm (.anlseg [anlseg$l_size], .anlseg));
1187 1576 2 perform (dcx$free_vm (.ctx [ctx$l_size], .ctx));
1188 1577 2 ctx = 0; ! mark context as gone
1189 1578 2 RETURN dcx$_normal;
1190 1579 2
1191 1580 1 END; ! Of dcx$analyze_done
```

			001C 00000	.ENTRY	DCX\$ANALYZE DONE, Save R2,R3,R4	1545
	54	04	AC D0 00002	MOVL	CONTEXT_ADDR, R4	1567
53	64		14 C1 00006	ADDL3	#20, (R4), R3	1568
			51 D4 0000A	CLRL	R1	1573
	50		64 D0 0000C	MOVL	(R4), R0	
			0000G 30 0000F	BSBW	DCX\$CTX_CHECK	
			10 11 00012	BRB	2\$	
	52	18	B3 0F 00014 1\$:	REMQUE	@24(R3), ANLSEG	1574
			0E 1D 00018	BVS	3\$	
			52 DD 0001A	PUSHL	ANLSEG	1575
		08	A2 DD 0001C	PUSHL	8(ANLSEG)	
	0000G CF		02 FB 0001F	CALLS	#2, DCX\$FREE_VM	
	ED		50 E8 00024 2\$:	BLBS	STATUS, 1\$	
			04 00027	RET		

DCX_ANALYZE
V04=000

dcx\$analyze_done -- release context and map

J 12
15-Sep-1984 23:38:18
14-Sep-1984 12:15:55

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCX.SRC]ANALYZE.B32;1
Page 43
(12)

			64	DD	00028	3\$:	PUSHL	(R4)	:	1576
			B4	DD	0002A		PUSHL	@0(R4)	:	
0000G	CF		02	FB	0002D		CALLS	#2, DCX\$FREE_VM	:	
	09		50	E9	00032		BLBC	STATUS, 4\$	:	
			64	D4	00035		CLRL	(R4)	:	1577
			8F	D0	00037		MOVL	#DCX\$_NORMAL, R0	:	1578
			04	0003E	4\$:		RET		:	1580

; Routine Size: 63 bytes, Routine Base: \$CODE\$ + 0A52

DCX_ANALYZE
V04=000

dcx\$analyze_done -- release context and map

K 12
15-Sep-1984 23:38:18
14-Sep-1984 12:15:55
! Of module analyze

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[DCX.SRC]ANALYZE.B32;1 Page 44 (13)

: 1193
: 1194
1581 1 END
1582 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	2705	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	7	0	581	00:01.0

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:ANALYZE/OBJ=OBJ\$:ANALYZE MSRC\$:ANALYZE/UPDATE=(ENHS:ANALYZE)

: Size: 2705 code + 0 data bytes
: Run Time: 00:58.4
: Elapsed Time: 02:55.8
: Lines/CPU Min: 1624
: Lexemes/CPU-Min: 27047
: Memory Used: 272 pages
: Compilation Complete

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

