

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

1

7

56

9

1

3


```
1 0001 0 MODULE dcx_subs ( ! Miscellaneous routines
2 0002 0 LANGUAGE (BLISS32),
3 0003 0 IDENT = 'V04-000'
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 ++
32 0032 1
33 0033 1 FACILITY:
34 0034 1
35 0035 1 DCX -- Data Compression / Expansion Facility
36 0036 1
37 0037 1 ABSTRACT:
38 0038 1
39 0039 1 The Data Compression / Expansion procedures provide a general
40 0040 1 method for reducing the storage requirement for a arbitrary data.
41 0041 1
42 0042 1 ENVIRONMENT:
43 0043 1
44 0044 1 VAX native, user mode.
45 0045 1
46 0046 1 --
47 0047 1
48 0048 1
49 0049 1 AUTHOR: David Thiel
50 0050 1
51 0051 1 CREATION DATE: July, 1981
52 0052 1
53 0053 1 MODIFIED BY:
54 0054 1
55 0055 1 --
```


DCX SUBS
V04=000

B 16
15-Sep-1984 23:43:44
14-Sep-1984 12:16:03

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

Page 2
(2)

Declarations

```
: 57 0056 1 %SBTTL 'Declarations';
: 58 0057 1
: 59 0058 1 LIBRARY
: 60 0059 1 'sys$library:starlet'; ! System macros
: 61 0060 1 REQUIRE
: 62 0061 1 'prefix'; ! DCX macro definitions
: 63 0204 1 REQUIRE
: 64 0205 1 'dcxdef'; ! DCX public structure definitions
: 65 0299 1 REQUIRE
: 66 0300 1 'dcxprvdef'; ! DCX private structure definitions
: 67 0466 1
: 68 0467 1 EXTERNAL ROUTINE
: 69 0468 1 lib$free_vm : ADDRESSING_MODE (GENERAL),
: 70 0469 1 lib$get_vm : ADDRESSING_MODE (GENERAL);
: 71 0470 1
: 72 0471 1 EXTERNAL LITERAL
: 73 0472 1 dcx$_invctx, ! Invalid context block
: 74 0473 1 dcx$_invmap, ! Invalid map
: 75 0474 1 dcx$_normal;
: 76 0475 1
: 77 0476 1 FORWARD ROUTINE
: 78 0477 1 dcx$get_vm, ! Allocate memory
: 79 0478 1 dcx$free_vm, ! Deallocate memory
: 80 0479 1 dcx$map_check : lkg_map_check, ! Check map
: 81 0480 1 dcx$ctx_check : lkg_ctx_check, ! Check context
: 82 0481 1 dcx$long_move : lkg_long_move NOVALUE; ! Arbitrary length data copy
```


dcx\$get_vm -- allocate virtual memory

```

84 0482 1 %SBTTL 'dcx$get_vm -- allocate virtual memory'
85 0483 1
86 0484 1 GLOBAL ROUTINE dcx$get_vm (bytes, addr) =
87 0485 2 BEGIN
88 0486 2 ++
89 0487 2
90 0488 2 Allocate memory for the data compression / expansion
91 0489 2 facility. The allocated memory is zeroed.
92 0490 2
93 0491 2 Inputs:
94 0492 2
95 0493 2 bytes Number of bytes to allocate
96 0494 2
97 0495 2 Outputs:
98 0496 2
99 0497 2 addr Address in which to store addr
100 0498 2 of allocated memory
101 0499 2
102 0500 2 Return value:
103 0501 2
104 0502 2 dcx$_normal All is well
105 0503 2 lib$_insvirmem Error allocation memory
106 0504 2 --
107 0505 2
108 0506 2 LOCAL
109 0507 2 p, ! pointer for zeroing memory
110 0508 2 c, ! remaining byte count
111 0509 2 status : LONG; ! return status
112 0510 2
113 0511 2 IF .bytes EQL 0
114 0512 2 THEN
115 0513 2 BEGIN
116 0514 2 .addr = 0;
117 0515 2 RETURN dcx$_normal;
118 0516 2 END
119 0517 2 ELSE IF NOT (status = lib$get_vm (bytes, .addr))
120 0518 2 THEN
121 0519 2 BEGIN
122 0520 2 .addr = 0;
123 0521 2 RETURN .status;
124 0522 2 END
125 0523 2 ELSE
126 0524 2 BEGIN
127 0525 2 p = .addr;
128 0526 2 c = .bytes;
129 0527 2 WHILE .c GTRU %X'FFFF' DO
130 0528 2 BEGIN
131 0529 2 p = CH$FILL (0, %X'FFFF', .p);
132 0530 2 c = .c - %X'FFFF';
133 0531 2 END;
134 0532 2 CH$FILL (0, .c, .p);
135 0533 2 RETURN dcx$_normal;
136 0534 2 END;
137 0535 1 END; ! of dcx$get_vm
```

.TITLE DCX SUBS

DCX_SUBS
V04=000

dcx\$get_vm -- allocate virtual memory

D 16
15-Sep-1984 23:43:44
14-Sep-1984 12:16:03

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

Page 4
(3)

.IDENT \V04-000\

.EXTRN LIB\$ANALYZE_SDESC_R2
.EXTRN LIB\$FREE_VM, LIB\$GET_VM
.EXTRN DCX\$_INVCTX, DCX\$_INVMAP
.EXTRN DCX\$_NORMAL

.PSECT \$CODE\$,NOWRT,2

.ENTRY DCX\$GET_VM, Save R2,R3,R4,R5,R6

TSTL	BYTES	: 0484
BNEQ	1\$	: 0511
CLRL	@ADDR	: 0514
BRB	5\$	: 0517
PUSHL	ADDR	
PUSHAB	BYTES	
CALLS	#2, LIB\$GET_VM	
BLBS	STATUS, 2\$	
CLRL	@ADDR	: 0520
RET		: 0521
MOVL	@ADDR, P	: 0525
MOVL	BYTES, C	: 0526
CMPL	C, #65535	: 0527
BLEQU	4\$	
MOVC5	#0, (SP), #0, #65535, (P)	: 0529
MOVAB	-65535(R6), C	: 0530
BRB	3\$	: 0527
MOVC5	#0, (SP), #0, C, (P)	: 0532
MOVL	#DCX\$_NORMAL, R0	: 0533
RET		: 0535

04	AC	D5	00002	
05	12	00005		
08	BC	D4	00007	
3C	11	0000A		
08	AC	DD	0000C	1\$:
04	AC	9F	0000F	
02	FB	00012		
50	E8	00019		
08	BC	D4	0001C	
04	0001F			
08	BC	D0	00020	2\$:
04	AC	D0	00024	
56	D1	00028		3\$:
11	1B	0002F		
00	2C	00031		
63	00038			
E6	9E	00039		
E6	11	00040		
00	2C	00042		4\$:
63	00047			
50	D0	00048		5\$:
04	0004F			

00000000G 00
04

53
56

0000FFFF 8F

FFFF 8F 00

6E

56 FFFF0001

56 00

6E

50 00000000G

8F

; Routine Size: 80 bytes, Routine Base: \$CODE\$ + 0000

DCX SUBS
V04=000

E 16
15-Sep-1984 23:43:44
14-Sep-1984 12:16:03

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

Page 5
(4)

dcx\$free_vm -- free virtual memory

```

139 0536 1 %SBTTL 'dcx$free_vm -- free virtual memory'
140 0537 1
141 0538 1 GLOBAL ROUTINE dcx$free_vm (bytes, addr) =
142 0539 2 BEGIN
143 0540 2 ++
144 0541 2
145 0542 2 Free memory for the data compression / expansion
146 0543 2 facility.
147 0544 2
148 0545 2 Inputs:
149 0546 2
150 0547 2 bytes Number of bytes to free
151 0548 2 addr Address of block to free
152 0549 2
153 0550 2 Outputs:
154 0551 2
155 0552 2 NONE
156 0553 2
157 0554 2 Return value:
158 0555 2
159 0556 2 dcx$_normal All is well
160 0557 2 lib$_insvirmem Error allocating memory
161 0558 2 --
162 0559 2
163 0560 2 IF .bytes NEQ 0
164 0561 2 THEN
165 0562 2 perform (lib$free_vm (bytes, addr));
166 0563 2 RETURN dcx$_normal;
167 0564 2
168 0565 1 END; ! of dcx$free_vm
```

		0000 00000	.ENTRY	DCX\$FREE_VM, Save nothing	: 0538
04	AC	D5 00002	TSTL	BYTES	: 0560
	10	13 00005	BEQL	1\$	:
08	AC	9F 00007	PUSHAB	ADDR	: 0562
04	AC	9F 0000A	PUSHAB	BYTES	:
00000000G	00	02 FB 0000D	CALLS	#2, LIB\$FREE_VM	:
	07	50 E9 00014	BLBC	STATUS, 2\$	:
50 00000000G	8F	D0 00017	MOVL	#DCX\$_NORMAL, R0	: 0563
		04 0001E	RET		: 0565

; Routine Size: 31 bytes, Routine Base: \$CODE\$ + 0050


```
dcx$ctx_check -- check context block

: 170 0566 1 %SBTTL 'dcx$ctx_check -- check context block'
: 171 0567 1
: 172 0568 1 GLOBAL ROUTINE dcx$ctx_check (ctx : REF BBLOCK, type) : lkg_ctx_check =
: 173 0569 2 BEGIN
: 174 0570 2 ++
: 175 0571 2
: 176 0572 2 Check context block for validity.
: 177 0573 2
: 178 0574 2 Inputs:
: 179 0575 2
: 180 0576 2 ctx Address of context block
: 181 0577 2 type Required context block type
: 182 0578 2
: 183 0579 2 Outputs:
: 184 0580 2
: 185 0581 2 NONE
: 186 0582 2
: 187 0583 2 Return value:
: 188 0584 2
: 189 0585 2 dcx$_normal All is well
: 190 0586 2 dcx$_invctx Invalid context block
: 191 0587 2 dcx$_invmap Invalid map
: 192 0588 2 --
: 193 0589 2
: 194 0590 2 IF .ctx [ctx$l_size] LSSU ctx$k_fixed_len
: 195 0591 2 THEN
: 196 0592 2 RETURN dcx$_invctx
: 197 0593 2 ELSE IF .ctx [ctx$l_sanity] NEQ ctx$c_sanity
: 198 0594 2 THEN
: 199 0595 2 RETURN dcx$_invctx
: 200 0596 2 ELSE IF .ctx [ctx$w_version] NEQ ctx$c_version
: 201 0597 2 THEN
: 202 0598 2 RETURN dcx$_invctx
: 203 0599 2 ELSE IF .ctx [ctx$b_type] NEQ .type
: 204 0600 2 THEN
: 205 0601 2 RETURN dcx$_invctx
: 206 0602 2 ELSE
: 207 0603 2 RETURN dcx$_normal;
: 208 0604 2
: 209 0605 1 END;

! of dcx$ctx_check
```

			14		60	D1 00000	DCX\$CTX_CHECK::			
					17	1F 00003	CMPL	(CTX), #20		0590
					A0	D1 00005	BLSSU	1\$		
		4F317C65	8F	0C	0D	12 0000D	CMPL	12(CTX), #1328643173		0593
				08	A0	B5 0000F	BNEQ	1\$		
					08	12 00012	TSTW	8(CTX)		0596
					00	ED 00014	BNEQ	1\$		
					08	13 0001A	CMPZV	#0, #8, 4(CTX), TYPE		0599
51	04	A0	08		00	ED 00014	BEQL	2\$		
					50	00000000G	MOVL	#DCX\$_INVCTX, R0		0601
					50	00000000G	RSB			
					8F	D0 00024	MOVL	#DCX\$_NORMAL, R0		0603

DCX SUBS
V04=000

dcx\$ctx_check -- check context block

05 0002B

RSB

; Routine Size: 44 bytes, Routine Base: \$CODE\$ + 006F

G 16
15-Sep-1984 23:43:44
14-Sep-1984 12:16:03

VAX-11 Bliss-32 V4.0-742
DISK\$VMMASTER:[DCX.SRC]SUBS.B32;1

Page 7
(5)

; 0605

```
dcx$map_check -- check map

: 211 0606 1 %SBTTL 'dcx$map_check -- check map'
: 212 0607 1
: 213 0608 1 GLOBAL ROUTINE dcx$map_check (dcxmap : REF BBLOCK) : lkg_map_check =
: 214 0609 2 BEGIN
: 215 0610 2 ++
: 216 0611 2
: 217 0612 2 Check map for validity.
: 218 0613 2
: 219 0614 2 Inputs:
: 220 0615 2
: 221 0616 2 dcxmap Address of map
: 222 0617 2
: 223 0618 2 Outputs:
: 224 0619 2
: 225 0620 2 NONE
: 226 0621 2
: 227 0622 2 Return value:
: 228 0623 2
: 229 0624 2 dcx$_normal All is well
: 230 0625 2 dcx$_invmap Invalid map
: 231 0626 2 --
: 232 0627 2
: 233 0628 2 IF .dcxmap [dcxmap$l_size] LSSU dcxmap$k_length
: 234 0629 2 THEN
: 235 0630 2 RETURN dcx$_invmap
: 236 0631 2 ELSE IF .dcxmap [dcxmap$l_sanity] NEQ dcxmap$c_sanity
: 237 0632 2 THEN
: 238 0633 2 RETURN dcx$_invmap
: 239 0634 2 ELSE IF .dcxmap [dcxmap$w_version] NEQ dcxmap$c_version
: 240 0635 2 THEN
: 241 0636 2 RETURN dcx$_invmap
: 242 0637 2 ELSE
: 243 0638 2 RETURN dcx$_normal;
: 244 0639 2
: 245 0640 1 END;

! of dcx$map_check
```

14	60	D1	00000	DCX\$MAP_CHECK::				
				CMPL	(DCXMAP), #20			: 0628
				BLSSU	1\$			: 0631
5BF5A3A7	8F	08	A0	D1	00005	CMPL	8(DCXMAP), #1542824871	: 0634
		04	05	12	0000D	BNEQ	1\$	: 0636
			A0	B5	0000F	TSTW	4(DCXMAP)	: 0638
			08	13	00012	BEQL	2\$	: 0640
50 00000000G	8F		D0	00014	1\$:	MOVL	#DCX\$_INVMAP, R0	
			05	0001B		RSB		
50 00000000G	8F		D0	0001C	2\$:	MOVL	#DCX\$_NORMAL, R0	
			05	00023		RSB		

; Routine Size: 36 bytes, Routine Base: \$CODE\$ + 009B

dcx\$long_move -- long data copy

```

: 247 0641 1 %SBTTL 'dcx$long_move -- long data copy'
: 248 0642 1
: 249 0643 1 GLOBAL ROUTINE dcx$long_move (size, source, target) : lkg_long_move NOVALUE =
: 250 0644 2 BEGIN
: 251 0645 2 ++
: 252 0646 2
: 253 0647 2 Copy arbitrary sized block of data.
: 254 0648 2
: 255 0649 2 Inputs:
: 256 0650 2
: 257 0651 2 size Length of data to copy
: 258 0652 2 source Address of data to copy
: 259 0653 2 target Address of destination
: 260 0654 2
: 261 0655 2 Outputs:
: 262 0656 2
: 263 0657 2 NONE
: 264 0658 2
: 265 0659 2 Return value:
: 266 0660 2
: 267 0661 2 NONE
: 268 0662 2 --
: 269 0663 2
: 270 0664 2 BUILTIN
: 271 0665 2 movc3;
: 272 0666 2
: 273 0667 2 WHILE .size GTRU %X'FFFF' DO
: 274 0668 2 BEGIN
: 275 0669 2
: 276 0670 2 LOCAL
: 277 0671 2 dummy0,
: 278 0672 2 dummy2;
: 279 0673 2
: 280 0674 2 movc3 (%REF(%X'FFFF'), .source, .target; dummy0, source, dummy2, target);
: 281 0675 2 size = .size - %X'FFFF';
: 282 0676 2 END;
: 283 0677 2 CH$MOVE (.size, .source, .target);
: 284 0678 2 RETURN;
: 285 0679 2
: 286 0680 1 END; ! of dcx$long_move
```

		0078	8F	BB	00000	DCX\$LONG MOVE::		
						PUSHR	#^M<R3,R4,R5,R6>	: 0643
						MOVL	R2, R3	
						MOVL	R0, R6	
	0000FFFF					CMPL	SIZE, #65535	: 0667
						BLEQU	2\$	
63		61	FFFF			MOV C3	#65535, (SOURCE), (TARGET)	: 0674
		56	FFFF0001			MOVAB	-65535(R6), SIZE	: 0675
						BRB	1\$	: 0667
63		61				MOV C3	SIZE, (SOURCE), (TARGET)	: 0677
						POPR	#^M<R3,R4,R5,R6>	: 0680
		0078		8F	BA	00026		
				05	0002A	RSB		

DCX_SUBS
V04=000

dcx\$long_move -- long data copy

J 16
15-Sep-1984 23:43:44
14-Sep-1984 12:16:03

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

Page 10
(7)

; Routine Size: 43 bytes, Routine Base: \$CODE\$ + 00BF

DCX_SUBS
V04=000

dcx\$long_move -- long data copy

: 288
: 289

0681 1 END
0682 0 ELUDOM

K 16
15-Sep-1984 23:43:44
14-Sep-1984 12:16:03
! Of module dcx_subrs

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

Page 11
(8)

PSECT SUMMARY

:
: Name Bytes Attributes
: \$CODE\$ 234 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

:
: File ----- Symbols ----- Pages Processing
: Total Loaded Percent Mapped Time
: _\$255\$DUA28:[SYSLIB]STARLET.L32;1 9776 6 0 581 00:01.0

COMMAND QUALIFIERS

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

: Size: 234 code + 0 data bytes
: Run Time: 00:08.4
: Elapsed Time: 00:33.5
: Lines/CPU Min: 4854
: Lexemes/CPU-Min: 26911
: Memory Used: 76 pages
: Compilation Complete

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

