


```

1 0001 0 MODULE  exch$util                                %TITLE 'Facility-wide misc routines'
2 0002 0
3 0003 0          (
4 0004 0          IDENT = 'V04-000',
5 0005 0          ADDRESSING_MODE (EXTERNAL=LONG_RELATIVE, NONEXTERNAL=WORD_RELATIVE)
6 0006 1 BEGIN
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 FACILITY:      EXCHANGE - Foreign volume interchange facility
33 0033 1
34 0034 1 ABSTRACT:      Miscellaneous utility routines
35 0035 1
36 0036 1 ENVIRONMENT:    VAX/VMS User mode
37 0037 1
38 0038 1 AUTHOR:        CW Hobbs          CREATION DATE: 8-July-1982
39 0039 1
40 0040 1 MODIFIED BY:
41 0041 1
42 0042 1          V03-002 CWH3002          CW Hobbs          12-Apr-1984
43 0043 1          Change the getdvi to use FULLDEVNAM.
44 0044 1
45 0045 1
46 0046 1 --
47 0047 1
48 0048 1 ! Include files:
49 0049 1
50 0050 1 MACRO $module_name string = 'exch$util' %;          ! The require file needs to know our module name
51 0051 1 REQUIRE 'SRC$EXCREQ'
52 0052 1

```

```
54 0149 1 %SBTTL 'Module table of contents'
55 0150 1
56 0151 1 ! Module table of contents:
57 0152 1 !
58 0153 1 FORWARD ROUTINE
59 0154 1     exch$util_block_check : jsb_r0r1r2 NOVALUE, ! Check the block type and size fields
60 0155 1     exch$util_dos11ctx_allocate, ! Allocate a DOS-11 file context block
61 0156 1     exch$util_dos11ctx_release : NOVALUE, ! Release it
62 0157 1     exch$util_fao_buffer, ! Pass arguments through FAO service
63 0158 1     exch$util_filb_allocate, ! Allocate a file block
64 0159 1     exch$util_filb_release : NOVALUE, ! Release a file block
65 0160 1     exch$util_file_error, ! Signal an RMS error
66 0161 1     exch$util_find_mounted_volb, ! Locate a mounted volume block in the volb in-use queue
67 0162 1     exch$util_namb_allocate, ! Allocate a name block
68 0163 1     exch$util_namb_release : NOVALUE, ! Release a name block
69 0164 1     exch$util_radix50_from_ascii, ! Convert an ascii string to radix50
70 0165 1     exch$util_radix50_to_ascii, ! Convert a radix50 string to ascii
71 0166 1     exch$util_rmsb_allocate, ! Allocate a file information block
72 0167 1     exch$util_rmsb_release : NOVALUE, ! Release a file information block
73 0168 1     exch$util_rt11ctx_allocate, ! Allocate an RT-11 file context block
74 0169 1     exch$util_rt11ctx_release : NOVALUE, ! Release it
75 0170 1     exch$util_vm_allocate, ! Call LIB$GET_VM and signal errors
76 0171 1     exch$util_vm_allocate_zeroed, ! Call LIB$GET_VM, clear memory and signal errors
77 0172 1     exch$util_vm_release : NOVALUE, ! Call LIB$FREE_VM and signal errors
78 0173 1     exch$util_vol_getdvi, ! Fill in the device characteristics fields in a volb
79 0174 1     exch$util_volb_allocate, ! Allocate a volume block
80 0175 1     exch$util_volb_release : NOVALUE, ! Release a volume block
81 0176 1     exch$util_up_case : NOVALUE jsb_r1r2r3 ! Convert string to uppercase
82 0177 1
83 0178 1
84 0179 1 ! EXCHANGE facility routines
85 0180 1
86 0181 1 ! EXTERNAL ROUTINE
87 0182 1 !
88 0183 1
89 0184 1 ! Equated symbols:
90 0185 1 !
91 0186 1 ! LITERAL
92 0187 1 !
93 0188 1
94 0189 1 ! Bound declarations:
95 0190 1 !
96 0191 1 ! BIND
97 0192 1 !
```

```

: 99 0193 1 GLOBAL ROUTINE exch$util_block_check (addr : $ref_bblock, code, %SBTTL 'exch$util_block_check'
: 100 0194 1 size_type : VECTOR [2, WORD]) : jsb_r0r1r2 NOVALUE -
: 101 0195 2 BEGIN
: 102 0196 2 !++
: 103 0197 2
: 104 0198 2 : FUNCTIONAL DESCRIPTION:
: 105 0199 2
: 106 0200 2 This routine checks a data structure for correct size and type fields
: 107 0201 2
: 108 0202 2 : INPUTS:
: 109 0203 2
: 110 0204 2 addr - address of the block
: 111 0205 2 code - error code to display if the block doesn't pass
: 112 0206 2 size_type - size and type values, size is in high word, type in low word
: 113 0207 2
: 114 0208 2 : IMPLICIT INPUTS:
: 115 0209 2
: 116 0210 2 none
: 117 0211 2
: 118 0212 2 : OUTPUTS:
: 119 0213 2
: 120 0214 2 none
: 121 0215 2
: 122 0216 2 : IMPLICIT OUTPUTS:
: 123 0217 2
: 124 0218 2 none
: 125 0219 2
: 126 0220 2 : ROUTINE VALUE:
: 127 0221 2
: 128 0222 2 none
: 129 0223 2
: 130 0224 2 : SIDE EFFECTS:
: 131 0225 2
: 132 0226 2 -- If the block does not pass, the image is terminated
: 133 0227 2
: 134 0228 2
: 135 0229 2 BIND
: 136 0230 2 size = size_type [1] : WORD,
: 137 0231 2 type = size_type [0] : WORD;
: 138 0232 2
: 139 0233 2 IF .addr EQL 0 ! Add 1000 to the error code if the block address is zero, this lets
: 140 0234 2 THEN ! us distinguish missing from bad blocks without defining additional error codes
: 141 0235 2 $exch_signal_stop (exch$_blockcheck0, 1, (1000+.code));
: 142 0236 2
: 143 0237 2 IF .addr [excg$_size] NEQ .size
: 144 0238 2 OR
: 145 0239 2 .addr [excg$_type] NEQ .type
: 146 0240 2 THEN
: 147 P 0241 2 $exch_signal_stop (exch$_blockcheck, 6, .code, .addr,
: 148 0242 2 .addr [excg$_size], .size, .addr [excg$_type], .type);
: 149 0243 2
: 150 0244 2 RETURN;
: 151 0245 1 END;
```

```
.TITLE EXCH$UTIL Facility-wide misc routines
.IDENT \V04-000\
```

.EXTN EXCH\$ BLOCKCHECK0
.EXTRN LIB\$STOP, EXCH\$ BLOCKCHECK
.PSECT EXCH\$UTIL_CODE, NOWRT, 2

			52	DD	00000	EXCH\$UTIL_BLOCK_CHECK::				
						PUSHL	R2			: 0193
			50	D5	00002	TSTL	ADDR			: 0233
			15	12	00004	BNEQ	1\$			
		03E8	C1	9F	00006	PUSHAB	1000(CODE)			: 0235
			01	DD	0000A	PUSHL	#1			
00000000G	00	00000000G	8F	DD	0000C	PUSHL	#EXCH\$ BLOCKCHECK0			
			03	FB	00012	CALLS	#3, LIB\$STOP			
			32	11	00019	BRB	3\$			
	02	AE	08	A0	B1 0001B	1\$:	CMPW	8(ADDR), SIZE		: 0237
			09	12	00020	BNEQ	2\$			
			52	0A	A0 9A 00022	MOVZBL	10(ADDR), R2			: 0239
			6E	52	B1 00026	CMPW	R2, TYPE			
			22	13	00029	BEQL	3\$			
			7E	6E	3C 0002B	2\$:	MOVZWL	TYPE, -(SP)		: 0242
			7E	A0	9A 0002E	MOVZBL	10(ADDR), -(SP)			
			7E	0A	AE 3C 00032	MOVZWL	SIZE, -(SP)			
			7E	08	A0 3C 00036	MOVZWL	8(ADDR), -(SP)			
			50	DD	0003A	PUSHL	ADDR			
			51	DD	0003C	PUSHL	CODE			
			06	DD	0003E	PUSHL	#6			
00000000G	00	00000000G	8F	DD	00040	PUSHL	#EXCH\$ BLOCKCHECK			
			08	FB	00046	CALLS	#8, LIB\$STOP			
			04	C0	0004D	3\$:	ADDL2	#4, SP		: 0245
			05	00	00050	RSB				

: Routine Size: 81 bytes, Routine Base: EXCH\$UTIL_CODE + 0000

```
153 0246 1 GLOBAL ROUTINE exch$util_dos11ctx_allocate (volb, filb) = %SBTTL 'exch$util_dos11ctx_allocate (volb, f
154 0247 2 BEGIN
155 0248 2 ++
156 0249 2
157 0250 2 FUNCTIONAL DESCRIPTION:
158 0251 2
159 0252 2 This routine allocates one DOS-11 file context block. If one is available, it is moved from the ava
160 0253 2 queue to the in-use queue. If none are available, then a fresh block is created and placed on the i
161 0254 2 queue.
162 0255 2
163 0256 2 INPUTS:
164 0257 2
165 0258 2 volb - pointer to the associated volb
166 0259 2 filb - pointer to the associated filb
167 0260 2
168 0261 2 IMPLICIT INPUTS:
169 0262 2
170 0263 2 exch$a_gbl [excg$q_dos11ctx_all] - List of allocated file blocks
171 0264 2 exch$a_gbl [excg$q_dos11ctx_avl] - queue of available file blocks
172 0265 2 exch$a_gbl [excg$q_dos11ctx_use] - queue of file blocks in use
173 0266 2
174 0267 2 OUTPUTS:
175 0268 2
176 0269 2 none
177 0270 2
178 0271 2 IMPLICIT OUTPUTS:
179 0272 2
180 0273 2 none
181 0274 2
182 0275 2 ROUTINE VALUE:
183 0276 2
184 0277 2 address of the allocated file block
185 0278 2
186 0279 2 SIDE EFFECTS:
187 0280 2
188 0281 2 All errors are fatal
189 0282 2 --
190 0283 2
191 0284 2 LOCAL
192 0285 2 offset, ! Local temporary
193 0286 2 ptr : $ref_block, ! A local pointer to the dos11ctx
194 0287 2 status
195 0288 2 :
196 0289 2
197 0290 2
198 0291 2 ! First, try to find one in the available queue
199 0292 2
200 0293 2 ptr = $queue_remove_head (exch$a_gbl [excg$q_dos11ctx_avl]);
201 0294 2
202 0295 2 ! If we didn't find one, then it will have to be created
203 0296 2
204 0297 2 IF .ptr EQL 0
205 0298 2 THEN
206 0299 2 BEGIN
207 0300 2
208 0301 2 ! Allocate a fresh dos11ctx from virtual memory. The entire block has been cleared to nulls
209 0302 2 !
```

```

210 0303 ptr = exch$util_vm_allocate_zeroed (exchblk$s_dos11ctx);
211 0304
212 0305 ! Place the dos11ctx at the head of the list of allocated blocks
213 0306 !
214 0307 ptr [dos11ctx$a_alloc] = .exch$a_gbl [excg$a_dos11ctx_alloc];
215 0308 exch$a_gbl [excg$a_dos11ctx_alloc] = .ptr;
216 0309
217 0310 ! Set the block identification fields
218 0311 !
219 0312 $block_init (.ptr, dos11ctx);
220 0313
221 0314 END;
222 0315
223 0316 ! Check our block type, fatal error if any problems
224 0317 !
225 0318 $block_check (2, .ptr, dos11ctx, 578);
226 0319
227 0320 ! Set the last part of the block to nulls
228 0321 !
229 0322 CH$FILL (0, dos11ctx$k_end_zero - dos11ctx$k_start_zero, .ptr + dos11ctx$k_start_zero);
230 0323
231 0324 ! Insert the block at the head of the in-use queue
232 0325 !
233 0326 $queue_insert_head (ptr [dos11ctx$q_header], exch$a_gbl [excg$q_dos11ctx_use]);
234 0327
235 0328 ! Set the two associated fields
236 0329 !
237 0330 ptr [dos11ctx$a_assoc_volb] = .volb;
238 0331 ptr [dos11ctx$a_assoc_filb] = .filb;
239 0332
240 0333 ! Return the address of the file block to the caller
241 0334 !
242 0335 RETURN .ptr;
243 0336
244 0337 1 END;

```

					.EXTRN	EXCH\$A_GBL	
			00FC 0000		.ENTRY	EXCH\$UTIL_DOS11CTX_ALLOCATE, Save R2,R3,R4,-;	0246
		57	00000000G	EF 9E 00002	MOVAB	R5,R6,R7	
51		67	00000064	8F C1 00009	ADDL3	EXCH\$A_GBL, R7	0293
		50	00	B1 0F 00011	REMQUE	#100, EXCH\$A_GBL, R1	
				04 1C 00015		@0(R1), -T-	
				56 D4 00017	BVC	1\$	
				03 11 00019	CLRL	PTR	
		56		50 D0 0001B	BRB	2\$	
				21 12 0001E	MOVL	T-, PTR	
				8F 9A 00020	BNEQ	3\$	0297
		7E	8A	01 FB 00024	MOVZBL	#138, -(SP)	0303
	0000V	CF		50 D0 00029	CALLS	#1, EXCH\$UTIL_VM_ALLOCATE_ZEROED	
		56		67 D0 0002C	MOVL	R0, PTR	
		50		AC D0 0002F	MOVL	EXCH\$A_GBL, R0	0307
	0C	A6	58	56 D0 00034	MOVL	88(R0) - 12(PTR)	0308
	58	A0		8F 9B 00038	MOVL	PTR, 88(R0)	0312
	08	A6	8A		MOVZBW	#138, 8(PTR)	

EXCH\$UTIL
V04-000

Facility-wide misc routines
exch\$util_dos11ctx_allocate (volb, filb)

F 12
16-Sep-1984 01:25:39
14-Sep-1984 12:29:09

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCUTIL.B32;1

Page 7
(4)

EX
VO

0A	A6	04	0003D	MNEGB	#4, 10(PTR)	:		
	52 008A00FC	8F	D0 00041	3\$:	MOVL	#9044220, R2	: 0318	
	51 0242	8F	3C 00048		MOVZWL	#578, R1	:	
	50	56	D0 0004D		MOVL	PTR, R0	:	
006E	8F	00	0000000G		JSB	EXCH\$UTIL_BLOCK_CHECK	:	
		6E	00	2C 00056	MOVCS	#0, (SP), #0, #T10, 28(PTR)	: 0322	
			1C	A6			:	
	50	67	0000005C	8F	C1 0005F	ADDL3	#92, EXCH\$A_GBL, R0	: 0326
		60		66	0E 00067	INSQUE	(PTR), (R0)	:
14	A6	04	AC	D0 0006A	MOVL	VOLB, 20(PTR)	: 0330	
1C	A6	08	AC	D0 0006F	MOVL	FILB, 16(PTR)	: 0331	
	50	56	D0 00074	MOVL	PTR, R0	:	: 0335	
		04	00077	RET		:	: 0337	

: Routine Size: 120 bytes, Routine Base: EXCH\$UTIL_CODE + 0051

```
246 0338 1 GLOBAL ROUTINE exch$util_dos11ctx_release (addr) : NOVALUE = %SBTTL 'exch$util_dos11ctx_release (addr)'  
247 0339 BEGIN  
248 0340 ++  
249 0341 ~~~~~  
250 0342 FUNCTIONAL DESCRIPTION:  
251 0343 ~~~~~  
252 0344 This routine deallocates one dos11ctx. The block is moved from the in-use queue to the available qu  
253 0345 ~~~~~  
254 0346 INPUTS:  
255 0347 ~~~~~  
256 0348 addr - address of the block to release  
257 0349 ~~~~~  
258 0350 IMPLICIT INPUTS:  
259 0351 ~~~~~  
260 0352 exch$a_gbl [excg$q_dos11ctx_avl] - queue of available file blocks  
261 0353 exch$a_gbl [excg$q_dos11ctx_use] - queue of file blocks in use  
262 0354 ~~~~~  
263 0355 OUTPUTS:  
264 0356 ~~~~~  
265 0357 none  
266 0358 ~~~~~  
267 0359 IMPLICIT OUTPUTS:  
268 0360 ~~~~~  
269 0361 none  
270 0362 ~~~~~  
271 0363 ROUTINE VALUE:  
272 0364 ~~~~~  
273 0365 none  
274 0366 ~~~~~  
275 0367 SIDE EFFECTS:  
276 0368 ~~~~~  
277 0369 All errors are fatal  
278 0370 --  
279 0371 ~~~~~  
280 0372 LOCAL  
281 0373 ptr : $ref_block, ! A local pointer to the dos11ctx  
282 0374 status  
283 0375 ;  
284 0376 ~~~~~  
285 0377 ! First, move the pointer to a local variable  
286 0378 !  
287 0379 ptr = .addr;  
288 0380 !  
289 0381 ! Check our block type, fatal error if any problems  
290 0382 !  
291 0383 $block_check (2, .ptr, dos11ctx, 579);  
292 0384 !  
293 0385 ! If there is a buffer allocated, free it  
294 0386 !  
295 0387 !  
296 0388 IF .ptr [dos11ctx$a_buffer] NEQ 0  
297 0389 THEN  
298 0390 exch$util_vm_release (ctx$k_buffer_length, .ptr [dos11ctx$a_buffer]);  
299 0391 !  
300 0392 ! Clear the pointers in the part of the block before the automatic zero  
301 0393 !  
302 0394 ptr [dos11ctx$a_assoc_filb] = 0;
```

```

: 303 0395 2 ptr [dos11ctx$a_assoc_volb] = 0;
: 304 0396 2 ptr [dos11ctx$a_buffer] = 0;
: 305 0397 2
: 306 0398 2 ! Remove the dos11ctx from where ever it is in the in-use queue
: 307 0399 2
: 308 0400 2 $queue_remove (ptr [dos11ctx$q_header]);
: 309 0401 2
: 310 0402 2 ! Place the dos11ctx at the end of the available queue and the head of the in-use queue
: 311 0403 2
: 312 0404 2 $queue_insert_tail (ptr [dos11ctx$q_header], exch$a_gbl [excg$q_dos11ctx_avl]);
: 313 0405 2
: 314 0406 2 RETURN;
: 315 0407 1 END;

```

			000C 00000	.ENTRY	EXCH\$UTIL_DOS11CTX_RELEASE, Save R2,R3	: 0338
	53	04	AC D0 00002	MOVL	ADDR, PTR	: 0380
	52	008A00FC	8F D0 00006	MOVL	#9044220, R2	: 0384
	51	0243	8F 3C 0000D	MOVZWL	#579, R1	
	50		53 D0 00012	MOVL	PTR, R0	
		00000000G	EF 16 00015	JSB	EXCH\$UTIL_BLOCK_CHECK	
		18	A3 D5 0001B	TSTL	24(PTR)	: 0388
			0D 13 0001E	BEQL	1\$	
		18	A3 DD 00020	PUSHL	24(PTR)	: 0390
	7E	1800	8F 3C 00023	MOVZWL	#6144, -(SP)	
	0000V	CF	02 FB 00028	CALLS	#2, EXCH\$UTIL_VM_RELEASE	
		10	A3 7C 0002D 1\$:	CLRQ	16(PTR)	: 0394
		18	A3 D4 00030	CLRL	24(PTR)	: 0396
	50		63 0F 00033	REMQUE	(PTR), T	: 0400
	50 00000000G	EF 00000064	8F C1 00036	ADDL3	#100, EXCH\$a_GBL, R0	: 0404
	04	B0	63 0E 00042	INSQUE	(PTR), @4(R0)	: 0404
			04 00046	RET		: 0407

: Routine Size: 71 bytes, Routine Base: EXCH\$UTIL_CODE + 00C9

```

317 0408 1 GLOBAL ROUTINE exch$util_fao_buffer (ctrstr : REF VECTOR[2], args : VECTOR [4]) = %SBTTL 'exch$util_fa
318 0409 2 BEGIN
319 0410 2 ++
320 0411 2
321 0412 2 FUNCTIONAL DESCRIPTION:
322 0413 2
323 0414 2 This routine passes an ascii string through the FAO system service with any number of specified para
324 0415 2
325 0416 2 INPUTS:
326 0417 2
327 0418 2 ctrstr Address of FAO control string descriptor
328 0419 2 args Any number of additional arguments
329 0420 2
330 0421 2 IMPLICIT INPUTS:
331 0422 2
332 0423 2 none
333 0424 2
334 0425 2 OUTPUTS:
335 0426 2
336 0427 2 none
337 0428 2
338 0429 2 IMPLICIT OUTPUTS:
339 0430 2
340 0431 2 none
341 0432 2
342 0433 2 ROUTINE VALUE:
343 0434 2
344 0435 2 Address of formatted descriptor
345 0436 2
346 0437 2 SIDE EFFECTS:
347 0438 2
348 0439 2 none
349 0440 2 --
350 0441 2
351 0442 2 BIND
352 0443 2 desc = exch$a_gbl [excg$t_fao_buffer] : VECTOR [3]
353 0444 2 ;
354 0445 2
355 0446 2
356 0447 2 desc [0] = excg$s_fao_buffer-8; ! Set up result descriptor
357 0448 2 desc [1] = desc [2];
358 0449 2
359 0450 2 $faol (ctrstr=.ctrstr, outlen=desc, outbuf=desc, prmlst=args);
360 0451 2
361 0452 2 RETURN desc;
362 0453 2 1 END;

```

```

                                .EXTRN  SYSSFAOL
                                .ENTRY  EXCH$UTIL FAO_BUFFER, Save R2
04 00000000G EF 000000E4 8F C1 00002 ADDL3 #228, EXCH$a_GBL, R2
                                62 FA 8F 9A 0000E MOVZBL #250, (R2)
                                08 A2 9E 00012 MOVAB 8(R2), 4(R2)
                                08 AC 9F 00017 PUSHAB ARGS
                                52 DD 0001A PUSHL R2

```

: 0408
: 0443
: 0447
: 0448
: 0450
:

EXCH\$UTIL
V04-000

Facility-wide misc routines
exch\$util_fao_buffer

J 12
16-Sep-1984 01:25:39
14-Sep-1984 12:29:09

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCUTIL.B32;1

Page 11
(6)

00000000G 00
50

04

52 DD 0001C
AC DD 0001E
04 FB 00021
52 D0 00028
04 0002B

PUSHL R2
PUSHL CTRSTR
CALLS #4, SYSS\$FAOL
MOVL R2, R0
RET

:
:
:
: 0452
: 0453

; Routine Size: 44 bytes, Routine Base: EXCH\$UTIL_CODE + 0110

EX
VC

```

: 364 0454 1 GLOBAL ROUTINE exch$util_filb_allocate = %SBTTL 'exch$util_filb_allocate'
: 365 0455 BEGIN
: 366 0456 ++
: 367 0457
: 368 0458 FUNCTIONAL DESCRIPTION:
: 369 0459
: 370 0460 This routine allocates one $FILB. If $FILBs are available, one is moved from the available queue to
: 371 0461 in-use queue. If none are available, then a fresh $FILB is created and placed on the in-use queue.
: 372 0462
: 373 0463 INPUTS:
: 374 0464
: 375 0465 none
: 376 0466
: 377 0467 IMPLICIT INPUTS:
: 378 0468
: 379 0469 exch$a_gbl [excg$a_filb_all] - list of allocated file blocks
: 380 0470 exch$a_gbl [excg$a_filb_avl] - queue of available file blocks
: 381 0471 exch$a_gbl [excg$a_filb_use] - queue of file blocks in use
: 382 0472
: 383 0473 OUTPUTS:
: 384 0474
: 385 0475 none
: 386 0476
: 387 0477 IMPLICIT OUTPUTS:
: 388 0478
: 389 0479 none
: 390 0480
: 391 0481 ROUTINE VALUE:
: 392 0482
: 393 0483 address of the allocated file block
: 394 0484
: 395 0485 SIDE EFFECTS:
: 396 0486
: 397 0487 All errors are fatal
: 398 0488 --
: 399 0489
: 400 0490 LOCAL
: 401 0491 ptr : $ref_bblock, ! A local pointer to the filb
: 402 0492 status
: 403 0493 :
: 404 0494
: 405 0495
: 406 0496 ! First, try to find one in the available queue
: 407 0497
: 408 0498 ptr = $queue_remove_head (exch$a_gbl [excg$a_filb_avl]);
: 409 0499
: 410 0500 ! If we didn't find one, then it will have to be created
: 411 0501
: 412 0502 IF .ptr EQL 0
: 413 0503 THEN
: 414 0504 BEGIN
: 415 0505
: 416 0506 ! Allocate a fresh filb from virtual memory.
: 417 0507
: 418 0508 ptr = exch$util_vm_allocate (exchblk$s_filb);
: 419 0509
: 420 0510 ! Place the filb at the head of the list of allocated blocks
```

```

421 0511 3      !
422 0512 3      ptr [filb$a_alloc] = .exch$a_gbl [excg$a_filb_alloc];
423 0513 3      exch$a_gbl [excg$a_filb_alloc] = .ptr;
424 0514 3
425 0515 3      ! Init the dynamic strings
426 0516 3
427 0517 3      $dyn_str_desc_init (ptr [filb$q_name_string]);
428 0518 3
429 0519 3      ! Set the block identification fields
430 0520 3
431 0521 3      $block_init (.ptr, filb);
432 0522 3
433 0523 3      END;
434 0524 3
435 0525 3      ! Check our block type, fatal error if any problems
436 0526 3
437 0527 3      $block_check (2, .ptr, filb, 481);
438 0528 3
439 0529 3      ! Place the filb at the head of the in-use queue
440 0530 3
441 0531 3      $queue_insert_head (ptr [filb$q_header], exch$a_gbl [excg$q_filb_use]);
442 0532 3
443 0533 3      ! Set the last part of the block to nulls
444 0534 3
445 0535 3      CH$FILL (0, filb$k_end_zero - filb$k_start_zero, .ptr + filb$k_start_zero);
446 0536 3
447 0537 3      ! Return the address of the file block to the caller
448 0538 3
449 0539 3      RETURN .ptr;
450 0540 3
451 0541 3      END;

```

					.EXTRN	EXCH\$GQ_DYN_STR_TEMPLATE	
			00FC 00000		.ENTRY	EXCH\$UTIL_FILB_ALLOCATE, Save R2,R3,R4,R5,-	0454
						R6,R7	
51	57	00000000G	EF 9E 00002		MOVAB	EXCH\$a_GBL, R7	
	67	00000078	8F C1 00009		ADDL3	#120, EXCH\$a_GBL, R1	0498
	50	00	B1 0F 00011		REMQUE	@0(R1), _T_	
			04 1C 00015		BVC	1\$	
			56 D4 00017		CLRL	PTR	
			03 11 00019		BRB	2\$	
	56		50 D0 0001B	1\$:	MOVL	_T_, PTR	
			2E 12 0001E	2\$:	BNEQ	3\$	0502
	7E	035B	8F 3C 00020		MOVZWL	#859, -(SP)	0508
0000V	CF		01 FB 00025		CALLS	#1, EXCH\$UTIL_VM_ALLOCATE	
	56		50 D0 0002A		MOVL	R0, PTR	
	50		67 D0 0002D		MOVL	EXCH\$a_GBL, R0	0512
0C	A6	6C	A0 D0 00030		MOVL	108(R0), 12(PTR)	
6C	A0		56 D0 00035		MOVL	PTR, 108(R0)	0513
	50	10	A6 9E 00039		MOVAB	16(PTR), R0	0517
	60	00000000G	EF 7D 0003D		MOVQ	TMPL, (R0)	
08	A6	035B	8F B0 00044		MOVW	#859, 8(PTR)	0521
0A	A6		06 8E 0004A		MNEGB	#6, 10(PTR)	
	52	035B00FA	8F D0 0004E	3\$:	MOVL	#56295674, R2	0527

EXCH\$UTIL
V04-000

Facility-wide misc routines
exch\$util_filb_allocate

M 12
16-Sep-1984 01:25:39
14-Sep-1984 12:29:09

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCUTIL.B32;1

Page 14
(7)

EX
VO

			51	01E1	8F	3C	00055	MOVZWL	#481, R1	:
			50		56	D0	0005A	MOVL	PTR, R0	:
				00000000G	EF	16	0005D	JSB	EXCH\$UTIL BLOCK CHECK	:
		50	67	00000070	8F	C1	00063	ADDL3	#112, EXCHSA_GBL, R0	: 0531
			60		66	0E	0006B	INSQUE	(PTR), (R0)	:
0042	8F	00	6E		00	2C	0006E	MOVCS	#0, (SP), #0, #66, 24(PTR)	: 0535
				18	A6		00075			:
			50		56	D0	00077	MOVL	PTR, R0	: 0539
					04	0007A	RET			: 0541

: Routine Size: 123 bytes, Routine Base: EXCH\$UTIL_CODE + 013C


```
453 0542 1 GLOBAL ROUTINE exch$util_filb_release (addr) : NOVALUE = %SBTTL 'exch$util_filb_release (addr)'  
454 0543 2 BEGIN  
455 0544 2 ++  
456 0545 2  
457 0546 2 FUNCTIONAL DESCRIPTION:  
458 0547 2  
459 0548 2 This routine deallocates one $FILB. The $FILB is moved from the in-use queue to the available queue  
460 0549 2  
461 0550 2 INPUTS:  
462 0551 2  
463 0552 2 addr - address of the block to release  
464 0553 2  
465 0554 2 IMPLICIT INPUTS:  
466 0555 2  
467 0556 2 exch$a_gbl [excg$q_filb_avl] - queue of available file blocks  
468 0557 2 exch$a_gbl [excg$q_filb_use] - queue of file blocks in use  
469 0558 2  
470 0559 2 OUTPUTS:  
471 0560 2  
472 0561 2 none  
473 0562 2  
474 0563 2 IMPLICIT OUTPUTS:  
475 0564 2  
476 0565 2 none  
477 0566 2  
478 0567 2 ROUTINE VALUE:  
479 0568 2  
480 0569 2 none  
481 0570 2  
482 0571 2 SIDE EFFECTS:  
483 0572 2  
484 0573 2 All errors are fatal  
485 0574 2 --  
486 0575 2  
487 0576 2 LOCAL  
488 0577 2 ptr : $ref_block, ! A local pointer to the filb  
489 0578 2 status  
490 0579 2 ;  
491 0580 2  
492 0581 2  
493 0582 2 ! First, move the pointer to a local variable  
494 0583 2  
495 0584 2 ptr = .addr;  
496 0585 2  
497 0586 2 ! Check our block type, fatal error if any problems  
498 0587 2  
499 0588 2 $block_check (2, .ptr, filb, 482);  
500 0589 2  
501 0590 2 ! Remove the filb from where ever it is in the in-use queue  
502 0591 2  
503 0592 2 $queue_remove (ptr [filb$q_header]);  
504 0593 2  
505 0594 2 ! Place the filb at the end of the available queue.  
506 0595 2  
507 0596 2 $queue_insert_tail (ptr [filb$q_header], exch$a_gbl [excg$q_filb_avl]);  
508 0597 2  
509 0598 2 RETJRN;
```

; 510

0599 1 END;

			000C 00000	.ENTRY	EXCH\$UTIL_FILB_RELEASE, Save R2,R3	: 0542
	53	04	AC D0 00002	MOVL	ADDR, PTR	: 0584
	52	035B00FA	8F D0 00006	MOVL	#56295674, R2	: 0588
	51	01E2	8F 3C 0000D	MOVZWL	#482, R1	
	50		53 D0 00012	MOVL	PTR, R0	
		00000000G	EF 16 00015	JSB	EXCH\$UTIL_BLOCK_CHECK	
	50		63 0F 0001B	REMQUE	(PTR), T	: 0592
50	00000000G	EF 00000078	8F C1 0001E	ADDL3	#120, EXCH\$A_GBL, R0	: 0596
	04	B0	63 0E 0002A	INSQUE	(PTR), @4(R0)	
			04 0002E	RET		: 0599

; Routine Size: 47 bytes, Routine Base: EXCH\$UTIL_CODE + 01B7

```

: 512 0600 1 GLOBAL ROUTINE exch$util_file_error (msg, rms_status, fabb : $ref_bblock, stv) = %SBTTL 'exch$util_fi
: 513 0601 2 BEGIN
: 514 0602 2 ++
: 515 0603 2
: 516 0604 2 FUNCTIONAL DESCRIPTION:
: 517 0605 2
: 518 0606 2 This routine signals an RMS error. The appropriate file name for the signal is found by
: 519 0607 2 examining the contents of the nam block.
: 520 0608 2
: 521 0609 2 INPUTS:
: 522 0610 2
: 523 0611 2 msg Error message value, assumed to have one !AS FAO argument
: 524 0612 2 rms_status Error message from RMS call
: 525 0613 2 fab$ Pointer to FAB, used to locate nam block
: 526 0614 2 stv The RMS STV error from the FAB or RAB
: 527 0615 2
: 528 0616 2 IMPLICIT INPUTS:
: 529 0617 2
: 530 0618 2 RMS nam block attached to the FAB (fabb)
: 531 0619 2
: 532 0620 2 OUTPUTS:
: 533 0621 2
: 534 0622 2 none
: 535 0623 2
: 536 0624 2 IMPLICIT OUTPUTS:
: 537 0625 2
: 538 0626 2 none
: 539 0627 2
: 540 0628 2 ROUTINE VALUE:
: 541 0629 2
: 542 0630 2 msg - with inhibit signal bit set
: 543 0631 2
: 544 0632 2 SIDE EFFECTS:
: 545 0633 2
: 546 0634 2 An error will be signalled
: 547 0635 2 --
: 548 0636 2
: 549 0637 2 LOCAL
: 550 0638 2 tmp_desc : $desc block, ! A descriptor for the file name
: 551 0639 2 nam_blk : $ref_bblock; ! Pointer to the name block
: 552 0640 2
: 553 0641 2 nam_blk = .fabb [fab$l_nam]; ! Get pointer to the name block
: 554 0642 2 tmp_desc [dsc$b_class] = dsc$k_class_s; ! Static desc
: 555 0643 2 tmp_desc [dsc$b_dtype] = dsc$k_dtype_t; ! String desc
: 556 0644 2
: 557 0645 2 IF .nam_blk [nam$b_rsl] GTRU 0
: 558 0646 2 THEN
: 559 0647 2 BEGIN
: 560 0648 2 tmp_desc [dsc$w_length] = .nam_blk [nam$b_rsl]; ! Create file name desc
: 561 0649 2 tmp_desc [dsc$a_pointer] = .nam_blk [nam$l_rsa]; ! ...
: 562 0650 2 END
: 563 0651 2 ELSE IF .nam_blk [nam$b_esl] GTRU 0
: 564 0652 2 THEN
: 565 0653 2 BEGIN
: 566 0654 2 tmp_desc [dsc$w_length] = .nam_blk [nam$b_esl]; ! Create file name desc
: 567 0655 2 tmp_desc [dsc$a_pointer] = .nam_blk [nam$l_esa]; ! ...
: 568 0656 2 END
```

```

: 569      0657 2 ELSE
: 570      0658 3     BEGIN
: 571      0659 3         tmp_desc [dsc$w_length] = .fabb [fab$b_fns];      ! Create file name desc
: 572      0660 3         tmp_desc [dsc$a_pointer] = .fabb [fab$a_fna];    ! ...
: 573      0661 2     END;
: 574      0662 2
: 575      0663 2 SIGNAL (.msg, 1, tmp_desc, .rms_status, .stv);
: 576      0664 2
: 577      0665 2 RETURN .msg;
: 578      0666 2
: 579      0667 1 END;

```

```

                                0000 00000      .ENTRY EXCH$UTIL_FILE_ERROR, Save nothing      : 0600
                                08 C2 00002      SUBL2 #8, SP
                                5E 0C AC D0 00005      MOVL FABB, R1      : 0641
                                51 28 A1 D0 00009      MOVL 40(R1), NAM_BLK
                                50 AE 010E 8F B0 0000D      MOVW #270, TMP_DESC+2      : 0643
                                02 AE 03 A0 95 00013      TSTB 3(NAM_BLK)      : 0645
                                0B 13 00016      BEQL 1$
                                6E 03 A0 9B 00018      MOVZBW 3(NAM_BLK), TMP_DESC      : 0648
                                04 AE 04 A0 D0 0001C      MOVL 4(NAM_BLK), TMP_DESC+4      : 0649
                                19 11 00021      BRB 3$
                                0B A0 95 00023 1$:      TSTB 11(NAM_BLK)      : 0651
                                0B 13 00026      BEQL 2$
                                6E 0B A0 9B 00028      MOVZBW 11(NAM_BLK), TMP_DESC      : 0654
                                04 AE 0C A0 D0 0002C      MOVL 12(NAM_BLK), TMP_DESC+4      : 0655
                                09 11 00031      BRB 3$
                                6E 34 A1 9B 00033 2$:      MOVZBW 52(R1), TMP_DESC      : 0659
                                04 AE 2C A1 D0 00037      MOVL 44(R1), TMP_DESC+4      : 0660
                                10 AC DD 0003C 3$:      PUSHL STV      : 0663
                                08 AC DD 0003F      PUSHL RMS_STATUS
                                08 AE 9F 00042      PUSHAB TMP_DESC
                                01 DD 00045      PUSHL #1
                                04 AC DD 00047      PUSHL MSG
                                00 00 00000GG 00      CALLS #5, LIB$SIGNAL
                                50 04 AC D0 00051      MOVL MSG, R0      : 0665
                                04 00055      RET      : 0667

```

; Routine Size: 86 bytes, Routine Base: EXCH\$UTIL_CODE + 01E6

```
581 0668 1 GLOBAL ROUTINE exch$util_find_mounted_volb (ident : $ref_bvector) = %SBTTL 'exch$util_find_mounted_volb
582 0669 2 BEGIN
583 0670 2 ++
584 0671 2
585 0672 2 FUNCTIONAL DESCRIPTION:
586 0673 2
587 0674 2 This routine scans the queue of in-use volume blocks to see if any have the same name as the
588 0675 2 input name.
589 0676 2
590 0677 2 INPUTS:
591 0678 2
592 0679 2 ident - address of the first byte
593 0680 2
594 0681 2 IMPLICIT INPUTS:
595 0682 2
596 0683 2 none
597 0684 2
598 0685 2 OUTPUTS:
599 0686 2
600 0687 2 none
601 0688 2
602 0689 2 IMPLICIT OUTPUTS:
603 0690 2
604 0691 2 none
605 0692 2
606 0693 2 ROUTINE VALUE:
607 0694 2
608 0695 2 0 if name not found, address of volb if name is found
609 0696 2
610 0697 2 SIDE EFFECTS:
611 0698 2
612 0699 2 none
613 0700 2 --
614 0701 2
615 0702 2 $dbgtrc_prefix ('util_find_mounted_volb> ');
616 0703 2
617 0704 2 LOCAL
618 0705 2 ptr : $ref_bblock, ! Pointer to scan along the queue
619 0706 2 status
620 0707 2 :
621 0708 2
622 0709 2
623 0710 2 ! Get the first volb, and scan the list of file names
624 0711 2
625 0712 2 ptr = .exch$a_gbl [excg$a_volb_use_flink];
626 0713 2
627 0714 2 WHILE .ptr NEQA exch$a_gbl [excg$q_volb_use]
628 0715 2 DO
629 0716 2 BEGIN
630 0717 2
631 0718 2 $block_check (2, .ptr, volb, 483);
632 0719 2
633 0720 2 IF CH$EQL (volb$s_vol_ident, .ident, volb$s_vol_ident, ptr [volb$t_vol_ident])
634 0721 2 THEN
635 0722 2 RETURN .ptr;
636 0723 2
637 0724 2 ptr = .ptr [volb$a_flink]; ! Advance to next volb in the in-use queue
```

```

: 638      0725 3
: 639      0726 2      END;
: 640      0727 2
: 641      0728 2 RETURN 0;
: 642      0729 1 END;

```

```

                                003C 00000      .ENTRY EXCH$UTIL_FIND_MOUNTED_VOLB, Save R2,R3,R4,-; 0668
                                EF 9E 00002      MOVAB EXCH$A_GBL, R5
55 00000000G      65 D0 00009      MOVL EXCH$A_GBL, R0
50                                CO D0 0000C      MOVL 192(R0), PTR      0712
54 00C0          C0 D0 0000C      ADDL3 #192, EXCH$A_GBL, R0
65 000000C0      8F C1 00011 1$:      CMPL PTR, R0      0714
50                                54 D1 00019      BEQL 3$
                                28 13 0001C      MOVL #68878579, R2      0718
52 041B00F3      8F D0 0001E      MOVZWL #483, R1
51 01E3          8F 3C 00025      MOVL PTR, R0
50                                54 D0 0002A      JSB EXCH$UTIL_BLOCK_CHECK
69 A4          04 BC 00000000G      CMPC3 #128, @IDENT, 105(PTR)      0720
                                8F 29 00033      BNEQ 2$
                                04 12 0003B      MOVL PTR, R0      0722
50                                54 D0 0003D      RET
54                                04 00040      MOVL (PTR), PTR      0724
                                CB 11 00044      BRB 1$      0714
                                50 D4 00046 3$:      CLRL R0      0728
                                04 00048      RET      0729

```

; Routine Size: 73 bytes, Routine Base: EXCH\$UTIL_CODE + 023C

```
644 0730 1 GLOBAL ROUTINE exch$util_namb_allocate = %SBTTL 'exch$util_namb_allocate'
645 0731 2 BEGIN
646 0732 2 ++
647 0733 2
648 0734 2 FUNCTIONAL DESCRIPTION:
649 0735 2
650 0736 2 This routine allocates one $NAMB. If $NAMBS are available, one is moved from the available queue to
651 0737 2 in-use queue. If none are available, then a fresh $NAMB is created and placed on the in-use queue.
652 0738 2
653 0739 2 INPUTS:
654 0740 2
655 0741 2 none
656 0742 2
657 0743 2 IMPLICIT INPUTS:
658 0744 2
659 0745 2 exch$a_gbl [excg$q_namb_all] - list of allocated name blocks
660 0746 2 exch$a_gbl [excg$q_namb_avl] - queue of available name blocks
661 0747 2 exch$a_gbl [excg$q_namb_use] - queue of name blocks in use
662 0748 2
663 0749 2 OUTPUTS:
664 0750 2
665 0751 2 none
666 0752 2
667 0753 2 IMPLICIT OUTPUTS:
668 0754 2
669 0755 2 none
670 0756 2
671 0757 2 ROUTINE VALUE:
672 0758 2
673 0759 2 address of the allocated name block
674 0760 2
675 0761 2 SIDE EFFECTS:
676 0762 2
677 0763 2 All errors are fatal
678 0764 2 --
679 0765 2
680 0766 2 LOCAL
681 0767 2 offset, ! Local temporary
682 0768 2 ptr : $ref_bblock, ! A local pointer to the namb
683 0769 2 status
684 0770 2 :
685 0771 2
686 0772 2 ! First, try to find one in the available queue
687 0773 2
688 0774 2
689 0775 2 ptr = $queue_remove_head (exch$a_gbl [excg$q_namb_avl]);
690 0776 2
691 0777 2 ! If we didn't find one, then it will have to be created
692 0778 2
693 0779 2 IF .ptr EQL 0
694 0780 2 THEN
695 0781 2 BEGIN
696 0782 2
697 0783 2 ! Allocate a fresh namb from virtual memory. The entire block has been cleared to nulls
698 0784 2
699 0785 2 ptr = exch$util_vm_allocate_zeroed (exchblk$s_namb);
700 0786 2
```

```

: 701 0787 3 Place the namb at the head of the list of allocated blocks
: 702 0788
: 703 0789 ptr [namb$a_alloc] = .exch$a_gbl [excg$a_namb_alloc];
: 704 0790 exch$a_gbl [excg$a_namb_alloc] = .ptr;
: 705 0791
: 706 0792 ; Set the block identification fields
: 707 0793
: 708 0794 $block_init (.ptr, namb);
: 709 0795
: 710 0796 ; Initialize the dynamic strings
: 711 0797
: 712 0798 $dyn_str_desc_init (ptr [namb$q_input]);
: 713 0799 $dyn_str_desc_init (ptr [namb$q_fullname]);
: 714 0800 $dyn_str_desc_init (ptr [namb$q_expanded]);
: 715 0801 $dyn_str_desc_init (ptr [namb$q_result]);
: 716 0802 $dyn_str_desc_init (ptr [namb$q_device_dvi]);
: 717 0803
: 718 0804 END;
: 719 0805
: 720 0806 ; Check our block type, fatal error if any problems
: 721 0807
: 722 0808 $block_check (2, .ptr, namb, 484);
: 723 0809
: 724 0810 ; Place the namb at the head of the in-use queue
: 725 0811
: 726 0812 $queue_insert_head (ptr [namb$q_header], exch$a_gbl [excg$q_namb_use]);
: 727 0813
: 728 0814 ; Set the last part of the block to nulls
: 729 0815
: 730 0816 CH$FILL (0, exchblk$s_namb - namb$k_start_zero, .ptr + namb$k_start_zero);
: 731 0817
: 732 0818 ; Return the address of the name block to the caller
: 733 0819
: 734 0820 RETURN .ptr;
: 735 0821
: 736 0822 1 END;

```

			00FC 0000		.ENTRY EXCH\$UTIL_NAMB_ALLOCATE, Save R2,R3,R4,R5,-	0730
					R6,R7	
51	57	00000000G	EF 9E 00002		MOVAB EXCH\$a_GBL, R7	
	67	0000008C	8F C1 00009		ADDL3 #140, EXCH\$a_GBL, R1	0775
	50	00	B1 0F 00011		REMQUE @0(R1), _T_	
			04 1C 00015		BVC 1\$	
			56 D4 00017		CLRL PTR	
			03 11 00019		BRB 2\$	
	56		50 D0 0001B	1\$:	MOVL _T_, PTR	
			6A 12 0001E	2\$:	BNEQ 3\$	0779
	7E	010A	8F 3C 00020		MOVZWL #266, -(SP)	0785
	0060V		01 FB 00025		CALLS #1, EXCH\$UTIL_VM_ALLOCATE_ZEROED	
			56 50 D0 0002A		MOVL R0, PTR	
			50 67 D0 0002D		MOVL EXCH\$a_GBL, R0	0789
	0C	A6 0080	C0 D0 00030		MOVL 128(R0), 12(PTR)	
	0080	C0	56 D0 00036		MOVL PTR, 128(R0)	0790

08	A6	010A	8F	B0	0003B	MOVW	#266, 8(PTR)	0794
0A	A6		09	8E	00041	MNEGB	#9, 10(PTR)	0798
	50	10	A6	9E	00045	MOVAB	16(PTR), R0	
	52	00000000G	EF	D0	00049	MOVL	TMP, R2	
	60		52	D0	00050	MOVL	R2, (R0)	
	51	00000000G	EF	D0	00053	MOVL	TMP+4, R1	
04	A0		51	D0	0005A	MOVL	R1, 4(R0)	
	50	18	A6	9E	0005E	MOVAB	24(PTR), R0	0799
	60		52	D0	00062	MOVL	R2, (R0)	
04	A0		51	D0	00065	MOVL	R1, 4(R0)	
	50	20	A6	9E	00069	MOVAB	32(PTR), R0	0800
	60		52	D0	0006D	MOVL	R2, (R0)	
04	A0		51	D0	00070	MOVL	R1, 4(R0)	
	50	28	A6	9E	00074	MOVAB	40(PTR), R0	0801
	60		52	D0	00078	MOVL	R2, (R0)	
04	A0		51	D0	0007B	MOVL	R1, 4(R0)	
	50	30	A6	9E	0007F	MOVAB	48(PTR), R0	0802
	60		52	D0	00083	MOVL	R2, (R0)	
04	A0		51	D0	00086	MOVL	R1, 4(R0)	
	52	010A00F7	8F	D0	0008A	MOVL	#17432823, R2	0808
	51	01E4	8F	3C	00091	MOVZWL	#484, R1	
	50		56	D0	00096	MOVL	PTR, R0	
		00000000G	EF	16	00099	JSB	EXCH\$UTIL BLOCK CHECK	
	67	00000084	8F	C1	0009F	ADDL3	#132, EXCH\$A_GBC, R0	0812
	60		66	0E	000A7	INSQUE	(PTR), (R0)	
00A2	8F	00	00	2C	000AA	MOVCS	%0, (%SP), #0, #162, 104(PTR)	0816
	6E		00	2C	000AA			
		68	A6		000B1			
	50		56	D0	000B3	MOVL	PTR, R0	0820
			04	00	000B6	RET		0822

; Routine Size: 183 bytes, Routine Base: EXCH\$UTIL_CODE + 0285

```

: 738 0823 1 GLOBAL ROUTINE exch$util_namb_release (addr) : NOVALUE = %SBTTL 'exch$util_namb_release (addr)'
: 739 0824 2 BEGIN
: 740 0825 2 ++
: 741 0826 2
: 742 0827 2 FUNCTIONAL DESCRIPTION:
: 743 0828 2
: 744 0829 2 This routine deallocates one $NAMB. The $NAMB is moved from the in-use queue to the available queue
: 745 0830 2
: 746 0831 2 INPUTS:
: 747 0832 2
: 748 0833 2 addr - address of the block to release
: 749 0834 2
: 750 0835 2 IMPLICIT INPUTS:
: 751 0836 2
: 752 0837 2 exch$a_gbl [excg$q_namb_avl] - queue of available name blocks
: 753 0838 2 exch$a_gbl [excg$q_namb_use] - queue of name blocks in use
: 754 0839 2
: 755 0840 2 OUTPUTS:
: 756 0841 2
: 757 0842 2 none
: 758 0843 2
: 759 0844 2 IMPLICIT OUTPUTS:
: 760 0845 2
: 761 0846 2 none
: 762 0847 2
: 763 0848 2 ROUTINE VALUE:
: 764 0849 2
: 765 0850 2 none
: 766 0851 2
: 767 0852 2 SIDE EFFECTS:
: 768 0853 2
: 769 0854 2 All errors are fatal
: 770 0855 2 --
: 771 0856 2
: 772 0857 2 LOCAL
: 773 0858 2 ptr : $ref_bblock, ! A local pointer to the namb
: 774 0859 2 status
: 775 0860 2 ;
: 776 0861 2
: 777 0862 2
: 778 0863 2 ! First, move the pointer to a local variable
: 779 0864 2
: 780 0865 2 ptr = .addr;
: 781 0866 2
: 782 0867 2 ! Check our block type, fatal error if any problems
: 783 0868 2
: 784 0869 2 $block_check (2, .ptr, namb, 485);
: 785 0870 2
: 786 0871 2 ! Remove the namb from where ever it is in the in-use queue
: 787 0872 2
: 788 0873 2 $queue_remove (ptr [namb$q_header]);
: 789 0874 2
: 790 0875 2 ! Place the namb at the end of the available queue.
: 791 0876 2
: 792 0877 2 $queue_insert_tail (ptr [namb$q_header], exch$a_gbl [excg$q_namb_avl]);
: 793 0878 2
: 794 0879 2 RETURN;

```

EXCH\$UTIL
V04-000

Facility-wide misc routines
exch\$util_namb_release (addr)

K 13
16-Sep-1984 01:25:39
14-Sep-1984 12:29:09

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCUTIL.B32;1

Page 25
(12)

EX
VO

: 795

0880 1 END;

			000C	00000
53	04	AC	D0	00002
52	010A00F7	8F	D0	00006
51	01E5	8F	3C	0000D
50		53	D0	00012
	00000000G	EF	16	00015
50		63	0F	0001B
50	00000000G	EF	C1	0001E
04		B0	63	0E 0002A
			04	0002E

.ENTRY	EXCH\$UTIL_NAMB_RELEASE, Save R2,R3
MOVL	ADDR, PTR
MOVL	#17432823, R2
MOVZWL	#485, R1
MOVL	PTR, R0
JSB	EXCH\$UTIL_BLOCK_CHECK
REMQUE	(PTR), T
ADDL3	#140, EXCH\$A_GBL, R0
INSQUE	(PTR), @4(R0)
RET	

:	0823
:	0865
:	0869
:	
:	
:	
:	
:	0873
:	0877
:	
:	0880

: Routine Size: 47 bytes, Routine Base: EXCH\$UTIL_CODE + 033C

```

797 0881 1 GLOBAL ROUTINE exch$util_radix50_from_ascii (asc_cnt, asc, r50_cnt, r50) = %$b:TL 'exch$util_radix50_fr
798 0882 2 BEGIN
799 0883 2 ++
800 0884 2
801 0885 2 FUNCTIONAL DESCRIPTION:
802 0886 2
803 0887 2 This converts ascii strings to Radix-50.
804 0888 2
805 0889 2 INPUTS:
806 0890 2
807 0891 2 asc_cnt - count of ascii characters to output
808 0892 2 asc - address of buffer of ascii characters
809 0893 2 r50_cnt - count of radix-50 characters
810 0894 2
811 0895 2 IMPLICIT INPUTS:
812 0896 2
813 0897 2 none
814 0898 2
815 0899 2 OUTPUTS:
816 0900 2
817 0901 2 r50 - address of Radix-50 string
818 0902 2
819 0903 2 IMPLICIT OUTPUTS:
820 0904 2
821 0905 2 none
822 0906 2
823 0907 2 ROUTINE VALUE:
824 0908 2
825 0909 2 true if conversion went smoothly, false if anything unusual
826 0910 2
827 0911 2 SIDE EFFECTS:
828 0912 2
829 0913 2 none
830 0914 2 --
831 0915 2
832 0916 2 LOCAL
833 0917 2 buf : $bvector [6]
834 0918 2 ;
835 0919 2
836 0920 2 EXTERNAL ROUTINE irad50 : ADDRESSING_MODE (GENERAL); ! F4P compatibility routine
837 0921 2
838 0922 2 $logic_check (2, (.asc_cnt LEQU 6), 165);
839 0923 2 CH$COPY (.asc_cnt, .asc, 32, 6, buf);
840 0924 2
841 0925 2 irad50 (r50_cnt, buf, .r50);
842 0926 2
843 0927 2 RETURN true;
844 0928 2
845 0929 1 END;

```

.EXTRN IRAD50, EXCH\$_BADLOGIC

003C 00000

.ENTRY EXCH\$UTIL_RADIX50_FROM_ASCII, Save R2,R3,- ; 0881

SE

08 C2 00002

SUBL2 #8, SP

:

EXCH\$UTIL
V04-000

Facility-wide misc routines
exch\$util_radix50_from_ascii

(asc_cnt, asc, r50
M 13
16-Sep-1984 01:25:39
14-Sep-1984 12:29:09

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCUTIL.B32;1

Page 27
(13)

EX
VO

		06	04	AC	D1	00005		(MPL	ASC_CNT, #6	:	0922
				13	1B	00009		BLEQU	1\$:	
		7E	A5	8F	9A	0000B		MOVZBL	#165, -(SP)	:	
				01	DD	0000F		PUSHL	#1	:	
			00000000G	8F	DD	00011		PUSHL	#EXCH\$ BADLOGIC	:	
06		00000000G	00	03	FB	00017		CALLS	#3, LIB\$STOP	:	
	20	08	BC	04	AC	2C	1\$:	MOVCS	ASC_CNT, @ASC, #32, #6, BUF	:	0923
				6E		00025				:	
				10	AC	DD		PUSHL	R50	:	0925
				04	AE	9F		PUSHAB	BUF	:	
				0C	AC	9F		PUSHAB	R50_CNT	:	
		00000000G	00	03	FB	0002F		CALLS	#3, -IRAD50	:	
			50	01	DD	00036		MOVL	#1, R0	:	0927
				04		00039		RET		:	0929

; Routine Size: 58 bytes, Routine Base: EXCH\$UTIL_CODE + 036B

```

: 847 0930 1 GLOBAL ROUTINE exch$util_radix50_to_ascii (asc_cnt, r50, asc) = %SBTTL 'exch$util_radix50_to_ascii (asc_cnt,
: 848 0931 2 BEGIN
: 849 0932 2 ++
: 850 0933 2
: 851 0934 2 FUNCTIONAL DESCRIPTION:
: 852 0935 2
: 853 0936 2 This converts Radix-50 strings to ascii.
: 854 0937 2
: 855 0938 2 INPUTS:
: 856 0939 2
: 857 0940 2 asc_cnt - count of ascii characters to output
: 858 0941 2 r50 - address of Radix-50 string. Asc_cnt implies the length of this string.
: 859 0942 2
: 860 0943 2 IMPLICIT INPUTS:
: 861 0944 2
: 862 0945 2 none
: 863 0946 2
: 864 0947 2 OUTPUTS:
: 865 0948 2
: 866 0949 2 asc - address of buffer to receive ascii characters
: 867 0950 2
: 868 0951 2 IMPLICIT OUTPUTS:
: 869 0952 2
: 870 0953 2 none
: 871 0954 2
: 872 0955 2 ROUTINE VALUE:
: 873 0956 2
: 874 0957 2 true if conversion went smoothly, false if anything unusual
: 875 0958 2
: 876 0959 2 SIDE EFFECTS:
: 877 0960 2
: 878 0961 2 none
: 879 0962 2 --
: 880 0963 2
: 881 0964 2
: 882 0965 2 EXTERNAL ROUTINE r50asc : ADDRESSING_MODE (GENERAL); ! F4P compatibility routine
: 883 0966 2 r50asc (asc_cnt, .r50, .asc);
: 884 0967 2
: 885 0968 2 RETURN true;
: 886 0969 2
: 887 0970 1 END;

```

```

                                .EXTRN R50ASC
                                .ENTRY EXCH$UTIL_RADIX50_TO_ASCII, Save nothing : 0930
MOVQ R50, -(SP) : 0966
PUSHAB ASC_CNT
CALLS #3, R50ASC
MOVL #1, R0 : 0968
RET : 0970

```

; Routine Size: 20 bytes, Routine Base: EXCH\$UTIL_CODE + 03A5

```

: 889 0971 1 GLOBAL ROUTINE exch$util_rmsb_allocate = %SBTTL 'exch$util_rmsb_allocate'
: 890 0972 2 BEGIN
: 891 0973 2 ++
: 892 0974 2
: 893 0975 2 FUNCTIONAL DESCRIPTION:
: 894 0976 2
: 895 0977 2 This routine allocates one $RMSB. If $RMSBs are available, one is moved from the available queue to
: 896 0978 2 in-use queue. If none are available, then a fresh $RMSB is created and placed on the in-use queue.
: 897 0979 2
: 898 0980 2 INPUTS:
: 899 0981 2
: 900 0982 2 none
: 901 0983 2
: 902 0984 2 IMPLICIT INPUTS:
: 903 0985 2
: 904 0986 2 exch$a_gbl [excg$q_rmsb_all] - list of allocated file blocks
: 905 0987 2 exch$a_gbl [excg$q_rmsb_avl] - queue of available file blocks
: 906 0988 2 exch$a_gbl [excg$q_rmsb_use] - queue of file blocks in use
: 907 0989 2
: 908 0990 2 OUTPUTS:
: 909 0991 2
: 910 0992 2 none
: 911 0993 2
: 912 0994 2 IMPLICIT OUTPUTS:
: 913 0995 2
: 914 0996 2 none
: 915 0997 2
: 916 0998 2 ROUTINE VALUE:
: 917 0999 2
: 918 1000 2 address of the allocated file block
: 919 1001 2
: 920 1002 2 SIDE EFFECTS:
: 921 1003 2
: 922 1004 2 All errors are fatal
: 923 1005 2 --
: 924 1006 2
: 925 1007 2 LOCAL
: 926 1008 2 offset, ! Local temporary
: 927 1009 2 ptr : $ref_bblock, ! A local pointer to the rmsb
: 928 1010 2 status
: 929 1011 2 :
: 930 1012 2
: 931 1013 2
: 932 1014 2 ! First, try to find one in the available queue
: 933 1015 2
: 934 1016 2 ptr = $queue_remove_head (exch$a_gbl [excg$q_rmsb_avl]);
: 935 1017 2
: 936 1018 2 ! If we didn't find one, then it will have to be created
: 937 1019 2
: 938 1020 2 IF .ptr EQL 0
: 939 1021 2 THEN
: 940 1022 2 BEGIN
: 941 1023 2
: 942 1024 2 ! Allocate a fresh rmsb from virtual memory. The entire block has been cleared to nulls
: 943 1025 2
: 944 1026 2 ptr = exch$util_vm_allocate_zeroed (exchblk$s_rmsb);
: 945 1027 2

```

```

: 946      1028 3      ! Place the rmsb at the head of the list of allocated blocks
: 947      1029      !
: 948      1030      ptr [rmsb$a_alloc] = .exch$a_gbl [excg$a_rmsb_alloc];
: 949      1031      exch$a_gbl [excg$a_rmsb_alloc] = .ptr;
: 950      1032      !
: 951      1033      ! Set the block identification fields
: 952      1034      !
: 953      1035      $block_init (.ptr, rmsb);
: 954      1036      !
: 955      1037      ! Several items are located at the end of the $RMSB, fill in the pointers
: 956      1038      !
: 957      1039      ptr [rmsb$a_fab] = .ptr + rmsb$k_length;          ! Fab is at end of block
: 958      1040      ptr [rmsb$a_rab] = .ptr [rmsb$a_fab] + fab$k_bln;    ! Rab right after Fab
: 959      1041      ptr [rmsb$a_nam] = .ptr [rmsb$a_rab] + rab$k_bln;    ! Nam after Rab
: 960      1042      ptr [rmsb$a_esbuf] = .ptr [rmsb$a_nam] + nam$k_bln;  ! Expanded string after Nam
: 961      1043      ptr [rmsb$a_rsbuf] = .ptr [rmsb$a_esbuf] + nam$c_maxrss; ! Result string after Ebuf
: 962      1044      !
: 963      1045      END;
: 964      1046      !
: 965      1047      ! Check our block type, fatal error if any problems
: 966      1048      !
: 967      1049      $block_check (2, .ptr, rmsb, 407);
: 968      1050      !
: 969      1051      ! Set the last part of the block to nulls
: 970      1052      !
: 971      1053      CH$FILL (0, exchblk$s_rmsb - rmsb$k_start_zero, .ptr + rmsb$k_start_zero);
: 972      1054      !
: 973      1055      ! Insert the block at the head of the in-use queue
: 974      1056      !
: 975      1057      $queue_insert_head (ptr [rmsb$q_header], exch$a_gbl [excg$q_rmsb_use]);
: 976      1058      !
: 977      1059      ! Return the address of the file block to the caller
: 978      1060      !
: 979      1061      RETURN .ptr;
: 980      1062      !
: 981      1063      END;

```

			00FC 0000		.ENTRY EXCH\$UTIL_RMSB_ALLOCATE, Save R2,R3,R4,R5,-	0971
					R6,R7	
					MOVAB EXCH\$a_GBL, R7	
51		57 00000000G	EF 9E 00002		ADDL3 #160, EXCH\$a_GBL, R1	1016
		67 000000A0	8F C1 00009		REMQUE @0(R1), -T_	
		50 00	B1 0F 00011		BVC 1\$	
			04 1C 00015		CLRL PTR	
			56 D4 00017		BRB 2\$	
			03 11 00019		MOVL T, PTR	
		56	50 D0 0001B	1\$:	BNEQ 3\$	1020
			52 12 0001E	2\$:	MOVZWL #790, -(SP)	1026
		7E 0316	8F 3C 00020		CALLS #1, EXCH\$UTIL_VM_ALLOCATE_ZEROED	
		0000V	01 FB 00025		MOVL R0, PTR	
			50 D0 0002A		MOVL EXCH\$a_GBL, R0	1030
			50 D0 0002D		MOVL 148(R0), 12(PTR)	
		0C A6 0094	C0 D0 00030		MOVL PTR, 148(R0)	1031
		0094 C0	56 D0 00036			

EXCH\$UTIL
V04-000

Facility-wide misc routines
exch\$util_rmsb_allocate

D 14
16-Sep-1984 01:25:39
14-Sep-1984 12:29:09

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCUTIL.B32;1

Page 31
(15)

EX
VO

		08	A6	0316	8F	B0	0003B	MOVW	#790, 8(PTR)	:	1035
		0A	A6		0A	8E	00041	MNEGB	#10, 10(PTR)	:	
		10	A6	24	A6	9E	00045	MOVAB	36(R6), 16(PTR)	:	1039
14	A6	10	A6	00000050	8F	C1	0004A	ADDL3	#80, 16(PTR), 20(PTR)	:	1040
18	A6	14	A6	00000044	8F	C1	00054	ADDL3	#68, 20(PTR), 24(PTR)	:	1041
1C	A6	18	A6	00000060	8F	C1	0005E	ADDL3	#96, 24(PTR), 28(PTR)	:	1042
20	A6	1C	A6	000000FF	8F	C1	00068	ADDL3	#255, 28(PTR), 32(PTR)	:	1043
		52		031600F6	8F	D0	00072	MOVL	#51773686, R2	:	1049
		51		0197	8F	3C	00079	MOVZWL	#407, R1	:	
		50			56	D0	0007E	MOVL	PTR, R0	:	
				00000000G	EF	16	00081	JSB	EXCH\$UTIL_BLOCK_CHECK	:	
02F2	8F				00	2C	00087	MOVCS	#0, (SP), #0, #754, 36(PTR)	:	1053
				24	A6		0008E			:	
		50		00000098	8F	C1	00090	ADDL3	#152, EXCH\$A_GBL, R0	:	1057
		60			66	0E	00098	INSQUE	(PTR), (R0)	:	
		50			56	D0	0009B	MOVL	PTR, R0	:	1061
						04	0009E	RET		:	1063

; Routine Size: 159 bytes. Routine Base: EXCH\$UTIL_CODE + 03B9

```
1064 1 GLOBAL ROUTINE exch$util_rmsb_release (addr) : NOVALUE = %SBTTL 'exch$util_rmsb_release (addr)'  
1065 2 BEGIN  
1066 3 ++  
1067 4  
1068 5 FUNCTIONAL DESCRIPTION:  
1069 6  
1070 7 This routine deallocates one $RMSB. The $RMSB is moved from the in-use queue to the available queue  
1071 8  
1072 9 INPUTS:  
1073 10  
1074 11 addr - address of the block to release  
1075 12  
1076 13 IMPLICIT INPUTS:  
1077 14  
1078 15 exch$a_gbl [excg$q_rmsb_avl] - queue of available file blocks  
1079 16 exch$a_gbl [excg$q_rmsb_use] - queue of file blocks in use  
1080 17  
1081 18 OUTPUTS:  
1082 19 none  
1083 20  
1084 21 IMPLICIT OUTPUTS:  
1085 22 none  
1086 23  
1087 24 ROUTINE VALUE:  
1088 25 none  
1089 26  
1090 27 SIDE EFFECTS:  
1091 28  
1092 29 All errors are fatal  
1093 30  
1094 31 --  
1095 32  
1096 33 LOCAL  
1097 34 ptr : $ref_block, ! A local pointer to the rmsb  
1098 35 status  
1099 36 ;  
1100 37  
1101 38 ; First, move the pointer to a local variable  
1102 39  
1103 40 ptr = .addr;  
1104 41  
1105 42 ; Check our block type, fatal error if any problems  
1106 43  
1107 44 $block_check (2, .ptr, rmsb, 519);  
1108 45  
1109 46 ; Remove the rmsb from where ever it is in the in-use queue  
1110 47  
1111 48 $queue_remove (ptr [rmsb$q_header]);  
1112 49  
1113 50 ; Place the rmsb at the end of the available queue and the head of the in-use queue  
1114 51  
1115 52 $queue_insert_tail (ptr [rmsb$q_header], exch$a_gbl [excg$q_rmsb_avl]);  
1116 53  
1117 54  
1118 55  
1119 56  
1120 57 RETURN;
```

EXCH\$UTIL
V04-000

Facility-wide misc routines
exch\$util_rmsb_release (addr)

: 1040

1121 1 END;

F 14
16-Sep-1984 01:25:39
14-Sep-1984 12:29:09

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCUTIL.B32;1

Page 33
(16)

EXI
VOI

```

                    000C 00000
53          04      AC  D0 00002
52 031600F6      8F  D0 00006
51          0207   8F  3C 0000D
50          00000000G EF  16 00015
50 00000000G      63  OF 0001B
    04          B0  8F  C1 0001E
    04          63  OE 0002A
    04          04  0002E

```

```

.ENTRY EXCH$UTIL_RMSB_RELEASE, Save R2,R3      : 1064
MOVL   ADDR, PTR                               : 1106
MOVL   #51773686, R2                          : 1110
MOVZWL #519, R1
MOVL   PTR, R0
JSB    EXCH$UTIL_BLOCK_CHECK
REMQUE (PTR), T                               : 1114
ADDL3  #160, EXCR$A_GBL, R0                   : 1118
INSQUE (PTR), @4(R0)
RET                                           : 1121

```

: Routine Size: 47 bytes, Routine Base: EXCH\$UTIL_CODE + 0458

```
1042 1122 1 GLOBAL ROUTINE exch$util_rt11ctx_allocate (volb, filb) = %SBTTL 'exch$util_rt11ctx_allocate (volb, fi
1043 1123 2 BEGIN
1044 1124 2 ++
1045 1125 2
1046 1126 2 FUNCTIONAL DESCRIPTION:
1047 1127 2
1048 1128 2 This routine allocates one RT-11 file context block. If one is available, it is moved from the avai
1049 1129 2 queue to the in-use queue. If none are available, then a fresh block is created and placed on the i
1050 1130 2 queue.
1051 1131 2
1052 1132 2 INPUTS:
1053 1133 2
1054 1134 2 volb - pointer to the associated volb
1055 1135 2 filb - pointer to the associated filb
1056 1136 2
1057 1137 2 IMPLICIT INPUTS:
1058 1138 2
1059 1139 2 exch$a_gbl [excg$q_rt11ctx_all] - list of allocated file blocks
1060 1140 2 exch$a_gbl [excg$q_rt11ctx_avl] - queue of available file blocks
1061 1141 2 exch$a_gbl [excg$q_rt11ctx_use] - queue of file blocks in use
1062 1142 2
1063 1143 2 OUTPUTS:
1064 1144 2
1065 1145 2 none
1066 1146 2
1067 1147 2 IMPLICIT OUTPUTS:
1068 1148 2
1069 1149 2 none
1070 1150 2
1071 1151 2 ROUTINE VALUE:
1072 1152 2
1073 1153 2 address of the allocated file block
1074 1154 2
1075 1155 2 SIDE EFFECTS:
1076 1156 2
1077 1157 2 All errors are fatal
1078 1158 2 --
1079 1159 2
1080 1160 2 LOCAL
1081 1161 2 offset, ! Local temporary
1082 1162 2 ptr : $ref_bblock, ! A local pointer to the rt11ctx
1083 1163 2 status
1084 1164 2 ;
1085 1165 2
1086 1166 2
1087 1167 2 ! First, try to find one in the available queue
1088 1168 2
1089 1169 2 ptr = $queue_remove_head (exch$a_gbl [excg$q_rt11ctx_avl]);
1090 1170 2
1091 1171 2 ! If we didn't find one, then it will have to be created
1092 1172 2
1093 1173 2 IF .ptr EQL 0
1094 1174 2 THEN
1095 1175 2 BEGIN
1096 1176 2
1097 1177 2 ! Allocate a fresh rt11ctx from virtual memory. The entire block has been cleared to nulls
1098 1178 2 !
```

```

: 1099      1179      3      ptr = exch$util_vm_allocate_zeroed (exchblk$s_rt11ctx);
: 1100      1180
: 1101      1181      3      ! Place the rt11ctx at the head of the list of allocated blocks
: 1102      1182
: 1103      1183      3      ptr [rt11ctx$a_alloc] = .exch$a_gbl [excg$a_rt11ctx_alloc];
: 1104      1184      3      exch$a_gbl [excg$a_rt11ctx_alloc] = .ptr;
: 1105      1185
: 1106      1186      3      ! Set the block identification fields
: 1107      1187
: 1108      1188      3      $block_init (.ptr, rt11ctx);
: 1109      1189
: 1110      1190      3      END;
: 1111      1191
: 1112      1192      2      ! Check our block type, fatal error if any problems
: 1113      1193
: 1114      1194      2      $block_check (2, .ptr, rt11ctx, 486);
: 1115      1195
: 1116      1196      2      ! Set the last part of the block to nulls
: 1117      1197
: 1118      1198      2      CH$FILL (0, rt11ctx$k_end_zero - rt11ctx$k_start_zero, .ptr + rt11ctx$k_start_zero);
: 1119      1199
: 1120      1200      2      ! Insert the block at the head of the in-use queue
: 1121      1201
: 1122      1202      2      $queue_insert_head (ptr [rt11ctx$q_header], exch$a_gbl [excg$q_rt11ctx_use]);
: 1123      1203
: 1124      1204      2      ! Set the two associated fields
: 1125      1205
: 1126      1206      2      ptr [rt11ctx$a_assoc_volb] = .volb;
: 1127      1207      2      ptr [rt11ctx$a_assoc_filb] = .filb;
: 1128      1208
: 1129      1209      2      ! Return the address of the file block to the caller
: 1130      1210
: 1131      1211      2      RETURN .ptr;
: 1132      1212
: 1133      1213      1      END;

```

				00FC 0000	.ENTRY	EXCH\$UTIL_RT11CTX_ALLOCATE, Save R2,R3,R4,-	1122
						R5,R6,R7	
51	57	00000000G	EF	9E 00002	MOVAB	EXCH\$a_GBL, R7	
	67	000000B4	8F	C1 00009	ADDL3	#180, EXCH\$a_GBL, R1	1169
	50	00	B1	0F 00011	REMQUE	@0(R1), _T_	
			04	1C 00015	BVC	1\$	
			56	D4 00017	CLRL	PTR	
			03	11 00019	BRB	2\$	
	56		50	D0 0001B	MOVL	_T_, PTR	
			23	12 0001E	BNEQ	3\$	1173
	7E	82	8F	9A 00020	MOVZBL	#130, -(SP)	1179
0000V	CF		01	FB 00024	CALLS	#1, EXCH\$UTIL_VM_ALLOCATE_ZEROED	
	56		50	D0 00029	MOVL	R0, PTR	
	50		67	D0 0002C	MOVL	EXCH\$a_GBL, R0	1183
0C	A6	00A8	C0	D0 0002F	MOVL	168(R0), 12(PTR)	
00A8	C0		56	D0 00035	MOVL	PTR, 168(R0)	1184
08	A6	82	8F	9B 0003A	MOVZBW	#130, 8(PTR)	1188

EXCH\$UTIL
V04-000

Facility-wide misc routines
exch\$util_rt11ctx_allocate (volb, filb)

I 14
16-Sep-1984 01:25:39
14-Sep-1984 12:29:09

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCUTIL.B32;1

Page 36
(17)

EX
VO

	0A	A6		0C	8E	0003F	MNEGB	#12, 10(PTR)	:	
		52	008200F4	8F	DO	00043	38: MOVL	#8519924, R2	:	1194
		51	01E6	8F	3C	0004A	MOVZWL	#486, R1	:	
		50		56	DO	0004F	MOVL	PTR, R0	:	
0066	8F		00000000G	EF	16	00052	JSB	EXCH\$UTIL_BLOCK_CHECK	:	
		6E		00	2C	00058	MOVCS	#0, (SP), #0, #T02, 28(PTR)	:	1198
			1C	A6		0005F			:	
		67	000000AC	8F	C1	00061	ADDL3	#172, EXCH\$A_GBL, R0	:	1202
	50	60		66	0E	00069	INSQUE	(PTR), (R0)	:	
		14	A6	04	AC	0006C	MOVL	VOLB, 20(PTR)	:	1206
		10	A6	08	AC	00071	MOVL	FILB, 16(PTR)	:	1207
			50	56	DO	00076	MOVL	PTR, R0	:	1211
				04	00079		RET		:	1213

; Routine Size: 122 bytes. Routine Base: EXCH\$UTIL_CODE + 0487

```
1135 1214 1 GLOBAL ROUTINE exch$util_rt11ctx_release (addr) : NOVALUE = %SBTTL 'exch$util_rt11ctx_release (addr)'  
1136 1215 2 BEGIN  
1137 1216 2 ++  
1138 1217 2  
1139 1218 2 FUNCTIONAL DESCRIPTION:  
1140 1219 2  
1141 1220 2 This routine deallocates one rt11ctx. The block is moved from the in-use queue to the available que  
1142 1221 2  
1143 1222 2 INPUTS:  
1144 1223 2  
1145 1224 2 addr - address of the block to release  
1146 1225 2  
1147 1226 2 IMPLICIT INPUTS:  
1148 1227 2  
1149 1228 2 exch$a_gbl [excg$q_rt11ctx_avl] - queue of available file blocks  
1150 1229 2 exch$a_gbl [excg$q_rt11ctx_use] - queue of file blocks in use  
1151 1230 2  
1152 1231 2 OUTPUTS:  
1153 1232 2  
1154 1233 2 none  
1155 1234 2  
1156 1235 2 IMPLICIT OUTPUTS:  
1157 1236 2  
1158 1237 2 none  
1159 1238 2  
1160 1239 2 ROUTINE VALUE:  
1161 1240 2  
1162 1241 2 none  
1163 1242 2  
1164 1243 2 SIDE EFFECTS:  
1165 1244 2  
1166 1245 2 All errors are fatal  
1167 1246 2 --  
1168 1247 2  
1169 1248 2 LOCAL  
1170 1249 2 ptr : $ref_block, ! A local pointer to the rt11ctx  
1171 1250 2 status  
1172 1251 2 ;  
1173 1252 2  
1174 1253 2  
1175 1254 2 ! First, move the pointer to a local variable  
1176 1255 2  
1177 1256 2 ptr = .addr;  
1178 1257 2  
1179 1258 2 ! Check our block type, fatal error if any problems  
1180 1259 2  
1181 1260 2 $block_check (2, .ptr, rt11ctx, 487);  
1182 1261 2  
1183 1262 2 ! If there is a buffer allocated, free it  
1184 1263 2  
1185 1264 2 IF .ptr [rt11ctx$a_buffer] NEQ 0  
1186 1265 2 THEN  
1187 1266 2 exch$util_vm_release (ctx$k_buffer_length, .ptr [rt11ctx$a_buffer]);  
1188 1267 2  
1189 1268 2 ! Clear the pointers in the part of the block before the automatic zero  
1190 1269 2  
1191 1270 2 ptr [rt11ctx$a_assoc_filb] = 0;
```

```

: 1192      1271 2 ptr [rt11ctx$a_assoc_volb] = 0;
: 1193      1272 2 ptr [rt11ctx$a_buffer] = 0;
: 1194      1273 2
: 1195      1274 2 ! Remove the rt11ctx from where ever it is in the in-use queue
: 1196      1275 2
: 1197      1276 2 $queue_remove (ptr [rt11ctx$q_header]);
: 1198      1277 2
: 1199      1278 2 ! Place the rt11ctx at the end of the available queue and the head of the in-use queue
: 1200      1279 2
: 1201      1280 2 $queue_insert_tail (ptr [rt11ctx$q_header], exch$a_gbl [excg$q_rt11ctx_avl]);
: 1202      1281 2
: 1203      1282 2 RETURN;
: 1204      1283 1 END;

```

			000C 00000	.ENTRY	EXCH\$UTIL_RT11CTX_RELEASE, Save R2,R3	: 1214
	53	04	AC D0 00002	MOVL	ADDR, PTR	: 1256
	52	008200F4	8F D0 00006	MOVL	#8519924, R2	: 1260
	51	01E7	8F 3C 0000D	MOVZWL	#487, R1	
	50		53 D0 00012	MOVL	PTR, R0	
		00000000G	EF 16 00015	JSB	EXCH\$UTIL_BLOCK_CHECK	
		18	A3 D5 0001B	TSTL	24(PTR)	: 1264
			0D 13 0001E	BEQL	1\$	
		18	A3 DD 00020	PUSHL	24(PTR)	: 1266
	7E	1800	8F 3C 00023	MOVZWL	#6144, -(SP)	
0000V	CF		02 FB 00028	CALLS	#2, EXCH\$UTIL_VM_RELEASE	
		10	A3 7C 0002D 1\$:	CLRQ	16(PTR)	: 1270
		18	A3 D4 00030	CLRL	24(PTR)	: 1272
	50		63 0F 00033	REMQUE	(PTR), T	: 1276
	50	00000000G	EF 00000B4	ADDL3	#180, EXCH\$a_GBL, R0	: 1280
	04	B0	63 0E 00042	INSQUE	(PTR), @4(R0)	
			04 00046	RET		: 1283

: Routine Size: 71 bytes, Routine Base: EXCH\$UTIL_CODE + 0501


```

: 1206 1284 1 GLOBAL ROUTINE exch$util_vm_allocate (size) = %SBTTL 'exch$util_vm_allocate (size)'
: 1207 1285 2 BEGIN
: 1208 1286 2 ++
: 1209 1287 2
: 1210 1288 2 FUNCTIONAL DESCRIPTION:
: 1211 1289 2
: 1212 1290 2 This routine calls the LIB$GET_VM service to allocate dynamic memory.
: 1213 1291 2
: 1214 1292 2 INPUTS:
: 1215 1293 2
: 1216 1294 2 size Number of bytes to allocate (by value)
: 1217 1295 2
: 1218 1296 2 IMPLICIT INPUTS:
: 1219 1297 2
: 1220 1298 2 none
: 1221 1299 2
: 1222 1300 2 OUTPUTS:
: 1223 1301 2
: 1224 1302 2 none
: 1225 1303 2
: 1226 1304 2 IMPLICIT OUTPUTS:
: 1227 1305 2
: 1228 1306 2 none
: 1229 1307 2
: 1230 1308 2 ROUTINE VALUE:
: 1231 1309 2
: 1232 1310 2 address of the allocated memory
: 1233 1311 2
: 1234 1312 2 SIDE EFFECTS:
: 1235 1313 2
: 1236 1314 2 All errors are fatal
: 1237 1315 2 --
: 1238 1316 2
: 1239 1317 2 LOCAL
: 1240 1318 2 addr,
: 1241 1319 2 status
: 1242 1320 2 ;
: 1243 1321 2
: 1244 1322 2 IF NOT (status = lib$get_vm (size, addr)) ! Pass the call through
: 1245 1323 2 THEN
: 1246 1324 2 $exch_signal_stop (.status);
: 1247 1325 2
: 1248 1326 2 RETURN .addr;
: 1249 1327 1 END;

```

					.EXTRN LIB\$GET_VM	
			0000 0000		.ENTRY EXCH\$UTIL_VM_ALLOCATE, Save nothing	: 1284
	5E		04 C2 00002		SUBL2 #4, SP	: 1322
			5E DD 00005		PUSHL SP	: 1324
		04	AC 9F 00007		PUSHAB SIZE	
	0000000G 00		02 FB 0000A		CALLS #2, LIB\$GET_VM	
	0A		50 EB 00011		BLBS STATUS, 1\$	
			50 DD 00014		PUSHL STATUS	: 1324
	0000000G 00		01 FB 00016		CALLS #1, LIB\$STOP	:

EXCH\$UTIL
V04-000

Facility-wide misc routines
exch\$util_vm_allocate (size)

M 14
16-Sep-1984 01:25:39
14-Sep-1984 12:29:09

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCUTIL.B32;1

Page 40
(19)

EX
VO

50

6E 04 0001D
D0 0001E 1\$:
04 00021

RET
MOVL ADDR, R0
RET

:
: 1326
: 1327

; Routine Size: 34 bytes, Routine Base: EXCH\$UTIL_CODE + 0548

```

1251 1328 1 GLOBAL ROUTINE exch$util_vm_allocate_zeroed (size) = %SBTTL 'exch$util_vm_allocate_zeroed (size)'
1252 1329 2 BEGIN
1253 1330 2 ++
1254 1331 2
1255 1332 2 FUNCTIONAL DESCRIPTION:
1256 1333 2
1257 1334 2 This routine allocates dynamic memory. The memory contents are set to nulls.
1258 1335 2
1259 1336 2 INPUTS:
1260 1337 2
1261 1338 2 size Number of bytes to allocate (by value)
1262 1339 2
1263 1340 2 IMPLICIT INPUTS:
1264 1341 2
1265 1342 2 none
1266 1343 2
1267 1344 2 OUTPUTS:
1268 1345 2
1269 1346 2 none
1270 1347 2
1271 1348 2 IMPLICIT OUTPUTS:
1272 1349 2
1273 1350 2 none
1274 1351 2
1275 1352 2 ROUTINE VALUE:
1276 1353 2
1277 1354 2 address of the allocated memory
1278 1355 2
1279 1356 2 SIDE EFFECTS:
1280 1357 2
1281 1358 2 All errors are fatal
1282 1359 2 --
1283 1360 2
1284 1361 2 REGISTER
1285 1362 2 addr, ! address of new memory
1286 1363 2 chunk : INITIAL (65535), ! used to force a large constant into a register
1287 1364 2 tmp_adr, ! temp pointer and size
1288 1365 2 tmp_siz
1289 1366 2 ;
1290 1367 2
1291 1368 2
1292 1369 2 ! Allocate the memory
1293 1370 2
1294 1371 2 addr = exch$util_vm_allocate (.size);
1295 1372 2
1296 1373 2 ! Zap the entire piece of memory to nulls. Since the VAX architecture only supports short strings, we must
1297 1374 2 ! it into 64K chunks
1298 1375 2
1299 1376 2 tmp_adr = .addr;
1300 1377 2 tmp_siz = .size;
1301 1378 2 WHILE .tmp_siz GTRU .chunk
1302 1379 2 DO
1303 1380 2 BEGIN
1304 1381 2 CH$FILL (0, .chunk, .tmp_adr);
1305 1382 2 tmp_adr = .tmp_adr + .chunk;
1306 1383 2 tmp_siz = .tmp_siz - .chunk;
1307 1384 2 END;

```

```

: 1308      1385 2
: 1309      1386 2 ; Do the last (usually only) piece of memor
: 1310      1387 2 ;
: 1311      1388 2 CH$FILL (0, .tmp_siz, .tmp_adr);
: 1312      1389 2
: 1313      1390 2 RETURN .adr;
: 1314      1391 1 END;

```

```

                                03FC 00000      .ENTRY EXCH$UTIL_VM_ALLOCATE_ZEROED, Save R2,R3,-
                                8F 3C 00002      R4,R5,R6,R7,R8,R9
                                04 AC DD 00007      #65535, CHUNK
                                DO AF 01 FB 0000A     MOVZWL SIZE
                                56 50 D0 0000E     PUSHL
                                58 56 D0 00011     CALLS #1, EXCH$UTIL_VM_ALLOCATE
                                59 04 AC D0 00014     MOVL R0, ADDR
                                57 59 D1 00018 1$:    MOVL ADDR, TMP_ADR
                                0E 1B 0001B         MOVL SIZE, TMP_SIZ
                                6E 00 2C 0001D     CMPL TMP_SIZ, CHUNK
                                68 00 00022     BLEQU 2$
                                58 57 C0 00023     MOVCS #0, (SP), #0, CHUNK, (TMP_ADR)
                                59 57 C2 00026     ADDL2 CHUNK, TMP_ADR
                                ED 11 00029     SUBL2 CHUNK, TMP_SIZ
                                6E 00 2C 0002B 2$:    BRB 1$
                                68 00 00030     MOVCS #0, (SP), #0, TMP_SIZ, (TMP_ADR)
                                50 56 D0 00031     MOVL ADDR, R0
                                04 00034     RET

```

; Routine Size: 53 bytes, Routine Base: EXCH\$UTIL_CODE + 056A

```

: 1316 1392 1 GLOBAL ROUTINE exch$util_vm_release (size, addr) : NOVALUE = %SBTTL 'exch$util_vm_release (size, addr)'
: 1317 1393 2 BEGIN
: 1318 1394 2 +-
: 1319 1395 2
: 1320 1396 2 FUNCTIONAL DESCRIPTION:
: 1321 1397 2
: 1322 1398 2 This routine calls the LIB$FREE_VM service to release dynamic memory.
: 1323 1399 2
: 1324 1400 2 INPUTS:
: 1325 1401 2
: 1326 1402 2 size Number of bytes to release (by value)
: 1327 1403 2 addr Address of longword containing address of memory to release
: 1328 1404 2
: 1329 1405 2 IMPLICIT INPUTS:
: 1330 1406 2
: 1331 1407 2 none
: 1332 1408 2
: 1333 1409 2 OUTPUTS:
: 1334 1410 2
: 1335 1411 2 none
: 1336 1412 2
: 1337 1413 2 IMPLICIT OUTPUTS:
: 1338 1414 2
: 1339 1415 2 none
: 1340 1416 2
: 1341 1417 2 ROUTINE VALUE:
: 1342 1418 2
: 1343 1419 2 Success, or status code of error converted to warning severity
: 1344 1420 2
: 1345 1421 2 SIDE EFFECTS:
: 1346 1422 2
: 1347 1423 2 Errors are signalled
: 1348 1424 2 --
: 1349 1425 2
: 1350 1426 2 LOCAL
: 1351 1427 2 status
: 1352 1428 2 ;
: 1353 1429 2
: 1354 1430 3 IF NOT (status = lib$free_vm (size, addr)) ! Pass the call through, no dots tho
: 1355 1431 2 THEN
: 1356 1432 2 $exch_signal_stop (.status);
: 1357 1433 2
: 1358 1434 2 RETURN;
: 1359 1435 1 END;

```

.EXTRN LIB\$FREE_VM

			0000 0000	.ENTRY	EXCH\$UTIL_VM_RELEASE, Save nothing	: 1392
	08	AC	9F 0002	PUSHAB	ADDR	: 1430
	04	AC	9F 0005	PUSHAB	SIZE	
00000000G	00	02	FB 0008	CALLS	#2, LIB\$FREE_VM	
	09	50	E8 000F	BLBS	STATUS, 1\$	
		50	DD 0012	PUSHL	STATUS	: 1432
00000000G	00	01	FB 0014	CALLS	#1, LIB\$STOP	
		04	0001B 1\$:	RET		: 1435

EXCH\$UTIL
V04-000

Facility-wide misc routines
exch\$util_vm_release (size, addr)

D 15
16-Sep-1984 01:25:39
14-Sep-1984 12:29:09

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCUTIL.B32:1

Page 44
(21)

; Routine Size: 28 bytes, Routine Base: EXCH\$UTIL_CODE + 059F

```
1361 1436 1 GLOBAL ROUTINE exch$util_vol_getdvi (devname : REF $desc_block, %SBTTL 'exch$util_vol_getdvi (devnam
1362 1437 1
1363 1438 2 BEGIN
1364 1439 2 ++
1365 1440 2
1366 1441 2 FUNCTIONAL DESCRIPTION:
1367 1442 2
1368 1443 2 Get standard device information for a volb
1369 1444 2
1370 1445 2 INPUTS:
1371 1446 2
1372 1447 2 devname - address of descriptor for device name
1373 1448 2
1374 1449 2 IMPLICIT INPUTS:
1375 1450 2
1376 1451 2 none
1377 1452 2
1378 1453 2 OUTPUTS:
1379 1454 2
1380 1455 2 volb - several characteristics fields in the volb are filled in
1381 1456 2
1382 1457 2 IMPLICIT OUTPUTS:
1383 1458 2
1384 1459 2 none
1385 1460 2
1386 1461 2 ROUTINE VALUE:
1387 1462 2
1388 1463 2 Success or worst error encountered.
1389 1464 2
1390 1465 2 SIDE EFFECTS:
1391 1466 2
1392 1467 2 none
1393 1468 2 --
1394 1469 2
1395 1470 2 $dbgtrc_prefix ('util_vol_getdvi> ');
1396 1471 2
1397 1472 2 LOCAL
1398 1473 2 status,
1399 1474 2 dev_item : VECTOR [22, LONG]
1400 1475 2 ;
1401 1476 2
1402 1477 2 $block_check (2, .volb, volb, 488);
1403 1478 2
1404 1479 2 ! Initialize the item list for the $GETDVI
1405 1480 2 !
1406 1481 2 dev_item [0] = (dvi$ devbufsiz^16 OR 4); ! Device buffer size, output length 4
1407 1482 2 dev_item [1] = volb [volb$l_devbufsiz]; ! Address of output buffer
1408 1483 2 dev_item [2] = 0; ! No returned length
1409 1484 2 dev_item [3] = (dvi$ devchar^16 OR 4);
1410 1485 2 dev_item [4] = volb [volb$l_devchar];
1411 1486 2 dev_item [5] = 0;
1412 1487 2 dev_item [6] = (dvi$ devclass^16 OR 4);
1413 1488 2 dev_item [7] = volb [volb$l_devclass];
1414 1489 2 dev_item [8] = 0;
1415 1490 2 dev_item [9] = (dvi$ devdepend^16 OR 4);
1416 1491 2 dev_item [10] = volb [volb$l_devdepend];
1417 1492 2 dev_item [11] = 0;
```

```

1418 1493 2 dev_item [12] = (dvi$ fulldevnam^16 OR 16);
1419 1494 2 dev_item [13] = volb [volb$t_devnam];
1420 1495 2 dev_item [14] = volb [volb$l_devnamlen];
1421 1496 2 dev_item [15] = (dvi$ devtype^16 OR 4);
1422 1497 2 dev_item [16] = volb [volb$l_devtype];
1423 1498 2 dev_item [17] = 0;
1424 1499 2 dev_item [18] = (dvi$ maxblock^16 OR 4);
1425 1500 2 dev_item [19] = volb [volb$l_devmaxblock];
1426 1501 2 dev_item [20] = 0;
1427 1502 2 dev_item [21] = 0;                                ! End of GETDVI item List
1428 1503 2
1429 1504 2 ! Get the device information
1430 1505 2
1431 1506 2 IF NOT (status = $getdviw (efn=0, devnam=.devname, itmlst=dev_item))
1432 1507 2 THEN
1433 1508 2     RETURN .status;
1434 1509 2
1435 1510 2 ! Do any manipulations necessary with the raw device info
1436 1511 2
1437 1512 2 volb [volb$l_volmaxblock] = .volb [volb$l_devmaxblock]; ! Assume device and volume same size
1438 1513 2
1439 1514 2 ! Debugging trace code
1440 1515 2
1441 1516 2 %IF switch_trace
1442 1517 2 %THEN
1443 1518 2     BEGIN
1444 1519 2     EXTERNAL ROUTINE
1445 1520 2     exch$dbg_utl_print_devchar;
1446 1521 2     LOCAL
1447 1522 2     tmp_desc : $desc_block;
1448 1523 2     BIND
1449 1524 2     dep = volb [volb$l_devdepend] : $block;
1450 1525 2
1451 1526 2     $stat_str_desc_init (tmp_desc, .volb [volb$l_devnamlen], volb [volb$t_devnam]);
1452 1527 2     $trace_print_fao ('Getdvi for name "!AS" resolved to device "!AS"', .devname, tmp_desc);
1453 1528 2     $trace_print_fao ('Bufsiz = !UL, Maxblocks = !UL, Class = !XB, Type = !XB', .volb [volb$l_devbufsiz],
1454 1529 2     .volb [volb$l_devmaxblock], .volb [volb$l_devclass], .volb [volb$l_devtype]);
1455 1530 2     $trace_print_fao ('Cylinders = !UL, Tracks = !UL, Sectors = !UL, DevChar = !XL',
1456 1531 2     .dep[0,16,16,0], .dep[0,8,8,0], .dep[0,0,8,0], .volb [volb$l_devchar]);
1457 1532 2     exch$dbg_utl_print_devchar (.volb [volb$l_devchar]);
1458 1533 2     END;
1459 1534 2 %FI
1460 1535 2
1461 1536 2 RETURN .status;
1462 1537 2 END;

```

.EXTRN SYSSGETDVIW

```

SE      AB      AE      000C 00000
53      08      AC      9E 00002
52 041B00F3 8F      D0 00006
51      01EB    8F      3C 00011
50      0000000G 53      D0 00016
                    EF      16 00019

```

```

.ENTRY EXCH$UTIL_VOL_GETDVI, Save R2,R3      : 1436
MOVAB  -88(SP), SP
MOVL   VOLB, R3                               : 1477
MOVL   #68878579, R2
MOVZWL #488, R1
MOVL   R3, R0
JSB    EXCH$UTIL_BLOCK_CHECK

```


04	6E	00080004	8F	D0	0001F	MOVL	#524292, DEV_ITEM	:	1481
	AE	28	A3	9E	00026	MOVAB	40(R3), DEV_ITEM+4	:	1482
		08	AE	D4	0002B	CLRL	DEV_ITEM+8	:	1483
0C	AE	00020004	8F	D0	0002E	MOVL	#131076, DEV_ITEM+12	:	1484
10	AE	2C	A3	9E	00036	MOVAB	44(R3), DEV_ITEM+16	:	1485
		14	AE	D4	0003B	CLRL	DEV_ITEM+20	:	1486
18	AE	00040004	8F	D0	0003E	MOVL	#262148, DEV_ITEM+24	:	1487
1C	AE	30	A3	9E	00046	MOVAB	48(R3), DEV_ITEM+28	:	1488
		20	AE	D4	0004B	CLRL	DEV_ITEM+32	:	1489
24	AE	000A0004	8F	D0	0004E	MOVL	#655364, DEV_ITEM+36	:	1490
28	AE	34	A3	9E	00056	MOVAB	52(R3), DEV_ITEM+40	:	1491
		2C	AE	D4	0005B	CLRL	DEV_ITEM+44	:	1492
30	AE	00E80010	8F	D0	0005E	MOVL	#15204368, DEV_ITEM+48	:	1493
34	AE	00E9	C3	9E	00066	MOVAB	233(R3), DEV_ITEM+52	:	1494
38	AE	38	A3	9E	0006C	MOVAB	56(R3), DEV_ITEM+56	:	1495
3C	AE	00060004	8F	D0	00071	MOVL	#393220, DEV_ITEM+60	:	1496
40	AE	3C	A3	9E	00079	MOVAB	60(R3), DEV_ITEM+64	:	1497
		44	AE	D4	0007E	CLRL	DEV_ITEM+68	:	1498
48	AE	001A0004	8F	D0	00081	MOVL	#1703940, DEV_ITEM+72	:	1499
4C	AE	40	A3	9E	00089	MOVAB	64(R3), DEV_ITEM+76	:	1500
		50	AE	7C	0008E	CLRQ	DEV_ITEM+80	:	1501
			7E	7C	00091	CLRQ	-(SP)	:	1506
			7E	7C	00093	CLRQ	-(SP)	:	
		10	AE	9F	00095	PUSHAB	DEV_ITEM	:	
		04	AC	DD	00098	PUSHL	DEVNAME	:	
			7E	7C	0009B	CLRQ	-(SP)	:	
00000000G	00		08	FB	0009D	CALLS	#8, SYSSGETDVIW	:	
	05		50	E9	000A4	BLBC	STATUS, 1\$:	
44	A3	40	A3	D0	000A7	MOVL	64(R3), 68(R3)	:	1512
			04	000AC	1\$:	RET		:	1537

; Routine Size: 173 bytes, Routine Base: EXCH\$UTIL_CODE + 05BB

EX
MC
--
EX
DE
LC
DE
TR
EX
EX
MC
SN
IN
MA
SY
LI
SY

```
1464 1538 1 GLOBAL ROUTINE exch$util_volb_allocate = %SBTTL 'exch$util_volb_allocate'
1465 1539 2 BEGIN
1466 1540 2 ++
1467 1541 2
1468 1542 2 FUNCTIONAL DESCRIPTION:
1469 1543 2
1470 1544 2 This routine allocates one $VOLB. If $VOLBs are available, one is moved from the available queue to
1471 1545 2 in-use queue. If none are available, then a fresh $VOLB is created and placed on the in-use queue.
1472 1546 2
1473 1547 2 INPUTS:
1474 1548 2
1475 1549 2 none
1476 1550 2
1477 1551 2 IMPLICIT INPUTS:
1478 1552 2
1479 1553 2 exch$a_gbl [excg$a_volb_alloc] - list of allocated volume blocks
1480 1554 2 exch$a_gbl [excg$q_volb_avl] - queue of available volume blocks
1481 1555 2 exch$a_gbl [excg$q_volb_use] - queue of volume blocks in use
1482 1556 2
1483 1557 2 OUTPUTS:
1484 1558 2
1485 1559 2 none
1486 1560 2
1487 1561 2 IMPLICIT OUTPUTS:
1488 1562 2
1489 1563 2 none
1490 1564 2
1491 1565 2 ROUTINE VALUE:
1492 1566 2
1493 1567 2 address of the allocated volume block
1494 1568 2
1495 1569 2 SIDE EFFECTS:
1496 1570 2
1497 1571 2 All errors are fatal
1498 1572 2 --
1499 1573 2
1500 1574 2 LOCAL
1501 1575 2 offset, ! Local temporary
1502 1576 2 ptr : $ref_block, ! A local pointer to the volb
1503 1577 2 status
1504 1578 2 ;
1505 1579 2
1506 1580 2
1507 1581 2 ! First, try to find one in the available queue
1508 1582 2
1509 1583 2 ptr = $queue_remove_head (exch$a_gbl [excg$q_volb_avl]);
1510 1584 2
1511 1585 2 ! If we didn't find one, then it will have to be created
1512 1586 2
1513 1587 2 IF .ptr EQL 0
1514 1588 2 THEN
1515 1589 2 BEGIN
1516 1590 2
1517 1591 2 ! Allocate a fresh volb from virtual memory. The entire block has been cleared to nulls
1518 1592 2
1519 1593 2 ptr = exch$util_vm_allocate_zeroed (exchblk$s_volb);
1520 1594 2
```

```

1521 1595 3 ! Place the volb at the head of the list of allocated blocks
1522 1596 3 !
1523 1597 3 ptr [volb$a_alloc] = .exch$a_gbl [excg$a_volb_alloc];
1524 1598 3 exch$a_gbl [excg$a_volb_alloc] = .ptr;
1525 1599 3 !
1526 1600 3 ! Set the block identification fields
1527 1601 3 !
1528 1602 3 $block_init (.ptr, volb);
1529 1603 3 !
1530 1604 3 ! Several items are located at the end of the $VOLB, fill in the pointers
1531 1605 3 !
1532 1606 3 ptr [volb$a_fab] = .ptr + volb$k_length; ! Fab is at end of block
1533 1607 3 ptr [volb$a_rab] = .ptr [volb$a_fab] + fab$k_bln; ! Rab right after Fab
1534 1608 3 ptr [volb$a_nam] = .ptr [volb$a_rab] + rab$k_bln; ! Nam after Rab
1535 1609 3 ptr [volb$a_esbuf] = .ptr [volb$a_nam] + nam$k_bln; ! Expanded string after Nam
1536 1610 3 ptr [volb$a_rsbuf] = .ptr [volb$a_esbuf] + nam$c_maxrss; ! Result string after Ebuf
1537 1611 3 !
1538 1612 2 2 END;
1539 1613 2 2 !
1540 1614 2 2 ! Check our block type, fatal error if any problems
1541 1615 2 2 !
1542 1616 2 2 $block_check (2, .ptr, volb, 489);
1543 1617 2 2 !
1544 1618 2 2 ! Set the last part of the block to nulls
1545 1619 2 2 !
1546 1620 2 2 CH$FILL (0, exchblk$s_volb - volb$k_start_zero, .ptr + volb$k_start_zero);
1547 1621 2 2 !
1548 1622 2 2 ! Place the volb at the head of the in-use queue
1549 1623 2 2 !
1550 1624 2 2 $queue_insert_head (ptr [volb$q_header], exch$a_gbl [excg$q_volb_use]);
1551 1625 2 2 !
1552 1626 2 2 ! Return the address of the volume block to the caller
1553 1627 2 2 !
1554 1628 2 2 RETURN .ptr;
1555 1629 2 2 !
1556 1630 1 1 END;

```

			00FC 0000C	.ENTRY	EXCH\$UTIL_VOLB_ALLOCATE, Save R2,R3,R4,R5,-	1538
					R6,R7	
51	57	00000000G	EF 9E 00002	MOVAB	EXCH\$a_GBL, R7	
	67	0000000C8	8F C1 00009	ADDL3	#200, EXCH\$a_GBL, R1	1583
	50	00	B1 0F 00011	REMQUE	@0(R1), _T_	
			04 1C 00015	BVC	1\$	
			56 D4 00017	CLRL	PTR	
			03 11 00019	BRB	2\$	
	56		50 D0 0001B	MOVL	T_, PTR	
			53 12 0001E	BNEQ	3\$	1587
	7E	041B	8F 3C 00020	MOVZWL	# 1, -(SP)	1593
FED8	CF		01 FB 00025	CALLS	# (CH\$UTIL_VM_ALLOCATE_ZEROED	
	56		50 D0 0002A	MOVL	R0, R	
	50		67 D0 0002D	MOVL	EXCH\$a_GBL, R0	1597
0C	A6	00BC	C0 D0 00030	MOVL	188(R0), 12(PTR)	
00BC	C0		56 D0 00036	MOVL	PTR, 188(R0)	1598

EXCH\$UTIL
V04-000

Facility-wide misc routines
exch\$util_volb_allocate

J 15
16-Sep-1984 01:25:39
14-Sep-1984 12:29:09

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCUTIL.B32;1

Page 50
(23)

		08	A6	041B	8F	B0	0003B	MOVW	#1051, 8(PTR)	1602	
		0A	A6		0D	8E	00041	MNEGB	#13, 10(PTR)	1603	
		10	A6	0129	C6	9E	00045	MOVAB	297(R6), 16(PTR)	1606	
14	A6	10	A6	00000050	8F	C1	0004B	ADDL3	#80, 16(PTR), 20(PTR)	1607	
18	A6	14	A6	00000044	8F	C1	00055	ADDL3	#68, 20(PTR), 24(PTR)	1608	
1C	A6	18	A6	00000060	8F	C1	0005F	ADDL3	#96, 24(PTR), 28(PTR)	1609	
20	A6	1C	A6	000000FF	8F	C1	00069	ADDL3	#255, 28(PTR), 32(PTR)	1610	
		52	041B00F3		8F	D0	00073	MOVL	#68878579, R2	1616	
		51	01E9		8F	3C	0007A	MOVZWL	#489, R1		
		50			56	D0	0007F	MOVL	PTR, R0		
			00000000G		EF	16	00082	JSB	EXCH\$UTIL_BLOCK_CHECK		
03F3	8F				00	2C	00088	MOVCS	#0, (SP), -#0, #T011, 40(PTR)	1620	
			28		A6		0008F				
		50			67	000000C0	8F	C1	#192, EXCH\$A_GBL, R0	1624	
					60		66	0E	(PTR), (R0)		
					50		D0	0009C	MOVL	PTR, R0	1628
							04	0009F	RET	1630	

; Routine Size: 160 bytes, Routine Base: EXCH\$UTIL_CODE + 0668

```
1558 1631 1 GLOBAL ROUTINE exch$util_volb_release (addr) : NOVALUE = %SBTTL 'exch$util_volb_release (addr)'  
1559 1632 2 BEGIN  
1560 1633 2 ++  
1561 1634 2  
1562 1635 2 FUNCTIONAL DESCRIPTION:  
1563 1636 2  
1564 1637 2 This routine deallocates one $VOLB. The $VOLBs is moved from the in-use queue to the available queue  
1565 1638 2  
1566 1639 2 INPUTS:  
1567 1640 2  
1568 1641 2 addr - address of the block to release  
1569 1642 2  
1570 1643 2 IMPLICIT INPUTS:  
1571 1644 2  
1572 1645 2 exch$a_gbl [excg$q_volb_avl] - queue of available volume blocks  
1573 1646 2 exch$a_gbl [excg$q_volb_use] - queue of volume blocks in use  
1574 1647 2  
1575 1648 2 OUTPUTS:  
1576 1649 2  
1577 1650 2 none  
1578 1651 2  
1579 1652 2 IMPLICIT OUTPUTS:  
1580 1653 2  
1581 1654 2 none  
1582 1655 2  
1583 1656 2 ROUTINE VALUE:  
1584 1657 2  
1585 1658 2 none  
1586 1659 2  
1587 1660 2 SIDE EFFECTS:  
1588 1661 2  
1589 1662 2 All errors are fatal  
1590 1663 2 --  
1591 1664 2  
1592 1665 2 LOCAL  
1593 1666 2 ptr : $ref_block, ! A local pointer to the volb  
1594 1667 2 spc : $ref_block, ! Pointer to volume specific structure  
1595 1668 2 status  
1596 1669 2 ;  
1597 1670 2  
1598 1671 2  
1599 1672 2 ! First, move the pointer to a local variable  
1600 1673 2  
1601 1674 2 ptr = .addr;  
1602 1675 2  
1603 1676 2 ! Check our block type, fatal error if any problems  
1604 1677 2  
1605 1678 2 $block_check (2, .ptr, volb, 490);  
1606 1679 2  
1607 1680 2 ! Perform some volume specific actions on the specific pointer  
1608 1681 2  
1609 1682 2 IF (spc = .ptr [volb$a_vfmt_specific]) NEQ 0  
1610 1683 2 THEN  
1611 1684 2 BEGIN  
1612 1685 2 LOCAL  
1613 1686 2 block_size  
1614 1687 2 ;
```

```

: 1615 1688 3
: 1616 1689 3 CASE .ptr [volb$b_vol_format] FROM volb$k_vfmt_lobound TO volb$k_vfmt_hibound OF
: 1617 1690 3 SET
: 1618 1691 4 [volb$k_vfmt_dos11] : BEGIN
: 1619 1692 4 LOCAL
: 1620 1693 4 ent : $ref_block;
: 1621 1694 4
: 1622 1695 4 ! Follow the chain of entries and release them
: 1623 1696 4 !
: 1624 1697 5 WHILE (ent = $queue_remove_head (spc [dos11$q_entry_header]))
: 1625 1698 4 DO
: 1626 1699 4 exch$util_vm_release (dos11ent$k_length, .ent); ! Release the entry
: 1627 1700 4
: 1628 1701 4 block_size = exchblk$s_dos11;
: 1629 1702 4 END;
: 1630 1703 3
: 1631 1704 3 [volb$k_vfmt_rt11] : block_size = exchblk$s_rt11;
: 1632 1705 3
: 1633 1706 3 [INRANGE, OVRANGE] : $logic_check (0, (false), 250);
: 1634 1707 3
: 1635 1708 3 TES;
: 1636 1709 3
: 1637 1710 3 exch$util_vm_release (.block_size, .spc); ! Release the extension
: 1638 1711 2 END;
: 1639 1712 2
: 1640 1713 2 ! Remove the volb from where ever it is in the in-use queue
: 1641 1714 2 !
: 1642 1715 2 $queue_remove (ptr [volb$q_header]);
: 1643 1716 2
: 1644 1717 2 ! Place the volb at the end of the available queue
: 1645 1718 2 !
: 1646 1719 2 $queue_insert_tail (ptr [volb$q_header], exch$a_gbl [excg$q_volb_avl]);
: 1647 1720 2
: 1648 1721 2 RETURN;
: 1649 1722 1 END;

```

```

003D 0008 001D 0008 00026 1$:
      54 04 AC 001C 0000 .ENTRY EXCH$UTIL_VOLB_RELEASE, Save R2,R3,R4 : 1631
      52 041B00F3 8F DO 00002 MOVL ADDR, PTR : 1674
      51 01EA 8F DO 00006 MOVL #68878579, R2 : 1678
      50 00000000G EF 3C 0000D MOVZWL #490, R1
      53 54 A4 DO 00012 MOVL PTR, R0
      4E 13 0001F JSB EXCH$UTIL_BLOCK_CHECK : 1682
      00 58 A4 DO 0001B MOVL 84(PTR), SPC
      00 0008 8F 00021 BEQL 9$ : 1689
      00000000G 00 03 0008 00026 2$: .WORD 2$-1$,-
      7E FA 8F 9A 0002E 2$: 3$-1$,-
      00000000G 00 01 DD 00032 MOVZBL #250, -(SP) : 1706
      00000000G 00 8F DD 00034 PUSHL #1
      00000000G 00 03 FB 0003A PUSHL #EXCH$ BADLOGIC
      CALLS #3, LIB$STOP

```



```

1651 1723 1 GLOBAL ROUTINE exch$util_up_case (in_siz, in_ptr, out_ptr) : NOVALUE jsb_r1r2r3 = %SBITL 'exch$util_up
1652 1724 2 BEGIN
1653 1725 2 ++
1654 1726 2
1655 1727 2 FUNCTIONAL DESCRIPTION:
1656 1728 2
1657 1729 2 This routine converts a string to uppercase. In testing it appears to be faster to do this sort
1658 1730 2 of loop than to execute the MOVTC instruction on the 117780.
1659 1731 2
1660 1732 2 INPUTS:
1661 1733 2
1662 1734 2 in_siz = size of input record to convert
1663 1735 2 in_ptr = address of input record to convert
1664 1736 2
1665 1737 2 IMPLICIT INPUTS:
1666 1738 2
1667 1739 2 none
1668 1740 2
1669 1741 2 OUTPUTS:
1670 1742 2
1671 1743 2 out_ptr = address of output record buffer
1672 1744 2
1673 1745 2 IMPLICIT OUTPUTS:
1674 1746 2
1675 1747 2 none
1676 1748 2
1677 1749 2 ROUTINE VALUE:
1678 1750 2
1679 1751 2 none
1680 1752 2
1681 1753 2 SIDE EFFECTS:
1682 1754 2
1683 1755 2 Input record copied to output record buffer and all
1684 1756 2 lowercase alphabetic characters converted to uppercase.
1685 1757 2 --
1686 1758 2
1687 1759 2 REGISTER
1688 1760 2 char : BYTE ! Character to test
1689 1761 2 ;
1690 1762 2
1691 1763 2 DECR count FROM .in_siz-1 TO 0 ! Upcase the characters
1692 1764 2 DO
1693 1765 2 BEGIN
1694 1766 2 char = CH$RCHAR_A (in_ptr); ! Get next character
1695 1767 2 IF .char GEQU 'a' AND .char LEQU 'z' ! Lower case letter?
1696 1768 2 THEN
1697 1769 2 char = .char - %0'40'; ! Convert to upper
1698 1770 2 CH$WCHAR_A (.char,out_ptr); ! Move character to buffer
1699 1771 2 END;
1700 1772 2
1701 1773 2 RETURN;
1702 1774 1 END;

```

B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
{
|
}
~
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^
_
`
a
b
c
d
e
f
g
h
i
j
k
l
m
n

EXCH\$UTIL
V04-000

Facility-wide misc routines
exch\$util_up_case

B 16
16-Sep-1984 01:25:39
14-Sep-1984 12:29:09

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCUTIL.B32;1

Page 55
(25)

```
15 11 00000 EXCH$UTIL UP_CASE::  
BRB 3$  
61 50 82 90 00002 1$: MOVB (IN_PTR)+, CHAR : 1763  
8F 50 91 00005 CMPB CHAR, #97 : 1766  
09 1F 00009 BLSSU 2$ : 1767  
7A 8F 50 91 0000B CMPB CHAR, #122 :  
03 1A 0000F BGTRU 2$ :  
50 20 82 00011 SUBB2 #32, CHAR : 1769  
83 50 90 00014 2$: MOVB CHAR, (OUT_PTR)+ : 1770  
E8 51 F4 00017 3$: SOBGEQ COUNT, 1$ : 1763  
05 0001A RSB : 1774
```

; Routine Size: 27 bytes, Routine Base: EXCH\$UTIL_CODE + 078B

EXCH\$UTIL Facility-wide misc routines
V04-000 exch\$util_up_case

C 16
16-Sep-1984 01:25:39
14-Sep-1984 12:29:09

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCUTIL.B32;1

Page 56
(26)

: 1704 1775 1 END
: 1705 1776 0 ELUDOM

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name Bytes Attributes
EXCH\$UTIL_CODE 1958 NOVEC,NOWRT, RD, EXE,NOSHR, LCL, REL, CON,NOPIE,ALIGN(2)

Library Statistics

File	Symbols		Pages Mapped	Processing Time
	Total	Loaded Percent		
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	30 0	1000	00:01.8
_\$255\$DUA28:[EXCHNG.OBJ]EXCLIB.L32;1	1151	140 12	79	00:01.4

COMMAND QUALIFIERS

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

: Size: 1958 code + 0 data bytes
: Run Time: 00:40.4
: Elapsed Time: 02:18.9
: Lines/CPU Min: 2640
: Lexemes/CPU-Min: 23715
: Memory Used: 141 pages
: Compilation Complete

