


```

EEEEEEEEEE XX      XX  CCCCCCCC FFFFFFFFFF IIIIII LL      11
EEEEEEEEEE XX      XX  CCCCCCCC FFFFFFFFFF IIIIII LL      11
EE          XX      XX  CC        FF          II     LL      1111  1111
EE          XX      XX  CC        FF          II     LL      1111  1111
EE          XX      XX  CC        FF          II     LL      11    11
EE          XX      XX  CC        FF          II     LL      11    11
EEEEEEEEEE      XX      XX  CC        FFFFFFFF III     LL      11    11
EFFFFFFF      XX      XX  CC        FFFFFFFF III     LL      11    11
EE          XX      XX  CC        FF          II     LL      11    11
EE          XX      XX  CC        FF          II     LL      11    11
EE          XX      XX  CC        FF          II     LL      11    11
EEEEEEEEEE XX      XX  CCCCCCCC FF          IIIIII LLLLLLLLLL 111111 111111
EEEEEEEEEE XX      XX  CCCCCCCC FF          IIIIII LLLLLLLLLL 111111 111111

```

```

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

```

.....

```

1 0001 0 MODULE  exch$fil11                                %TITLE 'Files-11 volume specific 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:      Files-11 specific routines
35 0035 1
36 0036 1 ENVIRONMENT:    VAX/VMS User mode
37 0037 1
38 0038 1 AUTHOR:        CW Hobbs                CREATION DATE: 26-Aug-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          Supply the close routine as the delete routine for files-11
44 0044 1          to prevent leaving files open after random errors.
45 0045 1
46 0046 1 --
47 0047 1
48 0048 1 ! Include files:
49 0049 1
50 0050 1 MACRO $module_name string = 'exch$fil11' %;          ! The require file needs to know our module name
51 0051 1 REQUIRE 'SRC$:EXCREQ'                                ! Facility-wide require file
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\$fil11_close_file,	: Files-11 specific file close routine
60	0155	1	exch\$fil11_create_file,	: Files-11 specific file create routine
61	0156	1	exch\$fil11_get,	: Get record
62	0157	1	exch\$fil11_open_file,	: Files-11 specific file open routine
63	0158	1	exch\$fil11_put	: Put record
64	0159	1	;	
65	0160	1		
66	0161	1	: EXCHANGE facility routines	
67	0162	1	: :	
68	0163	1	EXTERNAL ROUTINE	
69	0164	1	exch\$cmd_related_file_parse,	: Perform an RMS output file parse
70	0165	1	exch\$util_file_error,	: Signal an RMS error
71	0166	1	exch\$util_rmsb_allocate	: Get an RMSB
72	0167	1	;	
73	0168	1		
74	0169	1	: Equated symbols:	
75	0170	1	: :	
76	0171	1	LITERAL	
77	0172	1	: :	
78	0173	1		
79	0174	1	: Bound declarations:	
80	0175	1	: :	
81	0176	1	BIND	
82	0177	1	: :	

```
84 0178 1 GLOBAL ROUTINE exch$fil11_close_file (filb : $ref_bblock) = %SBTTL 'exch$fil11_close_file (filb)'  
85 0179 2 BEGIN  
86 0180 2 ++  
87 0181 2  
88 0182 2 FUNCTIONAL DESCRIPTION:  
89 0183 2  
90 0184 2 Perform files-11 volume specific close processing  
91 0185 2  
92 0186 2 INPUT/OUTPUT:  
93 0187 2  
94 0188 2 filb - pointer to block describing the file  
95 0189 2  
96 0190 2 IMPLICIT INPUTS:  
97 0191 2  
98 0192 2 none  
99 0193 2  
100 0194 2 OUTPUTS:  
101 0195 2  
102 0196 2 filb - receive info pertaining to the file to be closed  
103 0197 2  
104 0198 2 IMPLICIT OUTPUTS:  
105 0199 2  
106 0200 2 none  
107 0201 2  
108 0202 2 ROUTINE VALUE:  
109 0203 2  
110 0204 2 true if able to close the file, false otherwise  
111 0205 2  
112 0206 2 SIDE EFFECTS:  
113 0207 2  
114 0208 2 none  
115 0209 2 --  
116 0210 2  
117 0211 2 $dbgtrc_prefix ('fil11_close_file> ');  
118 0212 2  
119 0213 2 LOCAL  
120 0214 2 status  
121 0215 2 ;  
122 0216 2  
123 0217 2 BIND  
124 0218 2 namb = filb [filb$a_assoc_namb] : $ref_bblock,  
125 0219 2 ctx = filb [filb$a_context] : $ref_bblock,  
126 0220 2 fab = ctx [rmsb$a_fab] : $ref_bblock  
127 0221 2 ;  
128 0222 2  
129 0223 2 $debug_print_lit ('entry');  
130 0224 2  
131 0225 2 $block_check (2, .filb, filb, 497);  
132 0226 2 $block_check (2, .namb, namb, 498);  
133 0227 2 $block_check (2, .ctx, rmsb, 499);  
134 0228 2  
135 0229 2  
136 0230 2 ! Close the file  
137 0231 2  
138 0232 2 $trace_print_fao ('closing, fab=XL', .fab);  
139 0233 2 IF NOT (status = $close (fab = .fab))  
140 0234 2 THEN
```

!?? definitely over-zealous checking

```

: 141      0235 2      exch$util_file_error (exch$_closeerr, .status, .fab, .fab [fab$_stv]);
: 142      0236 2
: 143      0237 2 RETURN .status;
: 144      0238 1 END;

```

```

.TITLE EXCH$FIL11 Files-11 volume specific routines
.IDENT \V04-000\

```

```

.EXTRN EXCH$CMD RELATED FILE_PARSE
.EXTRN EXCH$UTIL_FILE_ERROR
.EXTRN EXCH$UTIL_RMSB_ALLOCATE
.EXTRN EXCH$UTIL_BLOCK_CHECK
.EXTRN SYSSCLOSE, EXCH$_CLOSEERR

```

```

.PSECT EXCH$FIL11_CODE, NOWRT, 2

```

			007C 00000	.ENTRY EXCH\$FIL11_CLOSE_FILE, Save R2,R3,R4,R5,R6	: 0178
		56 0000000G	EF 9E 00002	MOVAB EXCH\$UTIL_BLOCK_CHECK, R6	
53	04	AC	18 C1 00009	ADDL3 #24, FILB, R3	: 0218
54	04	AC	20 C1 0000E	ADDL3 #32, FILB, R4	: 0219
55		64	10 C1 00013	ADDL3 #16, (R4), R5	: 0220
		52 035B00FA	8F D0 00017	MOVL #56295674, R2	: 0225
		51 01F1	8F 3C 0001E	MOVZWL #497, R1	
		50 04	AC D0 00023	MOVL FILB, R0	
			66 16 00027	JSB EXCH\$UTIL_BLOCK_CHECK	
		52 010A00F7	8F D0 00029	MOVL #17432823, R2	: 0226
		51 01F2	8F 3C 00030	MOVZWL #498, R1	
		50	63 D0 00035	MOVL (R3), R0	
			66 16 00038	JSB EXCH\$UTIL_BLOCK_CHECK	
		52 031600F6	8F D0 0003A	MOVL #51773686, R2	: 0227
		51 01F3	8F 3C 00041	MOVZWL #499, R1	
		50	64 D0 00046	MOVL (R4), R0	
			66 16 00049	JSB EXCH\$UTIL_BLOCK_CHECK	
		52	65 D0 0004B	MOVL (R5), R2	: 0233
			52 DD 0004E	PUSHL R2	
00000000G		00	01 FB 00050	CALLS #1, SYSSCLOSE	
		53	50 D0 00057	MOVL R0, STATUS	
		14	53 E8 0005A	BLBS STATUS, 1\$	
			A2 DD 0005D	PUSHL 12(R2)	: 0235
			52 DD 00060	PUSHL R2	
			53 DD 00062	PUSHL STATUS	
		00000000G	8F DD 00064	PUSHL #EXCH\$_CLOSEERR	
00000000G		EF	04 FB 0006A	CALLS #4, EXCH\$UTIL_FILE_ERROR	
		50	53 DC 00071 1\$:	MOVL STATUS, R0	: 0237
			04 00074	RET	: 0238

: Routine Size: 117 bytes, Routine Base: EXCH\$FIL11_CODE + 0000

```

146 0239 1 GLOBAL ROUTINE exch$fil11_create_file = %SBTTL 'exch$fil11_create_file'
147 0240 2 BEGIN
148 0241 2 ++
149 0242 2
150 0243 2 FUNCTIONAL DESCRIPTION:
151 0244 2
152 0245 2 Perform Files-11 volume specific create processing
153 0246 2
154 0247 2 INPUT:
155 0248 2
156 0249 2 none
157 0250 2
158 0251 2 IMPLICIT INPUTS:
159 0252 2
160 0253 2 copy [copy$a_out_filb] - pointer to filb for the output file
161 0254 2 copy [copy$a_inp_filb] - pointer to filb for the input file
162 0255 2
163 0256 2 OUTPUTS:
164 0257 2
165 0258 2 out_filb - receive info pertaining to the created file
166 0259 2
167 0260 2 IMPLICIT OUTPUTS:
168 0261 2
169 0262 2 none
170 0263 2
171 0264 2 ROUTINE VALUE:
172 0265 2
173 0266 2 true if able to create a file, false otherwise
174 0267 2
175 0268 2 SIDE EFFECTS:
176 0269 2
177 0270 2 none
178 0271 2 --
179 0272 2
180 0273 2 $dbgtrc_prefix ('fil11_create_file> ');
181 0274 2
182 0275 2 LOCAL
183 0276 2 rfp : $bblock [nam$b_bln+nam$b_maxrss], ! An RMS NAM block plus the expanded string buffer for output
184 0277 2 status
185 0278 2 ;
186 0279 2
187 0280 2 BIND
188 0281 2 copy = exch$a_gbl [excg$a_copy_work] : $ref_bblock,
189 0282 2 out_name = copy [copy$a_output_filename] : $desc_block,
190 0283 2 inp_filb = copy [copy$a_inp_filb] : $ref_bblock,
191 0284 2 out_filb = copy [copy$a_out_filb] : $ref_bblock,
192 0285 2 ctx = out_filb [filb$a_context] : $ref_bblock,
193 0286 2 out_namb = out_filb [filb$a_assoc_namb] : $ref_bblock
194 0287 2 ;
195 0288 2
196 0289 2 $debug_print_lit ('entry');
197 0290 2
198 0291 2 $block_check (2, .out_filb, filb, 511);
199 0292 2 $block_check (2, .inp_filb, filb, 525);
200 0293 2 $block_check (2, .out_namb, namb, 512);
201 0294 2 $logic_check (2, (.out_filb [filb$a_assoc_volb] EQL 0), 138);
202 0295 2

```

```

203 0296 2 ! If the context block is null, then allocate an RMSB
204 0297 2
205 0298 2 IF .ctx EQL 0
206 0299 2 THEN
207 0300 2     ctx = exch$util_rmsb_allocate ( )           ! Get a fresh one
208 0301 2 ELSE
209 0302 2     $block_check (2, .ctx, rmsb, 513);       ! Check the old one
210 0303 2
211 0304 2 BEGIN
212 0305 2 BIND
213 0306 2     fab = ctx [rmsb$a_fab] : $ref_bblock,
214 0307 2     rab = ctx [rmsb$a_rab] : $ref_bblock,
215 0308 2     nam = ctx [rmsb$a_nam] : $ref_bblock;
216 0309 2
217 0310 2 ! Create a name string in the out_filb for the 'NOTCOPIED' message, just in case we exit with an error
218 0311 2
219 0312 2 out_filb [filb$l_result_name_len] = .out_name [dsc$w_length];
220 0313 2 CH$COPY (.out_name [dsc$w_length], .out_name [dsc$a_pointer], 0,
221 0314 2     filb$s_result_name, out_filb [filb$t_result_name]);
222 0315 2
223 0316 2 ! Perform an RMS output file parse on the related name (the result name for the input file) and the
224 0317 2 ! requested output name from the command line.
225 0318 2
226 0319 2 IF NOT (status = exch$cmd_related_file_parse (
227 0320 2     .out_name [dsc$w_length], .out_name [dsc$a_pointer],           ! Command line out p
228 0321 2     .inp_filb [filb$t_result_name_len], inp_filb [filb$t_result_name], ! Related name
229 0322 2     rfp) )                                                         ! Gets new name
230 0323 2 THEN
231 0324 2     $exch_signal_return (exch$_openout, 1, out_name, .status);
232 0325 2
233 0326 2 $trace_print_fao ('trying to create '!AF'', .rfp [nam$b_esl], .rfp [nam$l_esa]);
234 0327 2
235 0328 2 ! Initialize the RMS structures
236 0329 2
237 P 0330 2 $fab_init (
238 P 0331 2     FAB = .fab,           ! Output file FAB
239 P 0332 2     FAC = (BRO,PUT),     ! Put only, block I/O in case we can do things faster than
240 P 0333 2     FNA = .rfp [nam$l_esa], ! Set name addr
241 P 0334 2     FNS = .rfp [nam$b_esl], ! Set name size
242 P 0335 2     FOP = SEQ,          ! Sequential only
243 P 0336 2     NAM = .nam,         ! Name block
244 P 0337 2     RAT = CR,          ! Carriage-return carriage control
245 P 0338 2     RFM = VAR,         ! Variable-length records
246 P 0339 2     SHR = (GET,PUT,UPI)); ! Allow other readers/writers
247 P 0340 2 $rab_init (
248 P 0341 2     RAB = .rab,         ! Output file RAB
249 P 0342 2     MBF = 2,           ! Multi-buffer count (MBC from process or system default)
250 P 0343 2     RAC = SEQ,         ! Sequential only
251 P 0344 2     ROP = WBH,         ! Write behind
252 P 0345 2     FAB = .fab);       ! FAB addr
253 P 0346 2 $nam_init (
254 P 0347 2     NAM = .nam,         ! File name block
255 P 0348 2     RSA = .ctx [rmsb$a_rsbuf], ! Result name addr
256 P 0349 2     RSS = nam$c_maxrss, ! Result name size
257 P 0350 2     ESA = .ctx [rmsb$a_esbuf], ! Expanded name addr
258 0351 2     ESS = nam$c_maxrss); ! Expanded name size
259 0352 2

```



```
260 0353 3 ! Set the desired file attributes
261 0354 3
262 0355 3 fab [fab$v_mxv] = NOT .out_filb [filb$v_explicit_version]; ! Use explicit version if given, otherwise m
263 0356 3
264 0357 3 ! We allow several Files-11 "output" qualifiers to be placed on the input parameter. We interpret "output"
265 0358 3 ! qualifiers on the output spec (or the verb) as applying to all output files. "Output" qualifiers on the
266 0359 3 ! input specs apply to files created for that input spec. If on both, use the one from the output (or verb)
267 0360 3
268 0361 4 fab [fab$l_alq] = (IF .copy [copy$l_q_allocation] NEQ 0 ! If specified on the output
269 0362 4 THEN ! then
270 0363 4 .copy [copy$l_q_allocation] ! use that quantity
271 0364 4 ELSE IF .inp_filb [filb$l_q_allocation] NEQ 0 ! otherwise if /ALLOCATION was on the input
272 0365 4 THEN ! then
273 0366 4 .inp_filb [filb$l_q_allocation] ! use the /ALLOC quantity from the input f
274 0367 4 ELSE ! otherwise
275 0368 3 .inp_filb [filb$l_block_count]); ! use the size of the input file.
276 0369 3
277 0370 4 fab [fab$w_deq] = (IF .copy [copy$l_q_extension] NEQ 0
278 0371 4 THEN
279 0372 4 .copy [copy$l_q_extension]
280 0373 4 ELSE
281 0374 3 .inp_filb [filb$l_q_extension]);
282 0375 3
283 0376 4 fab [fab$v_cbt] = (IF .copy [copy$v_q_best_try_contiguous] ! Best try - overrides /contiguous if both p
284 0377 4 THEN
285 0378 4 true
286 0379 4 ELSE
287 0380 3 .inp_filb [filb$v_q_best_try_contiguous]);
288 0381 3
289 0382 4 fab [fab$v_ctg] = (IF .copy [copy$v_q_contiguous]
290 0383 4 THEN
291 0384 4 true
292 0385 4 ELSE
293 0386 3 .inp_filb [filb$v_q_contiguous]);
294 0387 3
295 0388 4 fab [fab$v_tef] = (IF .copy [copy$v_q_truncate] ! Truncate over-allocations
296 0389 4 THEN
297 0390 4 true
298 0391 4 ELSE
299 0392 3 .inp_filb [filb$v_q_truncate]);
300 0393 3 !?? should truncate depend on explicit allocation and/or /TR
301 0394 3
302 0395 3 ! If /RECORD_FORMAT was given then tell him we are ignoring
303 0396 3
304 0397 3 IF .out_filb [filb$v_rfmt_explicit]
305 0398 3 THEN
306 0399 4 BEGIN
307 0400 4 out_filb [filb$v_rfmt_explicit] = false;
308 0401 4 out_filb [filb$b_rec_format] = filb$k_rfmt_invalid;
309 0402 4 $exch_signal (exch$_fil11_norec);
310 0403 3 END;
311 0404 3
312 0405 3 ! If /CARRIAGE_CONTROL was given on either input or output then set the record attribute
313 0406 3
314 0407 3 IF .out_filb [filb$v_ctl_explicit]
315 0408 3 THEN
316 0409 4 fab [fab$b_rat] = (CASE .out_filb [filb$b_car_control] FROM filb$k_ctl_lobound TO filb$k_ctl_hibound
```

```
317 0410 4 SET
318 0411 4 [filb$sk_ctl_cr] : fab$m_cr;
319 0412 4 [filb$sk_ctl_fortran] : fab$m_ftn;
320 0413 4 [filb$sk_ctl_none] : 0;
321 0414 4 TES)
322 0415 3 ELSE IF .inp_filb [filb$v_ctl_explicit]
323 0416 3 THEN
324 0417 4 fab [fab$b_rat] = (CASE .inp_filb [filb$b_car_control] FROM filb$sk_ctl_lobound TO filb$sk_ctl_hibound
325 0418 4 SET
326 0419 4 [filb$sk_ctl_cr] : fab$m_cr;
327 0420 4 [filb$sk_ctl_fortran] : fab$m_ftn;
328 0421 4 [filb$sk_ctl_none] : 0;
329 0422 3 TES);
330 0423 3
331 0424 3 ! See if we need to override the record format, variable by default. We do not allow record format qualifie
332 0425 3 ! (except for block transfer) on Files-11 filespecs, so get all record format info from the input file.
333 0426 3
334 0427 3 IF .out_filb [filb$b_transfer_mode] EQL filb$sk_xfrm_block
335 0428 3 OR
336 0429 3 .inp_filb [filb$b_transfer_mode] EQL filb$sk_xfrm_block
337 0430 3 THEN
338 0431 4 BEGIN
339 0432 4 fab [fab$w_mrs] = 512;
340 0433 4 fab [fab$b_rfm] = fab$c_fix;
341 0434 4 END
342 0435 3 ELSE IF .inp_filb [filb$b_rec_format] EQL filb$sk_rfmt_fixed
343 0436 3 THEN
344 0437 4 BEGIN
345 0438 4 fab [fab$w_mrs] = .inp_filb [filb$l_fixed_len];
346 0439 4 fab [fab$b_rfm] = fab$c_fix;
347 0440 3 END;
348 0441 3
349 0442 3 ! Create and connect to the file
350 0443 3
351 0444 4 IF NOT (status = $create (fab = .fab))
352 0445 3 THEN
353 0446 4 BEGIN
354 0447 4 exch$util_file_error (exch$_openout, .status, .fab, .fab [fab$l_stv]);
355 0448 4 RETURN 0; ! Don't pass any status so that we won't get a chained messa
356 0449 3 END; ! attached to the 'NOTCOPIED' message
357 0450 3
358 0451 3 ! Create the result name string in the out_filb
359 0452 3
360 0453 3 $logic_check (2, ((.nam [nam$b_rsl] LEQU filb$s_result_name) AND (.nam [nam$b_rsl] GTRU 0)), 139);
361 0454 3 out_filb [filb$l_result_name_len] = .nam [nam$b_rsl];
362 0455 3 CH$COPY (.nam [nam$b_rsl], .nam [nam$l_rsa], 0, filb$s_result_name, out_filb [filb$t_result_name]);
363 0456 3
364 0457 3 $trace_print_fao ('Created '!AF'', .out_filb [filb$l_result_name_len], out_filb [filb$t_result_name]);
365 0458 3
366 0459 4 IF NOT (status = $connect (rab = .rab))
367 0460 3 THEN
368 0461 4 BEGIN
369 0462 4 exch$util_file_error (exch$_openout, .status, .fab, .rab [rab$l_stv]);
370 0463 4 $close (fab = .fab);
371 0464 4 RETURN 0; ! Don't pass any status so that we won't get a chained messa
372 0465 3 END; ! attached to the 'NOTCOPIED' message
373 0466 3
```

```

374 0467 3 ! Define a record stream for this file
375 0468 3 !
376 0469 3 out_filb [filb$a_record] = 0; ! No valid record or length
377 0470 3 out_filb [filb$a_record_len] = 0;
378 0471 3 out_filb [filb$v_files_created] = true; ! Made a file using this filb
379 0472 3
380 0473 3 ! Make sure that the record format in the filb is correct
381 0474 3 !
382 0475 3 !?? record format is in the rms structures
383 0476 3
384 0477 3 ! Save the addresses of our routines for this volume and record format.
385 0478 3 !
386 0479 3 out_filb [filb$a_close_routine] = exch$fil11_close_file;
387 0480 3 out_filb [filb$a_delete_routine] = exch$fil11_close_file;
388 0481 3 out_filb [filb$a_put_routine] = exch$fil11_put;
389 0482 3 out_filb [filb$a_get_routine] = 0; ! We don't want to do this, so make it hard
390 0483 3
391 0484 2 END; ! End of BIND to the rmsb components
392 0485 2
393 0486 2 RETURN true;
394 0487 2
395 0488 1 END;

```

```

                                .EXTRN EXCH$A_GBL, EXCH$_BADLOGIC
                                .EXTRN EXCH$_FIL11_NOREC
                                .EXTRN SYSS$CREATE, SYSS$CONNECT

                                .ENTRY EXCH$FIL11_CREATE_FILE, Save R2,R3,R4,R5,-
                                R6,R7,R8,R9,R10,RT1
50 00000000G 5E FE98 CE 9E 00002 MOVAB -360(SP), SP
EF 04 C1 00007 ADDL3 #4, EXCH$A_GBL, R0
58 60 D0 0000F MOVL (R0), R8
57 14 A8 9E 00012 MOVAB 20(R8), R7
56 44 A8 D0 00016 MOVL 68(R8), R6
52 035B00FA 8F D0 0001A MOVL #56295674, R2
51 01FF 8F 3C 00021 MOVZWL #511, R1
50 56 D0 00026 MOVL R6, R0
00000000G EF 16 00029 JSB EXCH$UTIL_BLOCK_CHECK
5A 3C A8 D0 0002F MOVL 60(R8), R10
52 035B00FA 8F D0 00033 MOVL #56295674, R2
51 020D 8F 3C 0003A MOVZWL #525, R1
50 5A D0 0003F MOVL R10, R0
00000000G EF 16 00042 JSB EXCH$UTIL_BLOCK_CHECK
52 010A00F7 8F D0 00048 MOVL #17432823, R2
51 0200 8F 3C 0004F MOVZWL #512, R1
50 18 A6 D0 00054 MOVL 24(R6), R0
00000000G EF 16 00058 JSB EXCH$UTIL_BLOCK_CHECK
1C A6 D5 0005E 1STL 28(R6)
13 13 00061 BEQL 1$
7E 8A 8F 9A 00063 MOVZBL #138, -(SP)
01 DD 00067 PUSHL #1
00000000G 8F DD 00069 PUSHL #EXCH$_BADLOGIC
00 03 FB 0006F CALLS #3, LIB$STOP
00000000G 00 20 A6 D5 00076 1$: TSTL 32(R6)
OD 12 00079 BNEQ 2$

```

			00000000G	EF	00	FB	0007B	CALLS	#0, EXCH\$UTIL_RMSB_ALLOCATE	0300	
			20	A6	50	DO	00082	MOVL	R0, 32(R6)		
					16	11	00086	BRB	3\$		
				52	031600F6	8F	DO	00088	2\$: MOVL	#51773686, R2	0302
				51	0201	8F	3C	0008F	MOVZWL	#513, R1	
				50	20	A6	DO	00094	MOVL	32(R6), R0	
					00000000G	EF	16	00098	JSB	EXCH\$UTIL_BLOCK_CHECK	
				6E	20	A6	DO	0009E	3\$: MOVL	32(R6), (SP)	0306
0100	8F	00		3A		67	3C	000A2	MOVZWL	(R7), 58(R6)	0312
				04		67	2C	000A6	MOVCS	(R7), 24(R7), #0, #256, 90(R6)	0314
					5A	A6		000AE			
					08	AE	9F	000B0	PUSHAB	RFP	0321
					5A	AA	9F	000B3	PUSHAB	90(R10)	
					3A	AA	DD	000B6	PUSHL	58(R10)	
					04	A7	DD	000B9	PUSHL	4(R7)	
						67	3C	000BC	MOVZWL	(R7), -(SP)	
			00000000G	7E		05	FB	000BF	CALLS	#5, EXCH\$CMD_RELATED_FILE_PARSE	
			04	EF		50	DO	000C6	MOVL	R0, STATUS	
				AE	04	AE	E8	000CA	BLBS	STATUS, 4\$	
				1E		8F	DO	000CE	MOVL	#16257184, TEMP	0324
				52	00F810A0	AE	DD	000D5	PUSHL	STATUS	
					04	57	DD	000D8	PUSHL	R7	
						01	DD	000DA	PUSHL	#1	
						52	DD	000DC	PUSHL	TEMP	
			00000000G	00		04	FB	000DE	CALLS	#4, LIB\$SIGNAL	
				50		52	DO	000E5	MOVL	TEMP, R0	
						04		000E8	RET		
				50		10	C1	000E9	4\$: ADDL3	#16, (SP), R0	0339
						60	DO	000ED	MOVL	(R0), R7	
0050	8F	00		6E		00	2C	000F0	MOVCS	#0, (SP), #0, #80, (R7)	
						67		000F7			
					5003	8F	B0	000FB	MOVW	#20483, (R7)	
				04	40	8F	9A	000FD	MOVZBL	#64, 4(R7)	
				16	4341	8F	B0	00102	MOVW	#17217, 22(R7)	
				1E	0202	8F	B0	00108	MOVW	#514, 30(R7)	
				50		18	C1	0010E	ADDL3	#24, (SP), R0	
						60	DO	00112	MOVL	(R0), R9	
				28		59	DO	00115	MOVL	R9, 40(R7)	
						AE	DO	00119	MOVL	RFP+12, 44(R7)	
				2C	14	AE	90	0011E	MOVB	RFP+11, 52(R7)	
				34	13	AE	90	0011E	ADDL3	#20, (SP), R0	0345
				50		14	C1	00123	MOVL	(R0), R11	
						60	DO	00127	MOVCS	#0, (SP), #0, #68, (R11)	
0044	8F	00		6E		00	2C	0012A			
						6B		00131			
					4401	8F	B0	00132	MOVW	#17409, (R11)	
				04	0400	8F	3C	00137	MOVZWL	#1024, 4(R11)	
					1E	AB	94	0013D	CLRB	30(R11)	
				36		02	90	00140	MOVB	#2, 54(R11)	
				3C		57	DO	00144	MOVL	R7, 60(R11)	
0060	8F	00		6E		00	2C	00148	MOVCS	#0, (SP), #0, #96, (R9)	0351
						69		0014F			
					6002	8F	B0	00150	MOVW	#24578, (R9)	
				02		01	8E	00155	MNEGB	#1, 2(R9)	
				50		20	C1	00159	ADDL3	#32, (SP), R0	
						60	DO	0015D	MOVL	(R0), 4(R9)	
				04		01	8E	00161	MNEGB	#1, 10(R9)	
				0A		1C	C1	00165	ADDL3	#28, (SP), R0	
				50							

				0C	A9		60	D0	00169	MOV	(R0), 12(R9)		
					6E	28	A6	9E	0016D	MOVAB	43(R6), (SP)		0355
	50	00	BE		01		05	EF	00171	EXTZV	#5, #1, 20(SP), R0		
					50		50	D2	00177	MCOML	R0, R0		
04	A7		01		01		50	F0	0017A	INSV	R0, #1, #1, 4(R7)		
						24	A8	D5	00180	TSTL	36(R8)		0361
							06	13	00183	BEQL	5\$		
					50	24	A8	D0	00185	MOV	36(R8), R0		0363
							0F	11	00189	BRB	7\$		
						2D	AA	D5	0018B	5\$: TSTL	45(R10)		0364
							06	13	0018E	BEQL	6\$		
					50	2D	AA	D0	00190	MOV	45(R10), R0		0366
							04	11	00194	BRB	7\$		
					50	3E	AA	D0	00196	6\$: MOV	62(R10), R0		0368
				10	A7		50	D0	0019A	7\$: MOV	R0, 16(R7)		0361
						28	A8	D5	0019E	TSTL	40(R8)		0370
							06	13	001A1	BEQL	8\$		
					50	28	A8	D0	001A3	MOV	40(R8), R0		0372
							04	11	001A7	BRB	9\$		
					50	31	AA	D0	001A9	8\$: MOV	49(R10), R0		0374
				14	A7		50	B0	001AD	9\$: MOVW	R0, 20(R7)		0370
					05	30	A8	E9	001B1	BLBC	48(R8), 10\$		0376
					50		01	D0	001B5	MOV	#1, R0		
							06	11	001B8	BRB	11\$		
	50	2C	AA		01		00	EF	001BA	10\$: EXTZV	#0, #1, 44(R10), R0		0380
06	A7		01		05		50	F0	001C0	11\$: INSV	R0, #5, #1, 6(R7)		0376
			05		30		01	E1	001C6	BBC	#1, 48(R8), 12\$		0382
					50		01	D0	001CB	MOV	#1, R0		
							06	11	001CE	BRB	13\$		
	50	2C	AA		01		01	EF	001D0	12\$: EXTZV	#1, #1, 44(R10), R0		0386
06	A7		01		04		50	F0	001D6	13\$: INSV	R0, #4, #1, 6(R7)		0382
			05		31		02	E1	001DC	BBC	#2, 49(R8), 14\$		0388
					50		01	D0	001E1	MOV	#1, R0		
							06	11	001E4	BRB	15\$		
	50	2C	AA		01		02	EF	001E6	14\$: EXTZV	#2, #1, 44(R10), R0		0392
07	A7		01		04		50	F0	001EC	15\$: INSV	R0, #4, #1, 7(R7)		0388
					14		00	BE	001F2	BLBC	20(SP), 16\$		0397
				00	BE		01	8A	001F6	BICB2	#1, 20(SP)		0400
						28	A6	94	001FA	CLRB	40(R6)		0401
						00000000G	8F	DD	001FD	PUSHL	#EXCH\$ FIL11 NOREC		0402
							01	FB	00203	CALLS	#1, LIB\$SIGNAL		
			0D		00		01	E1	0020A	16\$: BBC	#1, 20(SP), 18\$		0407
			02		00		2A	A6	8F	0020F	CASEB	42(R6), #0, #2	0409
			0022		001D		0018	00214	17\$: .WORD	20\$-17\$,-			
										21\$-17\$,-			
										22\$-17\$			
							10	11	0021A	BRB	20\$		
			1B		28		01	E1	0021C	18\$: BBC	#1, 43(R10), 24\$		0415
			02		00		2A	AA	8F	00221	CASEB	42(R10), #0, #2	0417
			0010		000B		0006	00226	19\$: .WORD	20\$-19\$,-			
										21\$-19\$,-			
										22\$-19\$			
					50		02	D0	0022C	20\$: MOV	#2, R0		
							07	11	0022F	BRB	25\$		
					50		01	D0	00231	21\$: MOV	#1, R0		
							02	11	00234	BRB	23\$		
							50	D4	00236	22\$: CLRL	R0		

1E	A7		50	90	00238	23\$:	MOVB	R0, 30(R7)		
	01	29	A6	91	0023C	24\$:	CMPB	41(R6), #1	0427	
			06	13	00240		BEQL	25\$		
	01	29	AA	91	00242		CMPB	41(R10), #1	0429	
			08	12	00246		BNEQ	26\$		
36	A7	0200	8F	B0	00248	25\$:	MOVW	#512, 54(R7)	0432	
			0B	11	0024E		BRB	27\$	0433	
	02	28	AA	91	00250	26\$:	CMPB	40(R10), #2	0435	
			09	12	00254		BNEQ	28\$		
36	A7	35	AA	B0	00256		MOVW	53(R10), 54(R7)	0438	
1F	A7		01	90	0025B	27\$:	MOVB	#1, 31(R7)	0439	
			57	DD	0025F	28\$:	PUSHL	R7	0444	
00000000G	00		01	FB	00261		CALLS	#1, SYSS\$CREATE		
04	AE		50	DD	00268		MOVL	R0, STATUS		
	17	04	AE	E8	0026C		BLBS	STATUS, 29\$		
		0C	A7	DD	00270		PUSHL	12(R7)	0447	
			57	DD	00273		PUSHL	R7		
		0C	AE	DD	00275		PUSHL	STATUS		
		00F810A0	8F	DD	00278		PUSHL	#16257184		
00000000G	EF		04	FB	0027E		CALLS	#4, EXCH\$UTIL_FILE_ERROR		
			78	11	00285		BRB	32\$	0448	
	52	03	A9	9A	00287	29\$:	MOVZBL	3(R9), R2	0453	
			13	12	00289		BNEQ	30\$		
	7E	8B	8F	9A	0028D		MOVZBL	#139, -(SP)		
			01	DD	00291		PUSHL	#1		
		00000000G	8F	DD	00293		PUSHL	#EXCH\$ BADLOGIC		
00000000G	00		03	FB	00299		CALLS	#3, LIB\$STOP		
3A	A6		52	DD	002A0	30\$:	MOVL	R2, 58(R6)	0454	
04	B9		52	2C	002A4		MOVCS	R2, 24(R9), #0, #256, 90(R6)	0455	
		5A	A6		002AC					
			5B	DD	002AE		PUSHL	R11	0459	
00000000G	00		01	FB	002B0		CALLS	#1, SYSS\$CONNECT		
04	AE		50	DD	002B7		MOVL	R0, STATUS		
	20	04	AE	E8	002BB		BLBS	STATUS, 31\$		
		0C	AB	DD	002BF		PUSHL	12(R11)	0462	
			57	DD	002C2		PUSHL	R7		
		0C	AE	DD	002C4		PUSHL	STATUS		
		00F810A0	8F	DD	002C7		PUSHL	#16257184		
00000000G	EF		04	FB	002CD		CALLS	#4, EXCH\$UTIL_FILE_ERROR		
			57	DD	002D4		PUSHL	R7	0463	
00000000G	00		01	FB	002D6		CALLS	#1, SYSS\$CLOSE		
			20	11	002DD		BRB	32\$	0464	
		42	A6	7C	002DF	31\$:	CLRQ	66(R6)	0470	
00	BE		10	88	002E2		BISB2	#16, 20(SP)	0471	
4A	A6	FCA1	CF	9E	002E6		MOVAB	EXCH\$FIL11_CLOSE_FILE, 74(R6)	0479	
4E	A6	FC9B	CF	9E	002EC		MOVAB	EXCH\$FIL11_CLOSE_FILE, 78(R6)	0480	
56	A6	0000V	CF	9E	002F2		MOVAB	EXCH\$FIL11_PUT, 86(R6)	0481	
		52	A6	D4	002F8		CLRL	82(R6)	0482	
	50		01	DD	002FB		MOVL	#1, R0	0486	
				04	002FE		RET			
			50	D4	002FF	32\$:	CLRL	R0	0488	
			04	00301			RET			

; Routine Size: 770 bytes, Routine Base: EXCH\$FIL11_CODE + 0075

```
397 0489 1 GLOBAL ROUTINE exch$fil11_get (filb : $ref_bblock) = %SBTTL 'exch$fil11_get (filb)'  
398 0490 2 BEGIN  
399 0491 2 !++  
400 0492 2  
401 0493 2 FUNCTIONAL DESCRIPTION:  
402 0494 2  
403 0495 2 Return a pointer to the next fixed-length record in the file  
404 0496 2  
405 0497 2 INPUTS:  
406 0498 2  
407 0499 2 filb - pointer to filb for an open Files-11 file  
408 0500 2  
409 0501 2 IMPLICIT INPUTS:  
410 0502 2  
411 0503 2 none  
412 0504 2  
413 0505 2 OUTPUTS:  
414 0506 2  
415 0507 2 none  
416 0508 2  
417 0509 2 IMPLICIT OUTPUTS:  
418 0510 2  
419 0511 2 none  
420 0512 2  
421 0513 2 ROUTINE VALUE:  
422 0514 2  
423 0515 2 true if success, false if any error  
424 0516 2  
425 0517 2 SIDE EFFECTS:  
426 0518 2  
427 0519 2 error conditions will be signaled  
428 0520 2 !--  
429 0521 2  
430 0522 2 $dbgtrc_prefix ('fil11_get> ');  
431 0523 2  
432 0524 2 LOCAL  
433 0525 2 status  
434 0526 2 ;  
435 0527 2  
436 0528 2 BIND  
437 0529 2 namb = filb [filb$a_assoc_namb] : $ref_bblock,  
438 0530 2 ctx = filb [filb$a_context] : $ref_bblock,  
439 0531 2 fab = ctx [rmsb$a_fab] : $ref_bblock,  
440 0532 2 rab = ctx [rmsb$a_rab] : $ref_bblock  
441 0533 2 ;  
442 0534 2  
443 0535 2 $debug_print_lit ('entry');  
444 0536 2  
445 0537 2 $block_check (2, .filb, filb, 500); !?? definitely over-zealous checking  
446 0538 2 $block_check (2, .namb, namb, 508);  
447 0539 2 $block_check (2, .ctx, rmsb, 501);  
448 0540 2  
449 0541 2 ! Set the user buffer fields in the rab  
450 0542 2  
451 0543 2 rab [rab$l_ubf] = filb [filb$t_record_buffer]; ! buffer address  
452 0544 2  
453 0545 2 ! Read a single record from SYS$INPUT
```

```
454 0546 2 !
455 0547 3 status = (IF .rab [rab$v_bio] ! If we are doing block I/O to the file
456 0548 3 THEN
457 0549 4 BEGIN
458 0550 4 rab [rab$w_usz] = 512; ! Buffer size
459 0551 5 $read (rab = .rab) ! Physical uses block i/o
460 0552 4 END
461 0553 3 ELSE
462 0554 4 BEGIN
463 0555 4 rab [rab$w_usz] = filb$s_record_buffer; ! buffer size
464 0556 5 $get (rab = .rab) ! Everything else is record i/o
465 0557 4 END);
466 0558 2
467 0559 2 ! Since we are using locate mode, RMS can return a record which is larger than our buffer. We check the
468 0560 2 ! returned record length and simulate an RMS$_RTB error if we see such an animal.
469 0561 2
470 0562 2 IF .rab [rab$w_rsz] GTRU filb$s_record_buffer
471 0563 2 THEN
472 0564 3 BEGIN
473 0565 3 status = rms$_rtb; ! Status is record too big
474 0566 3 rab [rab$l_stv] = .rab [rab$w_rsz]; ! STV contains the record size for the signal
475 0567 2 END;
476 0568 2
477 0569 2 ! Signal any rms (or simulated rms) errors
478 0570 2
479 0571 2 IF NOT .status
480 0572 2 THEN
481 0573 3 BEGIN
482 0574 3
483 0575 3 filb [filb$a_record] = 0; ! Invalidate record descriptor
484 0576 3 filb [filb$l_record_len] = 0;
485 0577 3
486 0578 3 ! If the error is anything but end of file then signal
487 0579 3
488 0580 3 IF .status NEQ rms$_eof
489 0581 3 THEN
490 0582 4 BEGIN
491 0583 4 exch$util_file_error (exch$_readerr, .status, .fab, .rab [rab$l_stv]);
492 0584 4 RETURN .status; ! Return the RMS error
493 0585 4 END
494 0586 4
495 0587 4 ! Normal exit, return 0
496 0588 4
497 0589 3 ELSE
498 0590 3 RETURN false;
499 0591 2 END;
500 0592 2
501 0593 2 ! Return the address and length of the record
502 0594 2
503 0595 2 filb [filb$a_record] = .rab [rab$l_rbf];
504 0596 2 filb [filb$l_record_len] = .rab [rab$w_rsz];
505 0597 2
506 0598 2 RETURN true;
507 0599 2
508 0600 1 END;
```


				.EXTRN		SYS\$READ, SYS\$GET			
				007C	00000	.ENTRY	EXCH\$FIL11_GET, Save R2,R3,R4,R5,R6		0489
		56	00000000G	EF	9E	MOVAB	EXCH\$UTIL_BLOCK_CHECK, R6		
		53	04	AC	DO	MOVL	FILB, R3		0529
55	20	A3		10	C1	ADDL3	#16, 32(R3), R5		0531
54	20	A3		14	C1	ADDL3	#20, 32(R3), R4		0532
		52	035B00FA	8F	DO	MOVL	#56295674, R2		0537
		51	01F4	8F	3C	MOVZWL	#500, R1		
		50		53	DO	MOVL	R3, R0		
				66	16	JSB	EXCH\$UTIL_BLOCK_CHECK		
		52	010A00F7	8F	DO	MOVL	#17432823, R2		0538
		51	01FC	8F	3C	MOVZWL	#508, R1		
		50	18	A3	DO	MOVL	24(R3), R0		
				66	16	JSB	EXCH\$UTIL_BLOCK_CHECK		
		52	031600F6	8F	DO	MOVL	#51773686, R2		0539
		51	01F5	8F	3C	MOVZWL	#501, R1		
		50	20	A3	DO	MOVL	32(R3), R0		
				66	16	JSB	EXCH\$UTIL_BLOCK_CHECK		
		52		64	DO	MOVL	(R4), R2		0543
	24	A2	015A	C3	9E	MOVAB	346(R3), 36(R2)		
11	05	A2		03	E1	BBC	#3, 5(R2), 1\$		0547
	20	A2	0200	8F	BO	MOVW	#512, 32(R2)		0550
				52	DD	PUSHL	R2		0551
		00000000G	00	01	FB	CALLS	#1, SYS\$READ		
				0F	11	BRB	2\$		
	20	A2	0200	8F	BO	MOVW	#512, 32(R2)		0555
				52	DD	PUSHL	R2		0556
		00000000G	00	01	FB	CALLS	#1, SYS\$GET		
				50	DO	MOVL	R0, STATUS		
	0200	8F	22	A2	B1	CMPW	34(R2), #512		0562
				0C	1B	BLEQU	3\$		
		54	000181A8	8F	DO	MOVL	#98728, STATUS		0565
		0C	22	A2	3C	MOVZWL	34(R2), 12(R2)		0566
				54	E8	BLBS	STATUS, 4\$		0571
			42	A3	7C	CLRQ	66(R3)		0576
		0001827A	8F	54	D1	CPL	STATUS, #98938		0580
				26	13	BEQL	5\$		
			0C	A2	DD	PUSHL	12(R2)		0583
				65	DD	PUSHL	(R5)		
				54	DD	PUSHL	STATUS		
			00F810B0	8F	DD	PUSHL	#16257200		
		00000000G	EF	04	FB	CALLS	#4, EXCH\$UTIL_FILE_ERROR		
			50	54	DO	MOVL	STATUS, R0		0590
					04	RET			
	46	A3	28	A2	DO	MOVL	40(R2), 70(R3)		0595
	42	A3	22	A2	3C	MOVZWL	34(R2), 66(R3)		0596
		50		01	DO	MOVL	#1, R0		0598
					04	RET			
				50	D4	CLRL	R0		0600
				04	000C8	RET			

; Routine Size: 201 bytes, Routine Base: EXCH\$FIL11_CODE + 0377

```
510 0601 1 GLOBAL ROUTINE exch$fil11_open_file = %SBTTL 'exch$fil11_open_file'
511 0602 2 BEGIN
512 0603 2 ++
513 0604 2
514 0605 2 FUNCTIONAL DESCRIPTION:
515 0606 2
516 0607 2 Perform Files-11 volume specific open processing
517 0608 2
518 0609 2 INPUT/OUTPUT:
519 0610 2
520 0611 2 none
521 0612 2
522 0613 2 IMPLICIT INPUTS:
523 0614 2
524 0615 2 copy verb work area
525 0616 2
526 0617 2 OUTPUTS:
527 0618 2
528 0619 2 none
529 0620 2
530 0621 2 IMPLICIT OUTPUTS:
531 0622 2
532 0623 2 none
533 0624 2
534 0625 2 ROUTINE VALUE:
535 0626 2
536 0627 2 true if able to open a file, false otherwise
537 0628 2
538 0629 2 SIDE EFFECTS:
539 0630 2
540 0631 2 file is opened, copy work area modified
541 0632 2 --
542 0633 2
543 0634 2 $dbgtrc_prefix ('fil11_open_file> ');
544 0635 2
545 0636 2 LOCAL
546 0637 2 xab : $bblock [xab$c_fhclen], ! File header char xab so that we can read the size of the f
547 0638 2 status
548 0639 2 :
549 0640 2
550 0641 2 BIND
551 0642 2 copy = exch$a_gbl [excg$a_copy_work] : $ref_bblock,
552 0643 2 inp_filb = copy [copy$a_inp_filb] : $ref_bblock,
553 0644 2 out_filb = copy [copy$a_out_filb] : $ref_bblock,
554 0645 2 ctx = inp_filb [filb$a_context] : $ref_bblock,
555 0646 2 namb = inp_filb [filb$a_assoc_namb] : $ref_bblock
556 0647 2 :
557 0648 2
558 0649 2 $debug_print_lit ('entry');
559 0650 2
560 0651 2 $block_check_if_nonzero (2, .out_filb, filb, 577);
561 0652 2 $block_check (2, .inp_filb, filb, 502);
562 0653 2 $block_check (2, .namb, namb, 503);
563 0654 2 $logic_check (2, (.inp_filb [filb$a_assoc_volb] EQL 0), 136);
564 0655 2
565 0656 2 ! If the context block is null, then allocate an RMSB
566 0657 2 !
```

```

567 0658 2 IF .ctx EQL 0
568 0659 2 THEN
569 0660 2   ctx = exch$util_rmsb_allocate ( )           ! Get a fresh one
570 0661 2 ELSE
571 0662 2   $block_check (2, .ctx, rmsb, 504);       ! Check the old one
572 0663 2
573 0664 2 ! Use the RTL routine to find the next file matched by the input name, unless we are reopening in which case
574 0665 2 ! is ready
575 0666 2
576 0667 2 IF NOT .copy [copy$v_reopen_input]
577 0668 2 THEN
578 0669 2   BEGIN
579 P 0670 2   $trace_print_fao ('before find file: fullname !AS, inpname !AS, wcc !XL',
580 0671 2     namb [namb$q_fullname], inp_filb [filb$q_name_string], .inp_filb [filb$a_fil11_wcc]);
581 0672 2   status = lib$find_file (namb [namb$q_fullname], inp_filb [filb$q_name_string], inp_filb [filb$a_fil11_wc
582 P 0673 2   $trace_print_fao ('find file status !XL, fullname !AS, inpname !AS, wcc !XL',
583 0674 2     .status, namb [namb$q_fullname], inp_filb [filb$q_name_string], .inp_filb [filb$a_fil11_wcc]
584 0675 2   IF NOT .status
585 0676 2   THEN
586 0677 4     BEGIN
587 0678 4
588 0679 4     IF NOT .inp_filb [filb$v_files_found]           ! If no files were found, then scream and shout
589 0680 4     THEN
590 0681 4       $exch_signal (exch$_filenotfound, 1, namb [namb$q_fullname], .status);
591 0682 4
592 0683 4     IF .status EQL rms$_nmf                           ! rms$_nmf means that we are done with this filespec
593 0684 4     OR
594 0685 6     (BEGIN                                         ! Or if the error is severe
595 0686 6       BIND
596 0687 6         sb = status : $block;
597 0688 6         .sb [sts$v_severity] EQL sts$k_severe
598 0689 6       END)
599 0690 4     THEN
600 0691 4       status = 0;                                     ! 0 status terminates the outer loop
601 0692 4
602 0693 4     RETURN .status;
603 0694 4     END;
604 0695 2   END;
605 0696 2
606 0697 2 BEGIN
607 0698 2 BIND
608 0699 2   fab = ctx [rmsb$a_fab] : $ref_block;
609 0700 2   rab = ctx [rmsb$a_rab] : $ref_block;
610 0701 2   nam = ctx [rmsb$a_nam] : $ref_block;
611 0702 2   res = inp_filb [filb$q_name_string] : $desc_block;
612 0703 2
613 0704 2 ! Initialize the RMS structures
614 0705 2 !
615 P 0706 2 $fab_init (
616 P 0707 2   FAB = .fab,           ! Input file FAB
617 P 0708 2   FAC = (BRO,GET),    ! Get only, block I/O in case we can do things faster that
618 P 0709 2   FNA = .res [dsc$a_pointer], ! Set name addr
619 P 0710 2   FNS = .res [dsc$w_length], ! Set name size
620 P 0711 2   FOP = SQO,       ! Sequential only
621 P 0712 2   NAM = .nam,      ! Name block
622 P 0713 2   SHR = (GET,PUT,UPI), ! Allow other readers/writers
623 0714 2   XAB = xab);       ! A file header char xab so that we can read the file size

```

```

624 P 0715 3 $rab_init (
625 P 0716 3     RAB = .rab,           ! Input file RAB
626 P 0717 3     MBF = 2,           ! Multi-buffer count (MBC from default)
627 P 0718 3     RAC = SEQ,         ! Sequential only
628 P 0719 3     ROP = (LOC,RAH),   ! Locate mode, read ahead
629 P 0720 3     FAB = .fab);       ! FAB addr
630 P 0721 3 $nam_init (
631 P 0722 3     NAM = .nam,           ! File name block
632 P 0723 3     RSA = .ctx [rmsb$a_rsbuf], ! Result name addr
633 P 0724 3     RSS = nam$c_maxrss,   ! Result name size
634 P 0725 3     ESA = .ctx [rmsb$a_esbuf], ! Expanded name addr
635 P 0726 3     ESS = nam$c_maxrss);   ! Expanded name size
636 P 0727 3 $xabfhc_init (
637 P 0728 3     XAB = xab);           ! File header char xab so that we can read the file size
638 P 0729 3                               ! RMS will fill it in when we open
639 P 0730 3 ! If this is a block transfer mode read, set the block i/o bit
640 P 0731 3 !
641 P 0732 5 rab [rab$v_bio] = ((.inp_filb [filb$b_transfer_mode] EQL filb$k_xfrm_block)
642 P 0733 3     OR
643 P 0734 3     (IF .out_filb EQL 0
644 P 0735 3     THEN
645 P 0736 3     0
646 P 0737 3     ELSE
647 P 0738 3     .out_filb [filb$b_transfer_mode] EQL filb$k_xfrm_block));
648 P 0739 3 !
649 P 0740 3 ! Open and connect to the file
650 P 0741 3 !
651 P 0742 3 $trace_print_fao ('opening, fab=!XL', .fab);
652 P 0743 4 IF NOT (status = $open (fab = .fab))
653 P 0744 3 THEN
654 P 0745 4 BEGIN
655 P 0746 4     exch$util_file_error (exch$_openin, .status, .fab, .fab [fab$l_stv]);
656 P 0747 4     RETURN .status;
657 P 0748 3 END;
658 P 0749 4 IF NOT (status = $connect (rab = .rab))
659 P 0750 3 THEN
660 P 0751 4 BEGIN
661 P 0752 4     exch$util_file_error (exch$_openin, .status, .fab, .rab [rab$l_stv]);
662 P 0753 4     $close (fab = .fab);
663 P 0754 4     RETURN .status;
664 P 0755 3 END;
665 P 0756 3 !
666 P 0757 3 !
667 P 0758 3 ! Create the result name string in the filb
668 P 0759 3 !
669 P 0760 3 $logic_check (2, ((.nam [nam$b_rsl] LEQU filb$s_result_name) AND (.nam [nam$b_rsl] GTRU 0)), 137);
670 P 0761 3 inp_filb [filb$l_result_name_len] = .nam [nam$b_rsl];
671 P 0762 3 CH$COPY (.nam [nam$b_rsl], .nam [nam$l_rsa], 0, filb$s_result_name, inp_filb [filb$t_result_name]);
672 P 0763 3 !
673 P 0764 3 $trace_print_fao ('Found "!AF"', .inp_filb [filb$l_result_name_len], inp_filb [filb$t_result_name]);
674 P 0765 3 !
675 P 0766 3 ! Define a record stream for this file
676 P 0767 3 !
677 P 0768 3 inp_filb [filb$a_record] = 0;           ! No valid record or length
678 P 0769 3 inp_filb [filb$l_record_len] = 0;
679 P 0770 3 inp_filb [filb$v_files_found] = true;   ! Found a file using this filb
680 P 0771 3 inp_filb [filb$l_block_count] = .xab [xab$l_ebk] - ! Put the file size in the filb where any routine ca

```


				0D	11	00183		BRB	11\$			
				50	D4	00185	10\$:	CLRL	R0			0738
	52		6E	29	C1	00187		ADDL3	#41, (SP), R2			
			01	62	91	00188		CMPB	(R2), #1			
				02	12	0018E		BNEQ	11\$			
				50	D6	00190		INCL	R0			
	53		50	51	89	00192	11\$:	BISB3	R1, R0, R3			0734
05	A6	01	03	53	F0	00196		INSV	R3, #3, #1, 5(R6)			
				57	DD	0019C		PUSHL	R7			0743
		00000000G	00	01	FB	0019E		CALLS	#1, SYS\$OPEN			
			5B	50	D0	001A5		MOVL	R0, STATUS			
			16	5B	EB	001AB		BLBS	STATUS, 12\$			
				A7	DD	001AB	0C	PUSHL	12(R7)			0746
				57	DD	001AE		PUSHL	R7			
				5B	DD	001B0		PUSHL	STATUS			
		00000000G	EF	00F81098	8F	DD	001B2	PUSHL	#16257176			
					04	FB	001B8	CALLS	#4, EXCH\$UTIL_FILE_ERROR			
				2C	11	001BF		BRB	13\$			0747
		00000000G	00	56	DD	001C1	12\$:	PUSHL	R6			0749
			5B	01	FB	001C3		CALLS	#1, SYS\$CONNECT			
			21	50	D0	001CA		MOVL	R0, STATUS			
				5B	EB	001CD	0C	BLBS	STATUS, 14\$			
				A6	DD	001D0		PUSHL	12(R6)			0752
				57	DD	001D3		PUSHL	R7			
				5B	DD	001D5		FUSHL	STATUS			
		00000000G	EF	00F81098	8F	DD	001D7	PUSHL	#16257176			
					04	FB	001DD	CALLS	#4, EXCH\$UTIL_FILE_ERROR			
				57	DD	001E4		PUSHL	R7			0753
		00000000C	00	01	FB	001E6		CALLS	#1, SYS\$CLOSE			
			50	5B	D0	001ED	13\$:	MOVL	STATUS, R0			0754
					04	001F0		RET				
			52	03	A9	9A	001F1	14\$:	MOVZBL	3(R9), R2		0760
					13	12	001F5		BNEQ	15\$		
			7E	89	9F	9A	001F7		MOVZBL	#137, -(SP)		
					01	DD	001FB		PUSHL	#1		
					8F	DD	001FD		PUSHL	#EXCH\$BADLOGIC		
		00000000G	00	03	FB	00203		CALLS	#3, LIB\$STOP			
			3A	52	D0	0020A	15\$:	MOVL	R2, 58(R8)			0761
0100	8F	00	04	52	2C	0020E		MOVC5	R2, @4(R9), #0, #256, 90(R8)			0762
					A8	00216						
				5A	A8	7C	00218		CLRQ	66(R8)		0769
				42	A8	88	0021B		BISB2	#8, 43(R8)		0770
		2B	A8		AE	B5	0021F		TSTW	XAB+20		0772
				18	04	13	00222		BEQL	16\$		
					50	D4	00224		CLRL	R0		
					03	11	00226		BRB	17\$		
			50	01	D0	00228	16\$:	MOVL	#1, R0			
	3E	A8	14	50	C3	0022B	17\$:	SUBL3	R0, XAB+16, 62(R8)			
				24	A7	D4	00231		CLRL	36(R7)		0774
			4A	A8	CF	9E	00234		MOVAB	EXCH\$FIL11_CLOSE_FILE, 74(R8)		0778
				56	A8	D4	0023A		CLRL	86(R8)		0779
			52	A8	CF	9E	0023D		MOVAB	EXCH\$FIL11_GET, 82(R8)		0780
				50	01	D0	00243		MOVL	#1, R0		0784
					04	00246		RET				0786

; Routine Size: 583 bytes, Routine Base: EXCH\$FIL11_CODE + 0440

EXCH\$FIL11
V04-000

Files-11 volume specific routines
exch\$fil11_open_file

M 5
16-Sep-1984 00:56:31
14-Sep-1984 12:29:04

YAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCFIL11.B32;1

Page 22
(6)

EXC
V04

: R


```
697 0787 1 GLOBAL ROUTINE exch$fil11_put = %SBTTL 'exch$fil11_put'
698 0788 2 BEGIN
699 0789 2 ++
700 0790 2
701 0791 2 FUNCTIONAL DESCRIPTION:
702 0792 2
703 0793 2     Add the next record to the file
704 0794 2
705 0795 2 INPUTS:
706 0796 2
707 0797 2     none
708 0798 2
709 0799 2 IMPLICIT INPUTS:
710 0800 2
711 0801 2     copy [copy$a_out_filb] - out_filb - pointer to filb for an open Files-11 output file
712 0802 2     copy [copy$a_inp_filb] - inp_filb - pointer to the input filb containing the record info
713 0803 2     inp_filb [filb$l_record len] - len - length of the record
714 0804 2     inp_filb [filb$a_record] - buf - address of the record
715 0805 2
716 0806 2 OUTPUTS:
717 0807 2
718 0808 2     none
719 0809 2
720 0810 2 IMPLICIT OUTPUTS:
721 0811 2
722 0812 2     out_filb will get updated
723 0813 2
724 0814 2 ROUTINE VALUE:
725 0815 2
726 0816 2     true if success, false if any error
727 0817 2
728 0818 2 SIDE EFFECTS:
729 0819 2
730 0820 2     error conditions will be signaled
731 0821 2 --
732 0822 2
733 0823 2 $dbgtrc_prefix ('fil11_put> ');
734 0824 2
735 0825 2 LOCAL
736 0826 2     status
737 0827 2 ;
738 0828 2
739 0829 2 BIND
740 0830 2     copy = exch$a_gbl [excg$a_copy work]: $ref_bblock, ! COPY verb work area
741 0831 2     out_filb = copy [copy$a_out_filb] : $ref_bblock, ! pointer to filb for an open Files-11 output file
742 0832 2     inp_filb = copy [copy$a_inp_filb] : $ref_bblock, ! pointer to the input filb with the record info
743 0833 2     len = inp_filb [filb$l_record len], ! length of the record
744 0834 2     buf = inp_filb [filb$a_record], ! address of the record
745 0835 2     ctx = out_filb [filb$a_context] : $ref_bblock, ! output file context block
746 0836 2     namb = out_filb [filb$a_assoc_namb] : $ref_bblock, ! associated output namb structure
747 0837 2     fab = ctx [rmsb$a_fab] : $ref_bblock, ! RMS FAB for the file
748 0838 2     rab = ctx [rmsb$a_rab] : $ref_bblock, ! RMS RAB
749 0839 2 ;
750 0840 2
751 0841 2 $debug_print_lit ('entry');
752 0842 2
753 0843 2 $block_check (2, .out_filb, filb, 505); !?? definitely over-zealous checking
```

```

754 0844 2 $block_check (2, .inp_filb, filb, 526);
755 0845 2 $block_check (2, .namB, namb, 506);
756 0846 2 $block_check (2, .ctx, rmsb, 507);
757 0847 2
758 0848 2 ! Set the record buffer fields in the rab
759 0849 2
760 0850 2 IF .fab [fab$b_rfm] EQL fab$c_fix ! If we have fixed-length output
761 0851 2 AND
762 0852 2 .fab [fab$w_mrs] NEQ .len ! And the input length isn't correct
763 0853 2 THEN
764 0854 2 BEGIN
765 0855 2 CH$COPY (.len, .buf, .inp_filb [filb$b_pad char], .fab [fab$w_mrs], out_filb [filb$t_record_buffer]);
766 0856 2 rab [rab$l_rbf] = out_filb [filb$t_record_buffer];
767 0857 2 rab [rab$w_rsz] = .fab [fab$w_mrs];
768 0858 2 END
769 0859 2 ELSE ! Otherwise just point the rab at the record
770 0860 2 BEGIN
771 0861 2 rab [rab$l_rbf] = .buf; ! buffer address
772 0862 2 rab [rab$w_rsz] = .len; ! buffer size
773 0863 2 END;
774 0864 2
775 0865 2 ! Write a single record to the output filb
776 0866 2
777 0867 2 IF NOT (status = $put (rab = .rab))
778 0868 2 THEN
779 0869 2 BEGIN
780 0870 2
781 0871 2 exch$util_file_error (exch$_writeerr, .status, .fab, .rab [rab$l_stv]);
782 0872 2 RETURN .status;
783 0873 2
784 0874 2 END;
785 0875 2
786 0876 2 RETURN true;
787 0877 1 END;

```

					.EXTRN	SYSSPUT		
				03FC 00000	.ENTRY	EXCH\$FIL11_PUT, Save R2,R3,R4,R5,R6,R7,R8,-	0787	
						R9		
					MOVAB	EXCH\$UTIL_BLOCK_CHECK, R9		
50	00000000G	59	00000000G	EF 9E 00002	ADDL3	#4, EXCH\$A_GBL, R0	0830	
51		60	00000044	8F C1 00009	ADDL3	#68, (R0), R1	0831	
50		60		3C C1 00019	ADDL3	#60, (R0), R0	0832	
		53		60 D0 0001D	MOVL	(R0), R3	0833	
		56		61 D0 00020	MOVL	(R1), R6	0835	
54	20	A6		10 C1 00023	ADDL3	#16, 32(R6), R4	0837	
58	20	A6		14 C1 00028	ADDL3	#20, 32(R6), R8	0838	
		52	035B00FA	8F D0 0002D	MOVL	#56295674, R2	0843	
		51	01F9	8F 3C 00034	MOVZWL	#505, R1		
		50		56 D0 00039	MOVL	R6, R0		
				69 16 0003C	JSB	EXCH\$UTIL_BLOCK_CHECK		
		52	035B00FA	8F D0 0003E	MOVL	#56295674, R2	0844	
		51	020E	8F 3C 00045	MOVZWL	#526, R1		
		50		53 D0 0004A	MOVL	R3, R0		
				69 16 0004D	JSB	EXCH\$UTIL_BLOCK_CHECK		

				52	010A00F7	8F	DO	0004F	MOVL	#17432823, R2	0845
				51	01FA	8F	3C	0C056	MOVZWL	#506, R1	
				50	18	A6	DO	0005B	MOVL	24(R6), R0	
						69	16	0005F	JSB	EXCH\$UTIL_BLOCK_CHECK	
				52	031600F6	8F	DO	00061	MOVL	#51773686, R2	0846
				51	01FB	8F	3C	00068	MOVZWL	#507, R1	
				50	20	A6	DO	0006D	MOVL	32(R6), R0	
						69	16	00071	JSB	EXCH\$UTIL_BLOCK_CHECK	
				57		64	DO	00073	MOVL	(R4), R7	0850
				01	1F	A7	91	00076	CMPB	3*(R7), #1	
						25	12	0007A	BNEQ	1\$	
42	A3	36	A7	10		00	ED	0007C	CMPZV	#0, #16, 54(R7), 66(R3)	0852
						1C	13	00083	BEQL	1\$	
36	A7	39	A3	46	B3	42	A3	2C	MOVCS	66(R3), #70(R3), 57(R3), 54(R7), 346(R6)	0855
							C6	0008E			
				52		68	DO	00091	MOVL	(R8), R2	0856
				28	A2	015A	C6	9E	MOVAB	346(R6), 40(R2)	
				22	A2	36	A7	B0	MOVW	54(R7), 34(R2)	0857
							0D	11	BRB	2\$	0850
				52		68	DO	000A1	MOVL	(R8), R2	0861
				28	A2	46	A3	DO	MOVL	70(R3), 40(R2)	
				22	A2	42	A3	B0	MOVW	66(R3), 34(R2)	0862
							52	DD	PUSHL	R2	0867
				00000000G	00	01	FB	000B0	CALLS	#1, SYSSPUT	
					53	50	DO	000B7	MOVL	R0, STATUS	
					18	53	E8	000BA	BLBS	STATUS, 3\$	
						0C	A2	DD	PUSHL	12(R2)	0871
						0088	8F	BB	PUSHR	#*M<R3, R7>	
						00F810D0	8F	DD	PUSHL	#16257232	
				00000000G	EF	04	FB	000CA	CALLS	#4, EXCH\$UTIL_FILE_ERROR	0872
					50	53	DO	000D1	MOVL	STATUS, R0	
						04	000D4	RET			
				50		01	DO	000D5	MOVL	#1, R0	0876
						04	000D8	RET			0877

: Routine Size: 217 bytes, Routine Base: EXCH\$FIL11_CODE + 0687

EXCH\$FIL11
V04-000

Files-11 volume specific routines
exch\$fil11_put

D 6
16-Sep-1984 00:56:31
14-Sep-1984 12:29:04

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCFIL11.032;1

Page 26
(8)

EXC
V04

: 789 0878 1 END
: 790 0879 0 ELUDOM

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name Bytes Attributes
EXCH\$FIL11_CODE 1888 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	117	0	1000	00:01.8
_\$255\$DUA28:[EXCHNG.OBJ]EXCLIB.L32;1	1151	86	7	79	00:01.3

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:EXCFIL11/OBJ=OBJ\$:EXCFIL11 MSRCS:EXCFIL11/UPDATE=(ENHS:EXCFIL11)

: Size: 1888 code + 0 data bytes
: Run Time: 00:40.1
: Elapsed Time: 02:12.9
: Lines/CPU Min: 1315
: Lexemes/CPU-Min: 30904
: Memory Used: 282 pages
: Compilation Complete

EXCFIL11
LIS

EXCINIT
LIS

EXCLB
LIS

EXCTO
LIS