


```

PPPPPPPP      AAAAAA      SSSSSSSS      000000      PPPPPPPP      EEEEEEEEEEE      NN      NN      222222
PPPPPPPP      AAAAAA      SSSSSSSS      000000      PPPPPPPP      EEEEEEEEEEE      NN      NN      222222
PP      PP      AA      AA      SS      00      00      FP      PP      EE      NN      NN      22      22
PP      PP      AA      AA      SS      00      00      PP      PP      EE      NN      NN      22      22
PP      PP      AA      AA      SS      00      00      PP      PP      EE      NNNN      NN      22      22
PP      PP      AA      AA      SS      00      00      PP      PP      EE      NNNN      NN      22      22
PPPPPPPP      AA      AA      SSSSSS      00      00      PPPPPPPP      EEEEEEEEE      NN      NN      NN      22      22
PPPPPPPP      AA      AA      SSSSSS      00      00      PPPPPPPP      EEEEEEEEE      NN      NN      NN      22      22
PP      AAAAAAAAAA      SS      00      00      PP      EE      NN      NNNN      NN      22      22
PP      AAAAAAAAAA      SS      00      00      PP      EE      NN      NNNN      NN      22      22
PP      AA      AA      SS      00      00      PP      EE      NN      NN      NN      22      22
PP      AA      AA      SS      00      00      PP      EE      NN      NN      NN      22      22
PP      AA      AA      SSSSSSSS      000000      PP      EEEEEEEEEEE      NN      NN      2222222222      .....
PP      AA      AA      SSSSSSSS      000000      PP      EEEEEEEEEEE      NN      NN      2222222222      .....

```

```

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
LLLLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLLLL      IIIIII      SSSSSSSS

```

```

1 0001 0 MODULE PASSOPEN2 ( %TITLE 'OPEN procedure'
2 0002 0 IDENT = '1-015' ! File: PASOPEN2.B32 Edit: SBL1015
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 **
31 0031 1 FACILITY: Pascal Language Support
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 This module contains PASSOPEN2 and PASS$OPEN, which open a file.
36 0036 1
37 0037 1 ENVIRONMENT: User mode - AST reentrant
38 0038 1
39 0039 1 AUTHOR: Steven B. Lionel, CREATION DATE: 1-April-1981
40 0040 1
41 0041 1 MODIFIED BY:
42 0042 1
43 0043 1 1-001 - Original. SBL 1-April-1981
44 0044 1 1-002 - Give ACCMETINC message if ACCESS METHOD:=DIRECT and device
45 0045 1 is not a random-access device. Don't store PFV address in enable
46 0046 1 argument until the FCB pointer is stored. SBL 21-June-1982
47 0047 1 1-003 - In CHECK_KEY_XABS, don't assume that XAB_SUM is still linked
48 0048 1 to the FAB. SBL 8-July-1982
49 0049 1 1-004 - In CHECK_KEY_XABS, allow any defined, but unrecognized key type.
50 0050 1 Don't enable prompting on spooled terminals. SBL 20-July-1982
51 0051 1 1-005 - Copy DEV$V_RND to FCB$V_RND. SBL 29-July-1982
52 0052 1 1-006 - Store DEV$V_FOD instead of DEV$V_RND. Keep DEV$V_RND for
53 0053 1 ACCESS METHOD:=DIRECT test. SBL 29-July-1982
54 0054 1 1-007 - Call PASS$REMOVE FILE from OPEN_HANDLER to deallocate FCB and
55 0055 1 store STATUS. QAR FT3-002 SBL 30-Aug-1982
56 0056 1 1-008 - Use FAB$W_BLS on non-disk-or-terminal devices to determine file's
57 0057 1 recordsize. Correct RECORD_LENGTH check. Remove VMS V2 variant

```

```
.. 58      0058 1 | code. SBL 27-Sept-1982
.. 59      0059 1 | 1-009 - Remove special case for "resultant string overflow" error. Don't
.. 60      0060 1 | enable prompting for an input-only file. SBL 9-Dec-1982
.. 61      0061 1 | 1-010 - Add new bit FCBSV INITIATE_PROMPT. This is used instead of
.. 62      0062 1 | FCBSV PROMPT_ENABLE to determine whether a look-ahead of that file
.. 63      0063 1 | should initiate prompts on other files. This allows GETs from
.. 64      0064 1 | readonly files to initiate prompting on other files. SBL 15-Dec-1982
.. 65      0065 1 | 1-011 - If reopening for prompting fails because of RMS$_RAT, go back to
.. 66      0066 1 | no prompting. SBL 5-Jan-1983
.. 67      0067 1 | 1-012 - Allow "quiet reopen for READONLY" to occur for UNKNOWN.
.. 68      0068 1 | SBL 10-Jan-1983
.. 69      0069 1 | 1-013 - Remove PROMPT xxx keyword recognition, since this was never
.. 70      0070 1 | supported. Change method of recognizing duplicate keywords to one
.. 71      0071 1 | that is extensible past 32 codes. Add stream recordtype support
.. 72      0072 1 | in advance of compiler support. SBL 17-Aug-1983
.. 73      0073 1 | 1-014 - Make KEYWD_NAME_TABLE global for use by PASSCLOSE2. SBL 19-Aug-1983
.. 74      0074 1 | 1-015 - Also set FABS$_OPI if SHARING:=READONLY and user wants write access
.. 75      0075 1 | to sequential file. SBL 24-Feb-1984
.. 76      0076 1 | --
.. 77      0077 1 |
```

```

79 0078 1 %SBTTL 'Declarations'
80 0079 1
81 0080 1 PROLOGUE DEFINITIONS:
82 0081 1
83 0082 1
84 0083 1 REQUIRE 'RTLIN:PASPROLOG';           ! Linkages, externals, PSECTs, structures
85 0147 1
86 0148 1
87 0149 1 Linkage definitions for internal procedures.
88 0150 1
89 0151 1
90 0152 1 LINKAGE
91 0153 1     CALL FILL_KEY_XABS =
92 0154 1     CALL (REGISTER=6, REGISTER=7),
93 0155 1     CALL CHECK_KEY_XABS =
94 0156 1     CALL (REGISTER=6, REGISTER=7);
95 0157 1
96 0158 1
97 0159 1 TABLE OF CONTENTS:
98 0160 1
99 0161 1
100 0162 1 FORWARD ROUTINE
101 0163 1     PASSOPEN2: NOVALUE,           ! Called by compiled code
102 0164 1     PASS$OPEN: CALL OPEN NOVALUE, ! Default OPEN called by RTL
103 0165 1     PASS$OPEN_IMPLICIT: JSB OPEN_IMPLICIT NOVALUE, ! Open INPUT and OUTPUT
104 0166 1     FILL_KEY_XABS: CALL FILL_KEY_XABS NOVALUE, ! Fill in KEY XABS
105 0167 1     CHECK_KEY_XABS: CALL CHECK_KEY_XABS NOVALUE, ! Check KEY XABS
106 0168 1     OPEN_HANDLER,           ! Condition handler for PASS$OPEN
107 0169 1     EXIT_HANDLER: NOVALUE;    ! Established by PASS$OPEN
108 0170 1
109 0171 1
110 0172 1 MACROS:
111 0173 1
112 0174 1     NONE
113 0175 1
114 0176 1 EQUATED SYMBOLS:
115 0177 1
116 0178 1     NONE
117 0179 1
118 0180 1 FIELDS:
119 0181 1
120 0182 1     NONE
121 0183 1
122 0184 1 OWN STORAGE:
123 0185 1
124 0186 1
125 0187 1
126 0188 1 Declare a longword which is used as a flag to indicate whether or not
127 0189 1 an exit handler has been declared.
128 0190 1
129 0191 1
130 0192 1 OWN
131 0193 1     EXITH_DECLARED: INITIAL (0);
132 0194 1
133 0195 1
134 0196 1 Declare two flags which indicate if the files INPUT and OUTPUT have ever
135 0197 1 been opened. If set, PASS$LOOK_AHEAD will not implicitly open them.

```

PASSOPEN2
1-015

OPEN procedure
Declarations

M 10
16-Sep-1984 01:46:15
14-Sep-1984 12:51:41

VAX-11 BLISS-32 V4.0-742
[PASRTL.SRC]PASOPEN2.B32;1

Page 4
(2)

```
: 136      0198 1  !-
: 137      0199 1
: 138      0200 1 GLOBAL
: 139      0201 1      PASS$GV_INPUT_OPENED: BYTE INITIAL (0),
: 140      0202 1      PASS$GV_OUTPUT_OPENED: BYTE INITIAL (0);
```

```

142 0203 1 %SBTTL 'PASSOPEN2 - Open a file'
143 0204 1 GLOBAL ROUTINE PASSOPEN2 (           ! Open a file
144 0205 1     PFV: REF $PASSPFV_FILE_VARIABLE,   ! File variable
145 0206 1     KEYWORDS: VECTOR [, LONG]        ! Keywords and specifiers
146 0207 1 ): NOVALUE =
147 0208 1
148 0209 1 --
149 0210 1 FUNCTIONAL DESCRIPTION:
150 0211 1
151 0212 1     This routine opens a Pascal file. This entry is called from the
152 0213 1     compiled code for the OPEN procedure.
153 0214 1
154 0215 1 CALLING SEQUENCE:
155 0216 1
156 0217 1     CALL PASSOPEN2 (PFV.mr.r [, keyword.rz.v [, keyword.rz.v ...]])
157 0218 1
158 0219 1 FORMAL PARAMETERS:
159 0220 1
160 0221 1     PFV           - The Pascal File Variable (PFV) passed by reference.
161 0222 1                 The structure of the PFV is defined in PASPFV.REQ.
162 0223 1
163 0224 1     keyword      - Zero or more values representing keywords in the
164 0225 1                 OPEN procedure. Some keywords are followed by
165 0226 1                 associated arguments and values. See PASOPNKEY.REQ
166 0227 1                 for the keyword value definitions.
167 0228 1
168 0229 1                 If the keyword ERROR is specified, it must be first
169 0230 1                 in the list, otherwise it will be ignored.
170 0231 1
171 0232 1 IMPLICIT INPUTS:
172 0233 1
173 0234 1     NONE
174 0235 1
175 0236 1 IMPLICIT OUTPUTS:
176 0237 1
177 0238 1     NONE
178 0239 1
179 0240 1 COMPLETION STATUS:
180 0241 1
181 0242 1     NONE
182 0243 1
183 0244 1 SIDE EFFECTS:
184 0245 1
185 0246 1     See PASS$OPEN.
186 0247 1
187 0248 1 SIGNALLED ERRORS:
188 0249 1
189 0250 1     See PASS$OPEN.
190 0251 1
191 0252 1 --
192 0253 1
193 0254 2 BEGIN
194 0255 2
195 0256 2 LOCAL
196 0257 2     FCB: REF $PASSFCB_CONTROL_BLOCK,   ! Control block
197 0258 2     PFV_ADDR: VOLATILE,               ! Enable argument
198 0259 2     UNWIND_ACT: VOLATILE,           ! Enable argument

```

```

: 199      0260 2      ERROR_ADDR: VOLATILE;          . Enable argument
: 200      0261 2
: 201      0262 2      BUILTIN
: 202      0263 2      ACTUALCOUNT,
: 203      0264 2      AP,
: 204      0265 2      CALLG;
: 205      0266 2
: 206      0267 2      ENABLE
: 207      0268 2      PASS$IO_HANDLER (PFV_ADDR, UNWIND_ACT, ERROR_ADDR);
: 208      0269 2
: 209      0270 2      !+
: 210      0271 2      ! Set PFV_ADDR enable argument.
: 211      0272 2      !-
: 212      0273 2
: 213      0274 2      PFV_ADDR = PFV [PFV$R_PFV];
: 214      0275 2
: 215      0276 2      !+
: 216      0277 2      ! If ERROR is present, it is the first keyword in the list. See
: 217      0278 2      ! if that is true. If so, set enable argument to the return address.
: 218      0279 2      !-
: 219      0280 2
: 220      0281 2      IF ACTUALCOUNT () GEQU 3      ! At least 3 needed for PFV, keyword, value
: 221      0282 2      THEN
: 222      0283 2      IF .KEYWORDS [0] EQL PASS$K_ERROR      ! Is it the ERROR keyword?
: 223      0284 2      THEN
: 224      0285 2      ERROR_ADDR = .KEYWORDS [1];
: 225      0286 2
: 226      0287 2      !+
: 227      0288 2      ! Validate and lock PFV.
: 228      0289 2      !-
: 229      0290 2
: 230      0291 2      PASS$VALIDATE_PFV (PFV [PFV$R_PFV]; FCB);
: 231      0292 2
: 232      0293 2      !+
: 233      0294 2      ! Set unwind action to unlock file.
: 234      0295 2      !-
: 235      0296 2
: 236      0297 2      UNWIND_ACT = PASS$K_UNWIND_UNLOCK;
: 237      0298 2
: 238      0299 2      !+
: 239      0300 2      ! If the file is already open, it's an error.
: 240      0301 2      !-
: 241      0302 2
: 242      0303 2      IF .PFV [PFV$V_OPEN]
: 243      0304 2      THEN
: 244      0305 2      $PASS$IO_ERROR (PASS$_FILALROPE,0);      ! File already open
: 245      0306 2
: 246      0307 2      !+
: 247      0308 2      ! Do the OPEN.
: 248      0309 2      !-
: 249      0310 2
: 250      0311 2      CALLG (.AP, PASS$OPEN);
: 251      0312 2
: 252      0313 2      !+
: 253      0314 2      ! Indicate successful completion
: 254      0315 2      ! Unlock the file variable.
: 255      0316 2      !-

```



```

: 256      0317 2
: 257      0318 2      FCB [FCB$SL_STATUS] = 0;
: 258      0319 2      PFV [PFV$V_LOCK] = 0;
: 259      0320 2
: 260      0321 2      RETURN;
: 261      0322 2
: 262      0323 1      END;

```

. End of routine PASSOPEN2

```

.TITLE PASSOPEN2 OPEN procedure
.IDENT \1-015\

```

.PSECT _PASSDATA,NOEXE, PIC,2

```

00000000 00000 EXITH_DECLARED:
                .LONG 0
00 00004 PASS$GV_INPUT_OPENED::
                .BYTE 0
00 00005 PASS$GV_OUTPUT_OPENED::
                .BYTE 0

```

```

.EXTRN PASSOPEN2, PASS$OPEN
.EXTRN PASS$OPEN_IMPLICIT
.EXTRN PASS$IO_HANDLER
.EXTRN PASS$VALIDATE_PFV
.EXTRN PASS$SIGNAL, PASSK_FILALROPE

```

.PSECT _PASSCODE,NOWRT, SHR, PIC,2

```

                00C4 00000 .ENTRY PASSOPEN2, Save R2,R6,R7 ; 0204
                08 C2 00002  SUBL2 #8, SP ;
                7E D4 00005  CLRL ERROR_ADDR ; 0254
                04 AE 7C 00007  CLRQ UNWIND_ACT
                6D 0047 CF DE 0000A  MOVAL 3$, (FP)
                56 04 AC D0 0000F  MOVL PFV, R6 ; 0274
                08 AE 56 D0 00013  MOVL R6, PFV_ADDR
                03 6C 91 00017  CMPB (AP), #3 ; 0281
                0A 1F 0001A  BLSSU 1$
                19 08 AC D1 0001C  CMPL KEYWORDS, #25 ; 0283
                04 12 00020  BNEQ 1$
                6E 0C AC D0 00022  MOVL KEYWORDS+4, ERROR_ADDR ; 0285
                00000000G 00 16 00026 1$: JSB PASS$VALIDATE_PFV ; 0291
                04 AE 01 D0 0002C  MOVL #1, UNWIND_ACT ; 0297
                OE 07 A6 05 E1 00030  BBC #5, 7(R6), -2$ ; 0303
                7E D4 00035  CLRL -(SP) ; 0305
                00G 8F 9A 00037  MOVZBL #PASSK_FILALROPE, -(SP)
                00 02 FB 0003B  CALLS #2, PASS$SIGNAL
                04 00042  RET
                0000V CF 6C FA 00043 2$: CALLG (AP), PASS$OPEN ; 0311
                D4 A7 D4 00048  CLRL -44(FCB) ; 0318
                50 04 AC D0 0004B  MOVL PFV, R0 ; 0319
                07 A0 80 8F 8A 0004F  BICB2 #128, 7(R0)
                04 00054  RET ; 0323
                0000 00055 3$: .WORD Save nothing ; 0254
                50 08 AC D0 00057  MOVL 8(AP), R0
                50 04 A0 D0 0005B  MOVL 4(R0), R0
                F4 A0 9F 0005F  PUSHAB ERROR_ADDR

```

PASSOPEN2
1-015

OPEN procedure
PASSOPEN2 - Open a file

D 11
16-Sep-1984 01:46:15
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742
[PASRTL.SRC]PASOPEN2.B32;1

Page 8
(3)

		F8	A0	9F	00062	PUSHAB	UNWIND_ACT	
		FC	A0	9F	00065	PUSHAB	PFV_ADDR	
			03	DD	00068	PUSHL	#3	
			5E	DD	0006A	PUSHL	SP	
	00000000G	7E	04	AC	7D	0006C	MOVQ	4(AP), -(SP)
		00		03	FB	00070	CALLS	#3, PASS\$IO_HANDLER
				04	00077	RET		

; Routine Size: 120 bytes, Routine Base: _PASS\$CODE + 0000

: 263 0324 1
: 264 0325 1 !<BLF/PAGE>

```

266 0326 1 %SBTTI 'PASS$OPEN - Open a file'
267 0327 1 GLOBAL ROUTINE PASS$OPEN (           | Open a file
268 0328 1     IN PFV: REF $PASS$PFV FILE_VARIABLE, | File variable
269 0329 1     KEYWORDS: VECTOR [, [ONG]);         | Keywords and specifiers
270 0330 1     FCB: REF $PASS$FCB CONTROL_BLOCK   | File control block
271 0331 1   ): CALL_OPEN NOVALUE =
272 0332 1
273 0333 1 ++
274 0334 1 FUNCTIONAL DESCRIPTION:
275 0335 1
276 0336 1     This procedure opens the Pascal file specified by "PFV"
277 0337 1     using the attributes possibly supplied as "keyword". PASS$OPEN
278 0338 1     is to be called from other RTL routines only after they have tested
279 0339 1     and set PFV$V_LOCK in "PFV".
280 0340 1
281 0341 1 CALLING SEQUENCE:
282 0342 1
283 0343 1     CALL PASS$OPEN (PFV.mr.r [, keyword.rz.v [, keyword.rz.v ...]])
284 0344 1
285 0345 1 FORMAL PARAMETERS:
286 0346 1
287 0347 1     PFV           - The Pascal File Variable (PFV) passed by reference.
288 0348 1                   The structure of the PFV is defined in PASPFV.REQ.
289 0349 1
290 0350 1     keyword       - Zero or more values representing keywords in the
291 0351 1                   OPEN procedure. Some keywords are followed by
292 0352 1                   associated arguments and values. See PASOPNKEY.REQ
293 0353 1                   for the keyword value definitions.
294 0354 1
295 0355 1 IMPLICIT INPUTS:
296 0356 1
297 0357 1     The PFV$V_LOCK bit is assumed to have been tested-and-set by the caller.
298 0358 1
299 0359 1 IMPLICIT OUTPUTS:
300 0360 1
301 0361 1     FCB           - The address of the File Control Block for the file
302 0362 1                   is returned in the register designated by the linkage.
303 0363 1
304 0364 1     PFV           - If the RELPFD or RELBUF bits are set in the PFV, those
305 0365 1                   self-relative addresses are resolved.
306 0366 1
307 0367 1 ROUTINE VALUE:
308 0368 1
309 0369 1     NONE
310 0370 1
311 0371 1 SIDE EFFECTS:
312 0372 1
313 0373 1     Declares an exit handler, if not already declared.
314 0374 1
315 0375 1     The file is opened. Internal RTL and RMS structures are allocated
316 0376 1     and initialized. The PFV is set to indicate an opened file.
317 0377 1
318 0378 1 SIGNALLED ERRORS:
319 0379 1
320 0380 1
321 0381 1
322 0382 1 --

```

```

323 0383 1
324 0384 2 BEGIN
325 0385 2
326 0386 2 LOCAL
327 0387 2 PFV: REF $PASSPFV FILE_VARIABLE, | Pascal File Variable
328 0388 2 PFV_ADDR: VOLATILE, | Address of PFV
329 0389 2 XABKEY_ADDR: VOLATILE, | Contains address of dynamically
330 0390 2 | allocated KEY XABs.
331 0391 2 XABKEY_SIZE: VOLATILE, | Contains size in bytes of
332 0392 2 | allocated KEY XABs.
333 0393 2
334 0394 2 XAB_FHC: $XABFHC_DECL, | RMS FHC XAB
335 0395 2 XAB_SUM: $XABSUM_DECL, | RMS SUM XAB
336 0396 2
337 0397 2 USER_ACTION_BPV: REF VECTOR [2, LONG], | USER_ACTION routine descriptor
338 0398 2 USR_ORG: BYTE, | User-specified organization
339 0399 2 USR_RFM: BYTE, | User-specified record format
340 0400 2 USR_USZ: WORD, | User-specified record size
341 0401 2 FILE_TYPE: SIGNED, | Is this INPUT, OUTPUT or neither?
342 0402 2 | Values can be:
343 0403 2 | K_INPUT = -1
344 0404 2 | K_NEITHER = 0
345 0405 2 | K_OUTPUT = 1
346 0406 2 RESULT_NAME_STRING: VECTOR [MAXU(NAM$C_MAXRSS, LNM$C_NAMLENGTH), BYTE];
347 0407 2 | Resultant name string
348 0408 2
349 0409 2 STACKLOCAL
350 0410 2 KEYWORDS_SEEN: BITVECTOR [PASSK_OPEKEYHI+1]; ! Keep track of keywords seen
351 0411 2
352 0412 2 !+
353 0413 2 ! PASS$AB_KEYWD_NAME_TABLE correlates the individual keyword code values to
354 0414 2 ! the names keyword. This is used to keep track of which keywords have
355 0415 2 ! been seen.
356 0416 2 !-
357 0417 2
358 0418 2 GLOBAL
359 0419 2 PASS$AB_KEYWD_NAME_TABLE: VECTOR [PASSK_OPEKEYHI+1, BYTE] PSECT (_PASSCODE)
360 0420 2 PRESET (
361 0421 2 [PASSK_FILE_NAME] = PASSK_FILE_NAME,
362 0422 2 [PASSK_DEFAULT_FILE_NAME] = PASSK_DEFAULT_FILE_NAME,
363 0423 2 [PASSK_HISTORY_OLD] = PASSK_HISTORY,
364 0424 2 [PASSK_HISTORY_NEW] = PASSK_HISTORY,
365 0425 2 [PASSK_HISTORY_UNKNOWN] = PASSK_HISTORY,
366 0426 2 [PASSK_HISTORY_READONLY] = PASSK_HISTORY,
367 0427 2 [PASSK_RECORD_LENGTH] = PASSK_RECORD_LENGTH,
368 0428 2 [PASSK_ACCESS_METHOD_SEQUENTIAL] = PASSK_ACCESS_METHOD,
369 0429 2 [PASSK_ACCESS_METHOD_DIRECT] = PASSK_ACCESS_METHOD,
370 0430 2 [PASSK_ACCESS_METHOD_KEYED] = PASSK_ACCESS_METHOD,
371 0431 2 [PASSK_RECORD_TYPE_FIXED] = PASSK_RECORD_TYPE,
372 0432 2 [PASSK_RECORD_TYPE_VARIABLE] = PASSK_RECORD_TYPE,
373 0433 2 [PASSK_RECORD_TYPE_STREAM] = PASSK_RECORD_TYPE,
374 0434 2 [PASSK_RECORD_TYPE_STREAM_CR] = PASSK_RECORD_TYPE,
375 0435 2 [PASSK_RECORD_TYPE_STREAM_LF] = PASSK_RECORD_TYPE,
376 0436 2 [PASSK_CARRIAGE_CONTROL_LIST] = PASSK_CARRIAGE_CONTROL,
377 0437 2 [PASSK_CARRIAGE_CONTROL_FORTRAN] = PASSK_CARRIAGE_CONTROL,
378 0438 2 [PASSK_CARRIAGE_CONTROL_NONE] = PASSK_CARRIAGE_CONTROL,
379 0439 2 [PASSK_ORGANIZATION_SEQUENTIAL] = PASSK_ORGANIZATION,

```

```

380 0440 2 [PASSK_ORGANIZATION_RELATIVE] = PASSK_ORGANIZATION,
381 0441 2 [PASSK_ORGANIZATION_INDEXED] = PASSK_ORGANIZATION,
382 0442 2 [PASSK_DISPOSITION_SAVE] = PASSK_DISPOSITION,
383 0443 2 [PASSK_DISPOSITION_DELETE] = PASSK_DISPOSITION,
384 0444 2 [PASSK_DISPOSITION_PRINT] = PASSK_DISPOSITION,
385 0445 2 [PASSK_DISPOSITION_PRINT_DELETE] = PASSK_DISPOSITION,
386 0446 2 [PASSK_DISPOSITION_SUBMIT] = PASSK_DISPOSITION,
387 0447 2 [PASSK_DISPOSITION_SUBMIT_DELETE] = PASSK_DISPOSITION,
388 0448 2 [PASSK_ERROR] = PASSK_ERROR,
389 0449 2 [PASSK_USER_ACTION] = PASSK_USER_ACTION,
390 0450 2 [PASSK_SHARING_NONE] = PASSK_SHARING,
391 0451 2 [PASSK_SHARING_READONLY] = PASSK_SHARING,
392 0452 2 [PASSK_SHARING_READWRITE] = PASSK_SHARING);
393 0453 2
394 0454 2 LITERAL
395 0455 2     K_INPUT = -1,           ! For FILE_TYPE - file is INPUT
396 0456 2     K_NEITHER = 0,      ! For FILE_TYPE - neither INPUT nor OUTPUT
397 0457 2     K_OUTPUT = 1;       ! For FILE_TYPE - file is OUTPUT
398 0458 2
399 0459 2 BUILTIN
400 0460 2     TESTBITCS,
401 0461 2     TESTBITSC,
402 0462 2     ACTUALCOUNT;
403 0463 2
404 0464 2 BIND
405 0465 2     RAB = FCB: REF BLOCK [, BYTE],
406 0466 2     FAB = FCB: REF $PASS$FAB_FCB_STRUCT,
407 0467 2     NAM = FCB: REF $PASS$NAM_FCB_STRUCT;
408 0468 2
409 0469 2 |*
410 0470 2 | Establish local condition handler which will close the file upon
411 0471 2 | an unwind.
412 0472 2 |*
413 0473 2
414 0474 2 ENABLE OPEN_HANDLER (PFV_ADDR, XABKEY_ADDR, XABKEY_SIZE);
415 0475 2
416 0476 2 |*
417 0477 2 | Fill in local XAB blocks.
418 0478 2 |*
419 0479 2
420 0480 2 $XABFHC_INIT (XAB=XAB_FHC, NXT=XAB_SUM);
421 0481 2 $XABSUM_INIT (XAB=XAB_SUM);
422 0482 2
423 0483 2 |*
424 0484 2 | Move PFV argument to local PFV.
425 0485 2 |*
426 0486 2
427 0487 2 PFV = .IN_PFV;
428 0488 2
429 0489 2 |*
430 0490 2 | If the PFD address is relative, resolve it.
431 0491 2 |*
432 0492 2
433 0493 2 IF .PFV [PFV$V_RELPFD]
434 0494 2 THEN
435 0495 2     BEGIN
436 0496 2     PFV [PFV$A_PFD] = .PFV [PFV$A_PFD] + PFV [PFV$R_PFV];

```

```

437 0497 3          PFV [PFV$V_RELPFD] = 0;
438 0498          END;
439 0499
440 0500          !+
441 0501          !- If the buffer address is relative, resolve it.
442 0502
443 0503
444 0504          IF TESTBITSC (PFV [PFV$V_RELBUF])
445 0505          THEN
446 0506              PFV [PFV$A_BUFFER] = .PFV [PFV$A_BUFFER] + PFV [PFV$R_PFV];
447 0507
448 0508          !+
449 0509          !- Declare exit handler if it hasn't yet been declared.
450 0510
451 0511
452 0512          IF TESTBITCS (EXITH_DECLARED)
453 0513          THEN
454 0514              BEGIN
455 0515                  !+
456 0516                  !- Allocate and fill in the control descriptor block.
457 0517
458 0518
459 0519                  LOCAL
460 0520                      EXITH_CONTROL_BLOCK: REF VECTOR [, LONG];
461 0521
462 0522                      EXITH_CONTROL_BLOCK = PASS$GET_VM (PFV [PFV$R_PFV], 20); ! 5 longwords
463 0523                      EXITH_CONTROL_BLOCK [1] = EXIT_HANDLER; ! Routine address
464 0524                      EXITH_CONTROL_BLOCK [2] = 1; ! 1 additional longword
465 0525                      EXITH_CONTROL_BLOCK [3] = EXITH_CONTROL_BLOCK [4]; ! Reason for exit
466 0526                      $DCLEXM (DESB[K=EXITH_CONTROL_BLOCK [0]]); ! Assume success
467 0527                      END;
468 0528
469 0529          !+
470 0530          !- Set FILE_TYPE depending on whether the file is INPUT, OUTPUT or neither.
471 0531
472 0532
473 0533          IF PFV [PFV$R_PFV] EQLA PASS$V_INPUT
474 0534          THEN
475 0535              FILE_TYPE = K_INPUT
476 0536          ELSE IF PFV [PFV$R_PFV] EQLA PASS$V_OUTPUT
477 0537          THEN
478 0538              FILE_TYPE = K_OUTPUT
479 0539          ELSE
480 0540              FILE_TYPE = K_NEITHER;
481 0541
482 0542          !+
483 0543          !- Allocate the FCB+RAB+FAB+NAM block. It will be zero-filled by
484 0544          !- PASS$GET_VM.
485 0545
486 0546
487 0547          FCB = PASS$GET_VM (PFV [PFV$R_PFV], PASS$K_FILE_DYN_BLN) + FCB$K_BLN;
488 0548
489 0549          !+
490 0550          !- Initialize blocks.
491 0551
492 0552
493 0553          RAB [RAB$B_BID] = RAB$C_BID;

```

```

494 0554 2 RAB [RAB$B_BLN] = RAB$C_BLN;
495 0555 2 FAB [FAB$B_BID] = FAB$C_BID;
496 0556 2 FAB [FAB$B_BLN] = FAB$C_BLN;
497 0557 2 NAM [NAM$B_BID] = NAM$C_BID;
498 0558 2 NAM [NAM$B_BLN] = NAM$C_BLN;
499 0559 2 NAM [NAM$L_ESA] = RESULT NAME STRING;
500 0560 2 NAM [NAM$B_ESS] = NAM$C_MAXRSS;
501 0561 2 NAM [NAM$L_RSA] = RESULT NAME STRING;
502 0562 2 NAM [NAM$B_RSS] = NAM$C_MAXRSS;
503 0563 2 RAB [RAB$L_FAB] = FAB [0,0,0,0];
504 0564 2 FAB [FAB$L_NAM] = NAM [0,0,0,0];
505 0565 2 FAB [FAB$L_XAB] = XAB_FHC;
506 0566
507 0567
508 0568
509 0569
510 0570
511 0571 2 FCB [FCB$A_PFV] = PFV [PFV$R_PFV];
512 0572
513 0573
514 0574 2 Store FCB address in PFV and set the FCB_VALID bit.
515 0575
516 0576
517 0577 2 PFV [PFV$A_FCB] = FCB [FCB$R_FCB];
518 0578 2 PFV [PFV$V_FCB_VALID] = 1;
519 0579 2 PFV_ADDR = PFV [PFV$R_PFV];
520 0580
521 0581
522 0582 2 Get information from PFD that we need.
523 0583
524 0584
525 0585
526 0586
527 0587 3 BEGIN
528 0588 3 LOCAL
529 0589 3 PFD: REF $PASSPFD_FILE_DESCRIPTOR; ! Pascal File Descriptor
530 0590 3 PFD = .PFV [PFV$A_PFD]; ! Get PFD address
531 0591 3 FCB [FCB$W_ATTRIB] = .PFD [PFD$W_ATTRIB]; ! Set attributes
532 0592 3 FCB [FCB$A_PFD] = PFD [PFD$R_PFD]; ! PFD address
533 0593
534 0594 3 Set the default RECORD_LENGTH. If a TEXTFILE, it's 133. Otherwise,
535 0595 3 its the length of the record (not including the length word for
536 0596 3 a VARYING.)
537 0597
538 0598
539 0599 3 IF NOT .FCB [FCB$V_TEXT]
540 0600 3 THEN
541 0601 4 BEGIN
542 0602 4 IF .PFD [PFD$L_LENGTH] GTRU 65535 ! Component type too long?
543 0603 4 THEN
544 0604 4 $PASSIO ERROR (PASS_INVRECLN,1,.PFD [PFD$L_LENGTH]);
545 0605 4 IF .FCB [FCB$V_VARYING]
546 0606 4 THEN
547 0607 4 RAB [RAB$W_USZ] = .PFD [PFD$L_LENGTH] - 2 ! Subtract for length word
548 0608 4 ELSE
549 0609 4 RAB [RAB$W_USZ] = .PFD [PFD$L_LENGTH];
550 0610 4 END

```

PASSOPEN2
1-015

OPEN procedure
PASS\$OPEN - Open a file

J 11
16-Sep-1984 01:46:15
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742
[PASRTL.SRC]PASOPEN2.B32;1

Page 14
(4)

```
: 551      0611      3      ELSE
: 552      0612      3      RAB [RAB$W_USZ] = 133;
: 553      0613      3
: 554      0614      3      USR_USZ = .RAB [RAB$W_USZ];
: 555      0615      2      END;
: 556      0616      2
```

. Textfile


```

558 0617 2  !+
559 0618 2  ! Look through keyword list and set all specified attributes.
560 0619 2  !-
561 0620 2
562 0621 2  KEYWORDS_SEEN <0,32> = 0; ! Indicate no keywords seen
563 0622 2  KEYWORDS_SEEN <32,PASSK_OPEKEYHI-31> = 0;
564 0623 2  INCR I FROM 0 TO (ACTUALCOUNT () - 2) DO
565 0624 2  BEGIN
566 0625 3  LOCAL
567 0626 3  KEYWD_VALUE;
568 0627 3
569 0628 3
570 0629 3  !+
571 0630 3  ! Check for valid keyword code. If not already seen, use the
572 0631 3  ! keyword.
573 0632 3  !-
574 0633 3
575 0634 3  KEYWD_VALUE = .KEYWORDS [I];
576 0635 3  IF .KEYWD_VALUE LSS PASSK_OPEKEYLO OR .KEYWD_VALUE GTR PASSK_OPEKEYHI
577 0636 3  THEN
578 0637 3  $PASSIO_ERROR (PASS_INVARGPAS,0);
579 0638 3
580 0639 3  IF TESTBITCS (KEYWORDS_SEEN [.PASS$AB_KEYWD_NAME_TABLE[.KEYWD_VALUE]])
581 0640 3  THEN
582 0641 3  CASE .KEYWD_VALUE FROM PASSK_OPEKEYLO TO PASSK_OPEKEYHI OF
583 0642 3  SET
584 0643 3
585 0644 3  [PASSK_FILE_NAME]:
586 0645 4  BEGIN
587 0646 4  LOCAL
588 0647 4  FNS: WORD;
589 0648 4  FNS = .KEYWORDS [(I=.I+1)]; ! Get string size
590 0649 4  IF .FNS GTRU 255
591 0650 4  THEN
592 0651 4  $PASSIO_ERROR (PASS_INVFILSYN,0);
593 0652 4  FAB [FAB$B_FNS] = .FNS;
594 0653 4  FAB [FAB$L_FNA] = .KEYWORDS [(I=.I+1)]; ! String address
595 0654 3  END;
596 0655 3
597 0656 3  [PASSK_DEFAULT_FILE_NAME]:
598 0657 4  BEGIN
599 0658 4  LOCAL
600 0659 4  DNS: WORD;
601 0660 4  DNS = .KEYWORDS [(I=.I+1)]; ! Get string size
602 0661 4  IF .DNS GTRU 255
603 0662 4  THEN
604 0663 4  $PASSIO_ERROR (PASS_INVFILSYN,0);
605 0664 4  FAB [FAB$B_DNS] = .DNS;
606 0665 4  FAB [FAB$L_DNA] = .KEYWORDS [(I=.I+1)]; ! String address
607 0666 3  END;
608 0667 3
609 0668 3  [PASSK_HISTORY_OLD]:
610 0669 4  BEGIN
611 0670 4  FCB [FCB$V_OLD_FILE] = 1;
612 0671 4  FAB [FAB$V_GET] = 1;
613 0672 4  FAB [FAB$V_PUT] = 1;
614 0673 4  FAB [FAB$V_TRN] = 1;

```

```

615      0674      4          FAB [FABS$V_DEL] = 1;
616      0675      4          FAB [FABS$V_UPD] = 1;
617      0676      3          END;
618      0677      3
619      0678      3          [PASS$K_HISTORY_NEW]:
620      0679      4          BEGIN
621      0680      4          FAB [FABS$V_GET] = 1;
622      0681      4          FAB [FABS$V_PUT] = 1;
623      0682      4          FAB [FABS$V_TRN] = 1;
624      0683      4          FAB [FABS$V_DEL] = 1;
625      0684      4          FAB [FABS$V_UPD] = 1;
626      0685      3          END;
627      0686      3
628      0687      3          [PASS$K_HISTORY_UNKNOWN]:
629      0688      4          BEGIN
630      0689      4          FAB [FABS$V_CIF] = 1;
631      0690      4          FAB [FABS$V_GET] = 1;
632      0691      4          FAB [FABS$V_PUT] = 1;
633      0692      4          FAB [FABS$V_TRN] = 1;
634      0693      4          FAB [FABS$V_DEL] = 1;
635      0694      4          FAB [FABS$V_UPD] = 1;
636      0695      3          END;
637      0696      3
638      0697      3          [PASS$K_HISTORY_READONLY]:
639      0698      4          BEGIN
640      0699      4          FCB [FCB$V_OLD_FILE] = 1;      ! MUST be old
641      0700      4          FCB [FCB$V_READ_ONLY] = 1;
642      0701      4          FAB [FABS$V_GET] = 1;
643      0702      3          END;
644      0703      3
645      0704      3          [PASS$K_RECORD_LENGTH]:
646      0705      4          BEGIN
647      0706      4          IF .KEYWORDS [(I=.I+1)] GTRU 65535
648      0707      4          THEN
649      0708      4          $PASS$IO_ERROR (PASS$INVRECLN,1,.KEYWORDS [I]);
650      0709      4          USR_USZ = .KEYWORDS [I];
651      0710      4
652      0711      4          !+
653      0712      4          ! If RECORD_LENGTH specified for textfile, use it.
654      0713      4          ! Otherwise, keep it so we can give an error later if
655      0714      4          ! it doesn't match existing file's record length.
656      0715      4          !-
657      0716      4
658      0717      4          IF .FCB [FCB$V_TEXT]
659      0718      4          THEN
660      0719      4          RAB [RAB$W_USZ] = .USR_USZ;
661      0720      3          END;
662      0721      3
663      0722      3          [PASS$K_ACCESS_METHOD_SEQUENTIAL]:
664      0723      4          BEGIN
665      0724      4          FCB [FCB$V_SEQUENTIAL] = 1;
666      0725      4          FAB [FABS$V_SQO] = 1;      ! Optimize network access
667      0726      3          END;
668      0727      3
669      0728      3          [PASS$K_ACCESS_METHOD_DIRECT]:
670      0729      4          BEGIN
671      0730      4          FCB [FCB$V_DIRECT] = 1;

```

```

: 672      0731  4      RAB [RAB$L_KBF] = FCB [FCB$L_COMPONENT];
: 673      0732  4      RAB [RAB$V_UIF] = 1;      ? Update if record exists
: 674      0733  3      END;
: 675      0734  3
: 676      0735  3      [PASSK_ACCESS_METHOD_KEYED]:
: 677      0736  4      BEGIN
: 678      0737  4      FCB [FCB$V_KEYED] = 1;
: 679      0738  3      END;
: 680      0739  3
: 681      0740  3      [PASSK_RECORD_TYPE_FIXED]:
: 682      0741  4      BEGIN
: 683      0742  4      FAB [FAB$B_RFM] = FAB$C_FIX;
: 684      0743  3      END;
: 685      0744  3
: 686      0745  3      [PASSK_RECORD_TYPE_VARIABLE]:
: 687      0746  4      BEGIN
: 688      0747  4      FAB [FAB$B_RFM] = FAB$C_VAR;
: 689      0748  3      END;
: 690      0749  3
: 691      0750  3      [PASSK_RECORD_TYPE_STREAM]:
: 692      0751  4      BEGIN
: 693      0752  4      FAB [FAB$B_RFM] = FAB$C_STM;
: 694      0753  3      END;
: 695      0754  3
: 696      0755  3      [PASSK_RECORD_TYPE_STREAM_CR]:
: 697      0756  4      BEGIN
: 698      0757  4      FAB [FAB$B_RFM] = FAB$C_STMCR;
: 699      0758  3      END;
: 700      0759  3
: 701      0760  3      [PASSK_RECORD_TYPE_STREAM_LF]:
: 702      0761  4      BEGIN
: 703      0762  4      FAB [FAB$B_RFM] = FAB$C_STMLF;
: 704      0763  3      END;
: 705      0764  3
: 706      0765  3      [PASSK_CARRIAGE_CONTROL_LIST]:
: 707      0766  4      BEGIN
: 708      0767  4      FAB [FAB$V_CR] = 1;
: 709      0768  3      END;
: 710      0769  3
: 711      0770  3      [PASSK_CARRIAGE_CONTROL_FORTRAN]:
: 712      0771  4      BEGIN
: 713      0772  4      FAB [FAB$V_FTN] = 1;
: 714      0773  3      END;
: 715      0774  3
: 716      0775  3      [PASSK_CARRIAGE_CONTROL_NONE]:
: 717      0776  3      ;      ! Do nothing
: 718      0777  3
: 719      0778  3      [PASSK_ORGANIZATION_SEQUENTIAL]:
: 720      0779  4      BEGIN
: 721      0780  4      FAB [FAB$B_ORG] = FAB$C_SEQ;
: 722      0781  3      END;
: 723      0782  3
: 724      0783  3      [PASSK_ORGANIZATION_RELATIVE]:
: 725      0784  4      BEGIN
: 726      0785  4      FAB [FAB$B_ORG] = FAB$C_REL;
: 727      0786  3      END;
: 728      0787  3
```

```
.....: 729      0788      3      [PASSK_ORGANIZATION_INDEXED]:  
.....: 730      0789      4      BEGIN  
.....: 731      0790      4      FAB [FABS$B_ORG] = FABS$C_IDX;  
.....: 732      0791      3      END;  
.....: 733      0792      3  
.....: 734      0793      3      [PASSK_DISPOSITION_SAVE]:  
.....: 735      0794      4      BEGIN  
.....: 736      0795      4      FCB [FCBS$V_SAVE] = 1;  
.....: 737      0796      3      END;  
.....: 738      0797      3  
.....: 739      0798      3      [PASSK_DISPOSITION_DELETE]:  
.....: 740      0799      4      BEGIN  
.....: 741      0800      4      FCB [FCBS$V_DELETE] = 1;  
.....: 742      0801      3      END;  
.....: 743      0802      3  
.....: 744      0803      3      [PASSK_DISPOSITION_PRINT]:  
.....: 745      0804      4      BEGIN  
.....: 746      0805      4      FCB [FCBS$V_SAVE] = 1;  
.....: 747      0806      4      FCB [FCBS$V_PRINT] = 1;  
.....: 748      0807      3      END;  
.....: 749      0808      3  
.....: 750      0809      3      [PASSK_DISPOSITION_PRINT_DELETE]:  
.....: 751      0810      4      BEGIN  
.....: 752      0811      4      FCB [FCBS$V_PRINT] = 1;  
.....: 753      0812      4      FCB [FCBS$V_DELETE] = 1;  
.....: 754      0813      3      END;  
.....: 755      0814      3  
.....: 756      0815      3      [PASSK_DISPOSITION_SUBMIT]:  
.....: 757      0816      4      BEGIN  
.....: 758      0817      4      FCB [FCBS$V_SUBMIT] = 1;  
.....: 759      0818      4      FCB [FCBS$V_SAVE] = 1;  
.....: 760      0819      3      END;  
.....: 761      0820      3  
.....: 762      0821      3      [PASSK_DISPOSITION_SUBMIT_DELETE]:  
.....: 763      0822      4      BEGIN  
.....: 764      0823      4      FCB [FCBS$V_SUBMIT] = 1;  
.....: 765      0824      4      FCB [FCBS$V_DELETE] = 1;  
.....: 766      0825      3      END;  
.....: 767      0826      3  
.....: 768      0827      3      [PASSK_USER_ACTION]:  
.....: 769      0828      4      BEGIN  
.....: 770      0829      4      USER_ACTION_BPV = .KEYWORDS [(I=.I+1)];  
.....: 771      0830      4      FCB [FCBS$V_USER_ACTION] = 1;  
.....: 772      0831      3      END;  
.....: 773      0832      3  
.....: 774      0833      3      [PASSK_SHARING_NONE]:  
.....: 775      0834      4      BEGIN  
.....: 776      0835      4      FAB [FABS$V_NIL] = 1;  
.....: 777      0836      3      END;  
.....: 778      0837      3  
.....: 779      0838      3      [PASSK_SHARING_READONLY]:  
.....: 780      0839      4      BEGIN  
.....: 781      0840      4      FAB [FABS$V_SHRGET] = 1;  
.....: 782      0841      3      END;  
.....: 783      0842      3  
.....: 784      0843      3      [PASSK_SHARING_READWRITE]:  
.....: 785      0844      4      BEGIN
```

```
: 786 0845 4  
: 787 0846 4  
: 788 0847 4  
: 789 0848 4  
: 790 0849 3  
: 791 0850 3  
: 792 0851 3  
: 793 0852 4  
: 794 0853 4  
: 795 0854 4  
: 796 0855 4  
: 797 0856 4  
: 798 0857 4  
: 799 0858 3  
800 0859 3  
: 801 0860 3  
: 802 0861 3  
: 803 0862 3  
: 804 0863 3  
: 805 0864 3  
: 806 0865 2  
: 807 0866 2  
END;
```

```
FAB [FAB$V_SHRPUT] = 1;  
FAB [FAB$V_SHRGET] = 1;  
FAB [FAB$V_SHRDEL] = 1;  
FAB [FAB$V_SHRUPD] = 1;  
END; /  
[PASS$K_ERROR]:  
BEGIN  
+  
! The ERROR parameter, if specified, was already processed  
! in PASS$OPEN2, so just ignore it here.  
-  
I = .I + 1; ! Ignore next parameter  
END;  
[INRANGE,OUTRANGE]:  
$PASS$IO_ERROR (PASS$_INVARGPAS,0); ! Invalid argument
```

TES;

```

809 0867 2      !+
810 0868 2      ! Apply defaults.
811 0869 2      -
812 0870 2      -
813 0871 2      !+
814 0872 2      ! Default FILE_NAME
815 0873 2      -
816 0874 2      -
817 0875 2      IF NOT .KEYWORDS_SEEN [PASS$K_FILE_NAME]
818 0876 2      THEN
819 0877 2      BEGIN
820 0878 2      -
821 0879 3      !+
822 0880 3      ! If this is standard file INPUT or OUTPUT, attempt to translate
823 0881 3      ! PASS$INPUT or PASS$OUTPUT, respectively. If no translation, use
824 0882 3      ! SYSS$INPUT or SYSS$OUTPUT instead. If not INPUT or OUTPUT, use
825 0883 3      ! file variable name.
826 0884 3      -
827 0885 3      -
828 0886 3      IF .FILE_TYPE NEQ K_NEITHER      ! neither INPUT nor OUTPUT?
829 0887 3      THEN
830 0888 4      BEGIN
831 0889 4      LOCAL
832 0890 4      LOGNAM_DSC: BLOCK [8, BYTE],      ! Descriptor for logical name
833 0891 4      RSLNAM_DSC: BLOCK [8, BYTE],      ! Descriptor for resultant name
834 0892 4      SUBSTITUTE_NAME;                ! Address for substitute name
835 0893 4
836 0894 4
837 0895 4      IF .FILE_TYPE EQL K_INPUT
838 0896 4      THEN
839 0897 5      BEGIN
840 0898 5      LOGNAM_DSC [DSC$W_LENGTH] = %CHARCOUNT ('PASS$INPUT');
841 0899 5      LOGNAM_DSC [DSC$A_POINTER] = UPLIT BYTE ('PASS$INPUT');
842 0900 5      SUBSTITUTE_NAME = UPLIT BYTE ('SYSS$INPUT');
843 0901 5      END
844 0902 4      ELSE
845 0903 5      BEGIN
846 0904 5      LOGNAM_DSC [DSC$W_LENGTH] = %CHARCOUNT ('PASS$OUTPUT');
847 0905 5      LOGNAM_DSC [DSC$A_POINTER] = UPLIT BYTE ('PASS$OUTPUT');
848 0906 5      SUBSTITUTE_NAME = UPLIT BYTE ('SYSS$OUTPUT');
849 0907 4      END;
850 0908 4      LOGNAM_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
851 0909 4      LOGNAM_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
852 0910 4      RSLNAM_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
853 0911 4      RSLNAM_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
854 0912 4      RSLNAM_DSC [DSC$W_LENGTH] = LNM$C_NAMLENGTH;      ! String size
855 0913 4      RSLNAM_DSC [DSC$A_POINTER] = RESULT NAME STRING;
856 0914 4      IF $STRNLOG (LOGNAM = LOGNAM_DSC, RS[BUF = RSLNAM_DSC) EQLU SS$_NOTRAN
857 0915 4      THEN
858 0916 5      BEGIN
859 0917 5      !+
860 0918 5      ! Do the substitution
861 0919 5      -
862 0920 5      FAB [FAB$L_FNA] = .SUBSTITUTE_NAME;
863 0921 5      FAB [FAB$B_FNS] = .LOGNAM_DSC[DSC$W_LENGTH]; ! Can't be > 255
864 0922 5      END
865 0923 4      ELSE

```

```

866      0924      5      BEGIN
867      0925      5      !+
868      0926      5      ! Use Pascal specific name
869      0927      5      !-
870      0928      5      FAB [FAB$L_FNA] = .LOGNAM_DSC [DSC$A_POINTER];
871      0929      5      FAB [FAB$B_FNS] = .LOGNAM_DSC [DSC$W_LENGTH]; ! Can't be >255
872      0930      4      END;
873      0931      3      END;
874      0932      3
875      0933      3      IF .FAB [FAB$L_FNA] EQL 0      ! No name given yet?
876      0934      3      THEN
877      0935      3          IF .FCB [FCB$V_EXTERN]      ! Filename is file variable name
878      0936      3          THEN
879      0937      4              BEGIN
880      0938      4                  LOCAL
881      0939      4                      PFD: REF $PASS$PFD FILE_DESCRIPTOR;
882      0940      4                      PFD = .FCB [FCB$A_PFD];
883      0941      4                      FAB [FAB$B_FNS] = %CHRCHAR (PFD [PFD$T_NAME]);
884      0942      4                      FAB [FAB$L_FNA] = PFD [PFD$T_NAME] + 1;
885      0943      4                      END
886      0944      3          ELSE
887      0945      4              BEGIN
888      0946      4                  FAB [FAB$V_TMD] = 1;      ! No file name, temporary marked for delete
889      0947      4                  FAB [FAB$B_FNS] = %CHARCOUNT ('PASNONAME.TMP');
890      0948      4                  FAB [FAB$L_FNA] = UPLIT BYTE ('PASNONAME.TMP');
891      0949      4                  END;
892      0950      2      END;
893      0951      2
894      0952      2      !+
895      0953      2      ! Default DEFAULT_FILE_NAME
896      0954      2      !-
897      0955      2
898      0956      2      IF (NOT .KEYWORDS_SEEN [PASS$K_DEFAULT_FILE_NAME]) AND (.FCB [FCB$V_EXTERN])
899      0957      2      THEN
900      0958      3          BEGIN
901      0959      3              FAB [FAB$B_DNS] = %CHARCOUNT ('.DAT');
902      0960      3              FAB [FAB$L_DNA] = UPLIT BYTE ('.DAT');
903      0961      3          END;
904      0962      2
905      0963      2      !+
906      0964      2      ! Default HISTORY
907      0965      2      !-
908      0966      2
909      0967      2      IF NOT .KEYWORDS_SEEN [PASS$K_HISTORY]
910      0968      2      THEN
911      0969      3          BEGIN
912      0970      3              !+
913      0971      3              ! Default is NEW
914      0972      3              !-
915      0973      3              FAB [FAB$V_GET] = 1;
916      0974      3              FAB [FAB$V_PUT] = 1;
917      0975      3              FAB [FAB$V_TRN] = 1;
918      0976      3              FAB [FAB$V_DEL] = 1;
919      0977      3              FAB [FAB$V_UPD] = 1;
920      0978      3          END;
921      0979      2
922      0980      2      !+

```

```

923 0981 2  | Default ACCESS_METHOD
924 0982 2  | -
925 0983 2  |
926 0984 2  | IF NOT .KEYWORDS_SEEN [PASS$ACCESS_METHOD]
927 0985 2  | THEN
928 0986 2  |     BEGIN
929 0987 2  |         +
930 0988 2  |         | Default is SEQUENTIAL
931 0989 2  |         | -
932 0990 2  |         FCB [FCB$V_SEQUENTIAL] = 1;
933 0991 2  |         FAB [FAB$V_SQO] = 1;      ! Optimize network access
934 0992 2  |         END
935 0993 2  |     ELSE IF .FCB [FCB$V_TEXT] AND NOT .FCB [FCB$V_SEQUENTIAL]
936 0994 2  |     THEN
937 0995 2  |         $PASSIO_ERROR (PASS_TEXREQSEQ,0);      ! Textfiles require sequential organization and access
938 0996 2  |
939 0997 2  |     +
940 0998 2  |     | Default RECORD_TYPE
941 0999 2  |     | -
942 1000 2  |     |
943 1001 2  |     | IF NOT .KEYWORDS_SEEN [PASS$RECORD_TYPE]
944 1002 2  |     | THEN
945 1003 2  |     |     IF .FCB [FCB$V_TEXT] OR .FCB [FCB$V_VARYING]
946 1004 2  |     |     THEN
947 1005 2  |     |         FAB [FAB$B_RFM] = FAB$C_VAR
948 1006 2  |     |     ELSE
949 1007 2  |     |         FAB [FAB$B_RFM] = FAB$C_FIX;
950 1008 2  |     |     USR_RFM = .FAB [FAB$B_RFM];
951 1009 2  |     |
952 1010 2  |     | +
953 1011 2  |     | | Default CARRIAGE_CONTROL
954 1012 2  |     | | -
955 1013 2  |     | |
956 1014 2  |     | | IF NOT .KEYWORDS_SEEN [PASS$CARRIAGE_CONTROL]
957 1015 2  |     | | THEN
958 1016 2  |     | |     IF .FCB [FCB$V_TEXT] OR .FCB [FCB$V_VARYING]
959 1017 2  |     | |     THEN
960 1018 2  |     | |         FAB [FAB$V_CR] = 1; ! LIST is the default here
961 1019 2  |     | |
962 1020 2  |     | | +
963 1021 2  |     | | | Default DISPOSITION
964 1022 2  |     | | | -
965 1023 2  |     | | |
966 1024 2  |     | | | IF NOT .KEYWORDS_SEEN [PASS$DISPOSITION]
967 1025 2  |     | | | THEN
968 1026 2  |     | | |     IF NOT .FAB [FAB$V_TMD] ! Not a temporary?
969 1027 2  |     | | |     THEN
970 1028 2  |     | | |         FCB [FCB$V_SAVE] = 1;
971 1029 2  |     | | |
972 1030 2  |     | | | +
973 1031 2  |     | | | | Default ORGANIZATION.
974 1032 2  |     | | | | -
975 1033 2  |     | | | |
976 1034 2  |     | | | | IF NOT .KEYWORDS_SEEN [PASS$ORGANIZATION]
977 1035 2  |     | | | | THEN
978 1036 2  |     | | | |     FAB [FAB$B_ORG] = FAB$C_SEQ      ! SEQUENTIAL
979 1037 2  |     | | | | ELSE IF .FCB [FCB$V_TEXT] AND (.FAB [FAB$B_ORG] NEQ FAB$C_SEQ)

```



```

: 980      1038 2      THEN
: 981      1039 2      $PASSIO_ERROR (PASS_TEXREQSEQ,0);      ! Textfiles require sequential organization and access
: 982      1040 2      USR_ORG = .FAB [FAB$B_ORG];
: 983      1041 2
: 984      1042 2      !+
: 985      1043 2      ! Default SHARING
: 986      1044 2      !-
: 987      1045 2
: 988      1046 2      !+
: 989      1047 2      ! If ORGANIZATION:=SEQUENTIAL (explicitly), and if the file is shared
: 990      1048 2      ! so that there is at least one writer, then set FAB$V_UPI.
: 991      1049 2      !-
: 992      1050 2
: 993      1051 2      IF (.KEYWORDS_SEEN [PASS$K_ORGANIZATION] AND
: 994      1052 2      (.FAB [FAB$B_ORG] EQL FAB$C_SEQ))
: 995      1053 2      THEN
: 996      1054 2      IF .FAB [FAB$V_SHRPUT] OR
: 997      1055 2      (.FAB [FAB$V_PUT] AND .FAB [FAB$V_SHRGET])
: 998      1056 2      THEN
: 999      1057 2      FAB [FAB$V_UPI] = 1;
: 1000     1058 2
: 1001     1059 2      IF NOT .KEYWORDS_SEEN [PASS$K_SHARING]
: 1002     1060 2      THEN
: 1003     1061 2      IF .FCB [FCB$V_READ_ONLY]
: 1004     1062 2      THEN
: 1005     1063 2      FAB [FAB$V_SHRGET] = 1
: 1006     1064 2      ELSE
: 1007     1065 2      FAB [FAB$V_NIL] = 1;
: 1008     1066 2
```

```

: 1010      1067      2      | +
: 1011      1068      2      | | Check for conflicts
: 1012      1069      2      | | +
: 1013      1070      2      | |
: 1014      1071      2      | | +
: 1015      1072      2      | | Check for incompatible access method
: 1016      1073      2      | | -
: 1017      1074      2      | |
: 1018      1075      2      | IF NOT .FCB [FCB$V_OLD_FILE]
: 1019      1076      2      | THEN
: 1020      1077      3      |     IF NOT (
: 1021      1078      3      |         (.FCB [FCB$V_SEQUENTIAL]) OR
: 1022      1079      4      |         (.FCB [FCB$V_DIRECT] AND
: 1023      1080      5      |             ((.FAB [FAB$B_ORG] EQL FAB$C_REL) OR
: 1024      1081      3      |             ((.FAB [FAB$B_ORG] EQL FAB$C_SEQ) AND (.FAB [FAB$B_RFM] EQL FAB$C_FIX))) OR
: 1025      1082      4      |         (.FCB [FCB$V_KEYED] AND
: 1026      1083      3      |             (.FAB [FAB$B_ORG] EQL FAB$C_IDX)))
: 1027      1084      2      |     THEN
: 1028      1085      2      |         $PASSIO_ERROR (PASS_ACCMETINC,0);      ! Incompatible access method
: 1029      1086      2      |
: 1030      1087      2      | | +
: 1031      1088      2      | | Check for incompatible DISPOSITION.
: 1032      1089      2      | | -
: 1033      1090      2      | |
: 1034      1091      2      | IF .FAB [FAB$V_TMD] AND
: 1035      1092      3      |     (.FCB [FCB$V_SAVE] OR .FCB [FCB$V_SUBMIT] OR .FCB [FCB$V_PRINT] OR
: 1036      1093      3      |     .FCB [FCB$V_OLD_FILE] OR .FAB [FAB$V_CIF])
: 1037      1094      2      | THEN
: 1038      1095      2      |     $PASSIO_ERROR (PASS_FILNAMREQ,0);      ! File name required
: 1039      1096      2      |
: 1040      1097      2      | | +
: 1041      1098      2      | | Check for incompatible RECORD LENGTH.  If creating non-text file
: 1042      1099      2      | | with fixed-length records, RECORD_LENGTH and component type must match.
: 1043      1100      2      | | -
: 1044      1101      2      | |
: 1045      1102      2      | IF NOT .FCB [FCB$V_TEXT] AND NOT .FCB [FCB$V_OLD_FILE] AND
: 1046      1103      2      |     (.FAB [FAB$B_RFM] EQL FAB$C_FIX) AND
: 1047      1104      3      |     (.USR_USZ NEQU .RAB [RAB$W_USZ])
: 1048      1105      2      | THEN
: 1049      1106      2      |     $PASSIO_ERROR (PASS_RECLEINCL,0);

```

```

: 1051      1107      2      !+
: 1052      1108      2      ! If key descriptor block is present, allocate and fill in KEY XABs.
: 1053      1109      2      !-
: 1054      1110      2
: 1055      1111      2      BEGIN
: 1056      1112      2
: 1057      1113      2      LOCAL
: 1058      1114      2      PFD: REF $PASS$PFD_FILE_DESCRIPTOR; ! Pascal File Descriptor
: 1059      1115      2
: 1060      1116      2      PFD = .PFV [PFV$A_PFD];
: 1061      1117      2
: 1062      1118      2      IF .PFD [PFDSA_KDB] NEQ 0
: 1063      1119      2      THEN
: 1064      1120      2          FILL_KEY_XABS (PFV [PFV$R_PFV],
: 1065      1121      2              FCB [FCB$R_FCB],
: 1066      1122      2              PFD [PFDSR_PFD] + .PFD [PFDSA_KDB], ! Resolve self relative
: 1067      1123      2              XAB_SUM,
: 1068      1124      2              XABKEY_ADDR,
: 1069      1125      2              XABKEY_SIZE);
: 1070      1126      2
: 1071      1127      2      END;
: 1072      1128      2
: 1073      1129      2      !+
: 1074      1130      2      ! Fill in remaining fields in RMS control blocks
: 1075      1131      2      !-
: 1076      1132      2
: 1077      1133      2      FAB [FAB$V_NEF] = 1;          ! Don't position to EOF
: 1078      1134      2      FAB [FAB$V_DFW] = 1;          ! Deferred write speeds up Relative and Indexed
: 1079      1135      2      RAB [RAB$V_RAH] = 1;          ! Read-ahead speeds up sequential
: 1080      1136      2      RAB [RAB$V_WBH] = 1;          ! Write-behind speeds up sequential
: 1081      1137      2
: 1082      1138      2      !+
: 1083      1139      2      ! Set maximum record size for all but variable-length sequential files,
: 1084      1140      2      ! unless the user specified RECORD_LENGTH.
: 1085      1141      2      !-
: 1086      1142      2      IF .FAB [FAB$B_ORG] NEQ FAB$C_SEQ OR
: 1087      1143      2      .FAB [FAB$B_RFM] EQL FAB$C_FIX OR
: 1088      1144      2      .KEYWORDS_SEEN [PASS$K_RECORD_LENGTH]
: 1089      1145      2      THEN
: 1090      1146      2          FAB [FAB$W_MRS] = .USR_USZ;
: 1091      1147      2
: 1092      1148      2
: 1093      1149      2      !+
: 1094      1150      2      ! If USER_ACTION was specified, call the user routine to do the
: 1095      1151      2      ! open and connect. Otherwise, do it here.
: 1096      1152      2      !-
: 1097      1153      2
: 1098      1154      2      BEGIN
: 1099      1155      2      LOCAL
: 1100      1156      2          STATUS;
: 1101      1157      2
: 1102      1158      2      IF .KEYWORDS_SEEN [PASS$K_USER_ACTION]
: 1103      1159      2      THEN
: 1104      1160      2          BEGIN
: 1105      1161      2          LINKAGE
: 1106      1162      2              USER_ACTION_LNK = CALL (REGISTER=1, STANDARD, STANDARD, STANDARD);
: 1107      1163      2          STATUS = USER_ACTION_LNK (.USER_ACTION_BPV [0], ! Address

```

```
.USER ACTION BPV [1],      ! Environment
FAB [0,0,0,0],             ! FAB address
RAB [0,0,0,0],             ! RAB address
PFV [PFV$R_PFV]);         ! PFV address
```

```

1108 1164 4
1109 1165 4
1110 1166 4
1111 1167 4
1112 1168 4
1113 1169 3
1114 1170 4
1115 1171 4
1116 1172 4
1117 1173 4
1118 1174 4
1119 1175 4
1120 1176 5
1121 1177 6
1122 1178 6
1123 1179 6
1124 1180 7
1125 1181 8
1126 1182 8
1127 1183 9
1128 1184 8
1129 1185 7
1130 1186 6
1131 1187 6
1132 1188 4
1133 1189 4
1134 1190 4
1135 1191 4
1136 1192 4
1137 1193 4
1138 1194 4
1139 1195 4
1140 1196 4
1141 1197 4
1142 1198 4
1143 1199 4
1144 1200 4
1145 1201 5
1146 1202 5
1147 1203 5
1148 1204 5
1149 1205 5
1150 1206 5
1151 1207 5
1152 1208 5
1153 1209 5
1154 1210 6
1155 1211 7
1156 1212 7
1157 1213 7
1158 1214 8
1159 1215 7
1160 1216 7
1161 1217 5
1162 1218 4
1163 1219 4
1164 1220 4

      END
ELSE
  BEGIN
    !+
    ! Open or create file.
    !-
    STATUS = (
      BEGIN
        LOCAL
          $$STATUS;
        DO ($$STATUS =
          (IF .FCB [FCB$V_OLD_FILE]
            THEN $OPEN (FAB=FAB [0,0,0,0])
            ELSE $CREATE (FAB=FAB [0,0,0,0])))
          UNTIL (.$$STATUS OR (.$$STATUS NEQU RMSS_ACT));
        .$$STATUS
      END);
    !+
    ! If we failed the open because we might not have been granted
    ! write permission to the file, try for just read permission.
    !-
    IF NOT .STATUS
    THEN
      IF (.STATUS EQLU RMSS_PRV) AND
        (.FCB [FCB$V_OLD_FILE] OR .FAB [FAB$V_CIF]) AND
        .FAB [FAB$V_PUT] ! PUT permission requested
      THEN
        BEGIN
          !+
          ! This was a $OPEN and we did not get permission to
          ! open the file. Set the FAC bits to just GET and
          ! try the $OPEN again. It might also fail.
          !-
          FAB [FAB$B_FAC] = FAB$M_GET; ! Ask for only GET access
          FAB [FAB$V_NIL] = 0; ! Allow other sharing
          FAB [FAB$V_NAM] = 1; ! Use NAM block to reopen file
          STATUS = (
            BEGIN
              LOCAL
                $$STATUS;
              DO ($$STATUS = $OPEN (FAB=FAB [0,0,0,0]))
                UNTIL ($$STATUS NEQU RMSS_ACT);
              .$$STATUS
            END);
        END;
    !+

```

```

: 1165      1221  4      ! If open/create succeeded, enable prompting on this file if
: 1166      1222  4      ! it qualifies. Then connect the record stream.
: 1167      1223  4      !-
: 1168      1224  4
: 1169      1225  4      IF .STATUS
: 1170      1226  4      THEN
: 1171      1227  5      BEGIN
: 1172      1228  5
: 1173      1229  5      !+
: 1174      1230  5      ! Set FCBSV_FOD if file is on a file oriented device. This is
: 1175      1231  5      ! used by $PASSRMS_OP to indicate that operations that fail with
: 1176      1232  5      ! RMSS_ACT can be retried.
: 1177      1233  5      !-
: 1178      1234  5
: 1179      1235  5      IF .BLOCK [FAB [FAB$L_DEV], DEV$V_FOD;4, BYTE]
: 1180      1236  5      THEN
: 1181      1237  5          FCB [FCBSV_FOD] = 1;
: 1182      1238  5
: 1183      1239  5      !+
: 1184      1240  5      ! If file is a terminal, and if prompting hasn't been
: 1185      1241  5      ! disabled, specify that look-ahead on this file will
: 1186      1242  5      ! initiate prompt output on PROMPT_ENABLED files.
: 1187      1243  5      !-
: 1188      1244  5
: 1189      1245  5      IF .FCB [FCBSV_TEXT] AND
: 1190      1246  5      .BLOCK [FAB [FAB$L_DEV], DEV$V_TRM;4, BYTE]
: 1191      1247  5      THEN
: 1192      1248  6      BEGIN
: 1193      1249  6          FCB [FCBSV_INITIATE_PROMPT] = 1;
: 1194      1250  6
: 1195      1251  6      !+
: 1196      1252  6      ! We want to enable prompting on a file which is a terminal
: 1197      1253  6      ! and which has the LIST (CR) carriagecontrol attribute,
: 1198      1254  6      ! but which is not spooled and which can accept output.
: 1199      1255  6      !-
: 1200      1256  6
: 1201      1257  6      IF .FAB [FAB$V_CR] AND .FAB [FAB$V_PUT] AND
: 1202      1258  6      .BLOCK [FAB [FAB$L_DEV], DEV$V_ODV;4, BYTE] AND
: 1203      1259  6      NOT .BLOCK [FAB [FAB$L_DEV], DEV$V_SPL;4, BYTE]
: 1204      1260  6      THEN
: 1205      1261  7      BEGIN
: 1206      1262  7          $PASSRMS OP ($CLOSE (FAB=FAB [0,0,0,0]));
: 1207      1263  7          FAB [FAB$B_RAT] = FAB$M_PRN;      ! Set print file format
: 1208      1264  7          FAB [FAB$B_RFM] = FAB$C_VFC;      ! Set VFC format
: 1209      1265  7          FAB [FAB$B_FSZ] = 2;      ! Set 2 byte control field
: 1210      1266  7          FAB [FAB$V_NAM] = 1;      ! Use NAM block inputs
: 1211      1267  7          STATUS = $PASSRMS_OP ($OPEN (FAB=FAB [0,0,0,0])); ! Reopen file
: 1212      1268  7
: 1213      1269  7      !+
: 1214      1270  7      ! If $OPEN failed, assume that we can't set up this
: 1215      1271  7      ! file for prompting. Therefore, go back to CR format.
: 1216      1272  7      !-
: 1217      1273  7
: 1218      1274  7      IF NOT .STATUS
: 1219      1275  7      THEN
: 1220      1276  8      BEGIN
: 1221      1277  8          FAB [FAB$B_RAT] = FAB$M_CR;

```

```

: 1222
: 1223
: 1224
: 1225
: 1226
: 1227
: 1228
: 1229
: 1230
: 1231
: 1232
: 1233
: 1234
: 1235
: 1236
: 1237
: 1238
: 1239
: 1240
: 1241
: 1242
: 1243
: 1244
: 1245
: 1246
: 1247
: 1248
: 1249

```

```

      1278 8
      1279 8
P     1280 8
      1281 8
      1282 8
      1283 7
      1284 8
      1285 8
      1286 8
      1287 8
      1288 8
      1289 8
      1290 8
      1291 8
      1292 8
      1293 7
      1294 6
      1295 5
      1296 4
      1297 4
      1298 4
      1299 4
      1300 4
      1301 4
      1302 4
      1303 5
      1304 4
      1305 3

```

```

      FAB [FAB$B_RFM] = FAB$C_VAR;
      FAB [FAB$B_FSZ] = 0;
      STATUS = $PASS$RMS OP (
        $OPEN (FAB=FAB [0,0,0,0]));
      END
    ELSE
      BEGIN
        !+
        ! Open for prompting successful.
        !-
        FCB [FCB$V_PROMPT_ENABLE] = 1;
        RAB [RAB$L_RHB] = FCB [FCB$W_PROMPT_CC]; ! Control area
      END;
    END;
  END;
  END;
  !+
  ! If the OPEN/CREATE succeeded, do the CONNECT.
  !-
  IF .STATUS
  THEN
    DO (STATUS = $CONNECT (RAB=RAB [0,0,0,0]))
      UNTIL (.STATUS OR (.STATUS NEQU RMSS_ACT));
  END;

```

```

: 1251      1306      3      !+
: 1252      1307      3      !- If open/create or connect failed, signal an error
: 1253      1308      3      !-
: 1254      1309      3
: 1255      1310      3      IF NOT .STATUS
: 1256      1311      3      THEN
: 1257      1312      3          SELECTONE .STATUS OF
: 1258      1313      3          SET
: 1259      1314      3
: 1260      1315      3          [RMSS$ FNF, RMSS$ DNF, RMSS$ DEV] :
: 1261      1316      3          $PASSIO_ERROR (PASS$_FILNOTFOU);      ! File not found
: 1262      1317      3
: 1263      1318      3          [RMSS$ FNM, RMSS$ NOD, RMSS$ DIR, RMSS$ TYP, RMSS$ VER, RMSS$ SYN] :
: 1264      1319      3          $PASSIO_ERROR (PASS$_INVFILSYN);      ! Invalid filename syntax
: 1265      1320      3
: 1266      1321      3          [OTHERWISE]:
: 1267      1322      3          $PASSIO_ERROR (PASS$_ERRDUROPE);      ! Error during OPEN
: 1268      1323      3
: 1269      1324      3          TES;
: 1270      1325      3
: 1271      1326      3      END;
: 1272      1327      2
: 1273      1328      2      !+
: 1274      1329      2      !- If HISTORY := UNKNOWN has opened an existing file, indicate in the FCB
: 1275      1330      2      !-
: 1276      1331      2
: 1277      1332      2      IF .FAB [FAB$V_CIF] AND .FAB [FAB$L_STS] NEQU RMSS$_CREATED
: 1278      1333      2      THEN
: 1279      1334      2          FCB [FCB$V_OLD_FILE] = 1;
: 1280      1335      2
: 1281      1336      2      !+
: 1282      1337      2      !- If we opened an existing file, get some attributes.
: 1283      1338      2      !-
: 1284      1339      2
: 1285      1340      2      IF .FCB [FCB$V_OLD_FILE]
: 1286      1341      2      THEN
: 1287      1342      3          BEGIN
: 1288      1343      3
: 1289      1344      3              !+
: 1290      1345      3              !- Check organization.
: 1291      1346      3              !-
: 1292      1347      3
: 1293      1348      3              IF .KEYWORDS_SEEN [PASS$K_ORGANIZATION] AND
: 1294      1349      3              .FAB [FAB$B_ORG] NEQ .USR_ORG
: 1295      1350      3              THEN
: 1296      1351      3                  $PASSIO_ERROR (PASS$_ORGSPEINC,0);      ! ORGANIZATION specified inconsistent
: 1297      1352      3
: 1298      1353      4              IF .FCB [FCB$V_TEXT] AND (.FAB [FAB$B_ORG] NEQ FAB$C_SEQ)
: 1299      1354      3              THEN
: 1300      1355      3                  $PASSIO_ERROR (PASS$_TEXREQSEQ,0);      ! Textfiles require sequential organization and access
: 1301      1356      3
: 1302      1357      3              !+
: 1303      1358      3              !- Check recordtype.
: 1304      1359      3              !-
: 1305      1360      3
: 1306      1361      3              IF .KEYWORDS_SEEN [PASS$K_RECORD_TYPE]
: 1307      1362      3              THEN

```

```

1308      1363      4      BEGIN
1309      1364      4      IF .USR_RFM NEQ .FAB [FAB$B_RFM]
1310      1365      4      THEN
1311      1366      5      IF NOT ((.USR_RFM EQL FAB$C_VAR) AND
1312      1367      5      (.FAB [FAB$B_RFM] EQL FAB$C_VFC))
1313      1368      4      THEN
1314      1369      4      $PASSIO_ERROR (PASS_RECTYPINC,0); ! Inconsistent record tyoe
1315      1370      3      END;
1316      1371      3
1317      1372      3
1318      1373      3      +
1319      1374      3      | Check record length.
1320      1375      3      -
1321      1376      3
1322      1377      3      +
1323      1378      3      | If not a disk or terminal, use the blocksize as the maximum recordsize
1324      1379      3      | (if not there already).
1325      1380      3      -
1326      1381      4      IF (NOT .BLOCK [FAB [FAB$L_DEV], DEV$V_RND;4, BYTE]) AND
1327      1382      3      (NOT .BLOCK [FAB [FAB$L_DEV], DEV$V_TRM;4, BYTE])
1328      1383      3      THEN
1329      1384      3      IF .FAB [FAB$W_MRS] EQL 0
1330      1385      3      THEN
1331      1386      3      FAB [FAB$W_MRS] = .FAB [FAB$W_BLS];
1332      1387      3
1333      1388      3      IF .FAB [FAB$W_MRS] NEQ 0
1334      1389      3      THEN
1335      1390      3      IF .FAB [FAB$B_RFM] EQL FAB$C_FIX
1336      1391      4      THEN
1337      1392      4      BEGIN
1338      1393      4      IF .FCB [FCB$V_TEXT]
1339      1394      5      THEN
1340      1395      5      BEGIN
1341      1396      6      IF .KEYWORDS_SEEN [PASS$K_RECORD_LENGTH] AND
1342      1397      5      (.USR_USZ NEQ .FAB [FAB$W_MRS])
1343      1398      5      THEN
1344      1399      5      $PASSIO_ERROR (PASS_RECLEININC,0);
1345      1400      4      END
1346      1401      5      ELSE IF (.RAB [RAB$W_USZ] NEQ .FAB [FAB$W_MRS]) OR
1347      1402      4      (.USR_USZ NEQ .FAB [FAB$W_MRS])
1348      1403      4      THEN
1349      1404      4      $PASSIO_ERROR (PASS_RECLEININC,0);
1350      1405      3      END
1351      1406      4      ELSE
1352      1407      4      BEGIN
1353      1408      4      IF .FCB [FCB$V_TEXT] OR .FCB [FCB$V_VARYING]
1354      1409      5      THEN
1355      1410      5      BEGIN
1356      1411      6      IF .KEYWORDS_SEEN [PASS$K_RECORD_LENGTH] AND
1357      1412      5      (.USR_USZ GTRU .FAB [FAB$W_MRS])
1358      1413      5      THEN
1359      1414      5      $PASSIO_ERROR (PASS_RECLEININC,0);
1360      1415      4      END
1361      1416      5      ELSE IF (.RAB [RAB$W_USZ] GTRU .FAB [FAB$W_MRS]) OR
1362      1417      4      (.USR_USZ GTRU .FAB [FAB$W_MRS])
1363      1418      4      THEN
1364      1419      3      $PASSIO_ERROR (PASS_RECLEININC,0);
1364      1419      3      END;

```



```

: 1411 1465 2      !+
: 1412 1466 2      ! If the file is indexed organization, get information about the defined
: 1413 1467 2      ! keys.
: 1414 1468 2      !-
: 1415 1469 2
: 1416 1470 2      IF .FAB [FAB$B_ORG] EQL FAB$C_IDX
: 1417 1471 2      THEN
: 1418 1472 2          CHECK_KEY_XABS (PFV [PFV$R_PFV],
: 1419 1473 2          FCB [FCB$R_FCB],
: 1420 1474 2          XAB_SUM,
: 1421 1475 2          XABKEY_ADDR,
: 1422 1476 2          XABKEY_SIZE);
: 1423 1477 2
: 1424 1478 2      !+
: 1425 1479 2      ! If this is the implicit open of INPUT or OUTPUT, set the status bits to
: 1426 1480 2      ! cause an implicit RESET or REWRITE appropriately.
: 1427 1481 2      !-
: 1428 1482 2
: 1429 1483 2      IF (.FILE_TYPE EQL K_INPUT) AND NOT .PAS$$GV_INPUT_OPENED
: 1430 1484 2      THEN
: 1431 1485 2          BEGIN
: 1432 1486 2          PAS$$GV_INPUT_OPENED = 1;          ! INPUT has been opened
: 1433 1487 2          PFV [PFV$V_VA[ID]] = 0;          ! File variable not valid
: 1434 1488 2          FCB [FCB$V_LAZY] = 1;          ! Implicit GET on next access
: 1435 1489 2          FCB [FCB$V_INSPECTION] = 1;      ! In Inspection mode
: 1436 1490 2          END
: 1437 1491 2      ELSE IF (.FILE_TYPE EQL K_OUTPUT) AND NOT .PAS$$GV_OUTPUT_OPENED
: 1438 1492 2      THEN
: 1439 1493 2          BEGIN
: 1440 1494 2          PAS$$GV_OUTPUT_OPENED = 1;          ! OUTPUT has been opened
: 1441 1495 2          FCB [FCB$V_GENERATION] = 1;      ! In Generation mode
: 1442 1496 2          FCB [FCB$V_EOF] = 1;          ! At EOF
: 1443 1497 2          PFV [PFV$V_EOF_DEFINED] = 1;    ! EOF function defined
: 1444 1498 2          PFV [PFV$V_VALID] = 1;          ! File variable valid
: 1445 1499 2          PFV [PFV$V_DFB] = 0;          ! File buffer undefined
: 1446 1500 2          RAB [RAB$V_TPT] = 1;          ! Truncate on first $PUT
: 1447 1501 2          END
: 1448 1502 2      ELSE
: 1449 1503 2          BEGIN
: 1450 1504 2          !+
: 1451 1505 2          ! User must do RESET or REWRITE before using file.
: 1452 1506 2          !-
: 1453 1507 2          PFV [PFV$V_EOF_DEFINED] = 0;    ! EOF function undefined
: 1454 1508 2          PFV [PFV$V_VALID] = 1;          ! File variable valid
: 1455 1509 2          PFV [PFV$V_DFB] = 0;          ! File buffer undefined
: 1456 1510 2          END;
: 1457 1511 2
: 1458 1512 2      !+
: 1459 1513 2      ! Allocate the record buffer, if necessary, and fill in the UBF address.
: 1460 1514 2      !-
: 1461 1515 2
: 1462 1516 2      IF .FCB [FCB$V_TEXT]
: 1463 1517 2      THEN
: 1464 1518 2          BEGIN
: 1465 1519 2          RAB [RAB$L_UBF] = PAS$$GET_VM (PFV [PFV$R_PFV], .RAB [RAB$W_USZ]);
: 1466 1520 2          FCB [FCB$V_DYNAMIC_UBF] = T;
: 1467 1521 2          END

```

```

: 1468      1522  2      ELSE IF .FCB [FCB$V_VARYING]
: 1469      1523  2      THEN
: 1470      1524  2      RAB [RAB$L_UBF] = .PFV [PFV$A_BUFFER] + 2
: 1471      1525  2      ELSE
: 1472      1526  2      RAB [RAB$L_UBF] = .PFV [PFV$A_BUFFER];
: 1473      1527  2
: 1474      1528  2      !+
: 1475      1529  2      ! Set up record pointers in FCB
: 1476      1530  2      !-
: 1477      1531  2
: 1478      1532  2      FCB [FCB$A_RECORD_BEG] = .RAB [RAB$L_UBF];
: 1479      1533  2      FCB [FCB$A_RECORD_CUR] = .RAB [RAB$L_UBF];
: 1480      1534  2      FCB [FCB$L_RECORD_LEN] = .RAB [RAB$W_USZ];
: 1481      1535  2      FCB [FCB$A_RECORD_END] = .FCB [FCB$A_RECORD_CUR] + .FCB [FCB$L_RECORD_LEN];
: 1482      1536  2
: 1483      1537  2      !+
: 1484      1538  2      ! If the file has no name (TMD) then zero the expanded and resultant
: 1485      1539  2      ! name strings.
: 1486      1540  2      !-
: 1487      1541  2
: 1488      1542  2      IF .FAB [FAB$V_TMD]
: 1489      1543  2      THEN
: 1490      1544  3      BEGIN
: 1491      1545  3      NAM [NAM$B_ESL] = 0;
: 1492      1546  3      NAM [NAM$B_RSL] = 0;
: 1493      1547  3      END
: 1494      1548  3
: 1495      1549  2      ELSE
: 1496      1550  2
: 1497      1551  2      !+
: 1498      1552  2      ! Allocate space for the resultant name string, move the local string
: 1499      1553  2      ! to the allocated one, and change the RSA pointer.
: 1500      1554  2      !-
: 1501      1555  2
: 1502      1556  3      BEGIN
: 1503      1557  3      NAM [NAM$L_RSA] = PASS$GET_VM (PFV [PFV$R_PFV], .NAM [NAM$B_RSL]);
: 1504      1558  3      CH$MOVE (.NAM [NAM$B_RSL], _RESULT_NAME_STRING, .NAM [NAM$L_RSA]);
: 1505      1559  3      FCB [FCB$V_DYNAMIC_RSN] = 1; ! Indicate dynamic resultant name
: 1506      1560  2      END;
: 1507      1561  2
: 1508      1562  2      !+
: 1509      1563  2      ! Set linelimit.
: 1510      1564  2      !-
: 1511      1565  2
: 1512      1566  2      FCB [FCB$L_LINELIMIT] = -1; ! Initially indicate infinite limit
: 1513      1567  2
: 1514      1568  2      IF .FCB [FCB$V_TEXT] ! If textfile then look at PASS$LINELIMIT
: 1515      1569  2      THEN
: 1516      1570  3      BEGIN
: 1517      1571  3      LOCAL
: 1518      1572  3      LOGNAM_DSC: BLOCK [8, BYTE], ! Descriptor for logical name
: 1519      1573  3      LIMIT_DSC: BLOCK [8, BYTE]; ! Descriptor for limit string
: 1520      1574  3
: 1521      1575  3
: 1522      1576  3      !+
: 1523      1577  3      ! Translate the logical name PASS$LINELIMIT to find a possible
: 1524      1578  3      ! linelimit.

```

```

: 1525      1579      3      !-
: 1526      1580      3
: 1527      1581      3      LOGNAM_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
: 1528      1582      3      LOGNAM_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
: 1529      1583      3      LOGNAM_DSC [DSC$W_LENGTH] = %CHARCOUNT ('PASS$LINELIMIT');
: 1530      1584      3      LOGNAM_DSC [DSC$A_POINTER] = UPLIT BYTE ('PASS$LINELIMIT');
: 1531      1585      3      LIMIT_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
: 1532      1586      3      LIMIT_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
: 1533      1587      3      LIMIT_DSC [DSC$W_LENGTH] = LNM$C_NAMLENGTH;      ! Maximum resultant string length
: 1534      1588      3      LIMIT_DSC [DSC$A_POINTER] = RESULT_NAME_STRING;
: 1535      1589      3
: 1536      P 1590      3      IF STRNLOG (LOGNAM = LOGNAM_DSC
: 1537      P 1591      3          RSLLEN = LIMIT_DSC [DSC$W_LENGTH], ! Returned length
: 1538      1592      4          RSLBUF = LIMIT_DSC)
: 1539      1593      3      EQLU SSS$NORMAL      ! Translation must succeed to continue
: 1540      1594      3      THEN
: 1541      1595      4      BEGIN
: 1542      1596      4      +
: 1543      1597      4      | Try to translate string as a signed decimal number. If
: 1544      1598      4      | successful, store as linelimit. Otherwise set linelimit
: 1545      1599      4      | back to -1.
: 1546      1600      4      |
: 1547      1601      4      |
: 1548      1602      4      IF NOT OT$CVT_TI_L (LIMIT_DSC, FCB [FCB$L_LINELIMIT],
: 1549      1603      4          4,      ! Value size is 4 bytes
: 1550      1604      4          1)      ! Ignore blanks
: 1551      1605      4      THEN
: 1552      1606      4      FCB [FCB$L_LINELIMIT] = -1;
: 1553      1607      3      END;
: 1554      1608      2      END;
: 1555      1609      2
: 1556      1610      2      !+
: 1557      1611      2      | Mark XAB chain invalid.
: 1558      1612      2      |
: 1559      1613      2      |
: 1560      1614      2      FAB [FAB$L_XAB] = 0;
: 1561      1615      2
: 1562      1616      2      !+
: 1563      1617      2      | Deallocate any KEY XABs.
: 1564      1618      2      |
: 1565      1619      2      |
: 1566      1620      2      IF .XABKEY_ADDR NEQ 0
: 1567      1621      2      THEN
: 1568      1622      2      PASS$FREE_VM (.XABKEY_SIZE, XABKEY_ADDR);
: 1569      1623      2
: 1570      1624      2      !+
: 1571      1625      2      | Mark PFV to indicate file open.
: 1572      1626      2      |
: 1573      1627      2      |
: 1574      1628      2      PFV [PFV$V_OPEN] = 1;
: 1575      1629      2
: 1576      1630      2      !+
: 1577      1631      2      | Add file to list of open files.
: 1578      1632      2      |
: 1579      1633      2      |
: 1580      1634      2      PASS$ADD_FILE (FCB [FCB$R_FCB]);
: 1581      1635      2

```

PASS\$OPEN2
1-015

OPEN procedure
PASS\$OPEN - Open a file

E 13
16-Sep-1984 01:46:15
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742
[PASRTL.SRC]PASOPEN2.B32;1

Page 35
(10)

: 1582
: 1583
: 1584

1636 2 RETURN;
1637 2
1638 1 END;

. End of routine PASS\$OPEN

```

00 00078 PASS$AB_KEYWD_NAME_TABLE::
OD OD OD OB OB OB OB OB 07 03 03 03 03 02 01 00079 .BYTE 0
1B 1B 1B 1A 19 13 13 13 13 13 13 10 10 00088 .BYTE 1, 2, 3, 3, 3, 3, 7, 8, 8, 8, 11, 11, 13, -
54 55 50 4E 49 24 53 41 50 00096 .BYTE 0[8]
54 55 50 4E 49 24 53 59 53 0009E .BYTE 11, 11, 11
54 55 50 54 55 4F 24 53 41 50 000A1 P.AAA: .ASCII \PASS$INPUT\
54 55 50 54 55 4F 24 53 41 50 000AA P.AAB: .ASCII \SYSS$INPUT\
54 55 50 54 55 4F 24 53 41 50 000B3 P.AAC: .ASCII \PASS$OUTPUT\
50 5D 54 2E 45 4D 41 4E 4F 4E 53 41 50 000BD P.AAD: .ASCII \SYSS$OUTPUT\
54 49 4D 49 4C 45 4E 49 4C 24 53 41 50 000C7 P.AAE: .ASCII \PASNONAME.TMP\
54 49 4D 49 4C 45 4E 49 4C 24 53 41 50 000D4 P.AAF: .ASCII \.DAT\
54 49 4D 49 4C 45 4E 49 4C 24 53 41 50 000DB P.AAG: .ASCII \PASS$LINELIMIT\

.EXTRN PASS$GET_VM, SYSS$DCLEXH
.EXTRN PASS$K_INVRECLN
.EXTRN PASS$K_INVARGPAS
.EXTRN PASS$K_INVFILSYN
.EXTRN SYSS$TRNLOG, PASS$K_TEXREQSEQ
.EXTRN PASS$K_ACCMETINC
.EXTRN PASS$K_FILNAMREQ
.EXTRN PASS$K_RECLNINC
.EXTRN SYSS$OPEN, SYSS$CREATE
.EXTRN SYSS$CLOSE, SYSS$CONNECT
.EXTRN PASS$K_FILNOTFOU
.EXTRN PASS$K_ERRDUROPE
.EXTRN PASS$K_ORGSPEINC
.EXTRN PASS$K_RECTYPINC
.EXTRN OTSS$CVT_TITL, PASS$FREE_VM
.EXTRN PASS$ADD_FILE
.WEAK PASS$FV_INPUT, PASS$FV_OUTPUT

0F7C 00000 .ENTRY PASS$OPEN, Save R2,R3,R4,R5,R6,R8,R9,R10,-
5E FE88 CE 9E 00002 MOVAB -376(SP), SP
F4 AD 7C 00007 CLRQ XABKEY SIZE
FC AD D4 0000A CLRL PFV_ADDR
6D 08F2 CF DE 0000D MOVAL 1605, (FP)
6E 00 2C 00012 MOVCS #0, (SP). #0, #44, $RMS_PTR
C8 AD 00017
CC AD 2C1D 8F B0 00019 MOVW #11293, $RMS_PTR
AD BC AD 9E 0001F MOVAB XAB_SUM, $RMS_PTR+4
6E 00 2C 00024 MOVCS #0, (SP), #0, #12, $RMS_PTR
BC AD BC AD 00029
0C16 8F B0 0002B MOVW #3094, $RMS_PTR
04 AC D0 00031 MOVL IN PFV, PFV
6E 04 A6 9E 00035 MOVAB 4(PFV), (SP)
OA 00 BE 1C E1 00039 BBC #28, @0(SP), 15
08 A6 56 C0 0003E ADDL2 PFV, 8(PFV)
00 BE 01 1C 00  F0 00042 INSV #0, #28, #1, @0(SP)

```

03	00	BE	1B	E5	00048	1\$:	BBCC	#27, @0(SP), 2\$	0504		
		66	56	C0	0004D		ADDL2	PFV, (PFV)	0506		
21	00000000'	EF	00	F2	00050	2\$:	BBSS	#0, EXITH_DECLARED, 3\$	0512		
			14	DD	00058		PUSHL	#20	0522		
	00000000G	00	01	FB	0005A		CALLS	#1, PASS\$GET_VM			
		04	CF	9E	00061		MOVAB	EXIT_HANDLER, 4(EXITH_CONTROL_BLOCK)	0523		
		08	01	D0	00067		MOVL	#1, 8(EXITH_CONTROL_BLOCK)	0524		
		0C	A0	9E	0006B		MOVAB	16(R0), 12(EXITH_CONTROL_BLOCK)	0525		
	00000000G	00	50	DD	00070		PUSHL	EXITH_CONTROL_BLOCK	0526		
		50	01	FB	00072		CALLS	#1, SY\$DCLEXR			
		00000000G	00	9E	00079	3\$:	MOVAB	PASS\$V INPUT, R0	0533		
			56	D1	00080		CMPL	PFV, R0			
			06	12	00083		BNEQ	4\$			
		0C	01	CE	00085		MNEGL	#1, FILE_TYPE	0535		
			15	11	00089		BRB	6\$			
		50	00	9E	0008B	4\$:	MOVAB	PASS\$V OUTPUT, R0	0536		
		50	56	D1	00092		CMPL	PFV, R0			
			06	12	00095		BNEQ	5\$			
		0C	01	D0	00097		MOVL	#1, FILE_TYPE	0538		
			03	11	00096		BRB	6\$			
			0C	AE	D4	0009D	5\$:	CLRL	FILE_TYPE	0540	
		00000000G	7E	0138	8F	3C	000A0	6\$:	MOVZWL	#312, -(SP)	0547
			00	01	FB	000A5		CALLS	#1, PASS\$GET_VM		
			57	44	A0	9E	000AC		MOVAB	68(R0), FCB	
			67	4401	8F	B0	000B0		MOVW	#17409, (RAB)	0553
			5B	44	A7	9E	000B5		MOVAB	68(RAB), R11	0555
			6B		03	90	000B9		MOVB	#3, (R11)	
		45	A7	50	8F	90	000BC		MOVB	#80, 69(RAB)	0556
	0094	C7	6002	8F	B0	000C1		MOVW	#24578, 148(RAB)	0557	
	00A0	C7	34	AE	9E	000C8		MOVAB	RESULT_NAME_STRING, 160(RAB)	0559	
	009E	C7		01	8E	000CE		MNEGB	#1, 158(RAB)	0560	
	04	AE	0098	C7	9E	000D3		MOVAB	152(RAB), 4(SP)	0561	
	04	BE	34	AE	9E	000D9		MOVAB	RESULT_NAME_STRING, @4(SP)		
	0096	C7		01	8E	000DE		MNEGB	#1, 150(RAB)	0562	
	3C	A7		5B	D0	000E3		MOVL	R11, 60(RAB)	0563	
	6C	A7	0094	C7	9E	000E7		MOVAB	148(RAB), 108(RAB)	0564	
	68	A7	C8	AD	9E	000ED		MOVAB	XAB_FHC, 104(RAB)	0565	
	DC	A7		56	D0	000F2		MOVL	PFV, -36(RAB)	0571	
	0C	A6		57	D0	000F6		MOVL	RAB, 12(PFV)	0577	
00	BE	01	1E	01	F0	000FA		INSV	#1, #30, #1, @0(SP)	0578	
			FC	AD	56	D0	00100		MOVL	PFV, PFV_ADDR	0579
			52	08	A6	D0	00104		MOVL	8(PFV), PFD	0589
			59	F8	A7	9E	00108		MOVAB	-8(RAB), R9	0590
			69	04	A2	B0	0010C		MOVW	4(PFD), (R9)	
		E4	A7	52	D0	00110		MOVL	PFD, -28(RAB)	0591	
			55	20	A7	9E	00114		MOVAB	32(RAB), R5	0607
			21	69	E8	00118		BLBS	(R9), 9\$	0599	
	0000FFFF	8F	08	A2	D1	0011B		CMPL	8(PFD), #65535	0602	
				06	1B	00123		BLEQU	7\$		
			08	A2	D0	00125		PUSHL	8(PFD)	0604	
				0111	31	00128		BRW	25\$		
07		69	02	E1	0012B	7\$:	BBC	#2, (R9), 8\$	0605		
65	08	A2	02	A3	0012F		SUBW3	#2, 8(PFD), (R5)	0607		
			0A	11	00134		BRB	10\$			
		65	08	A2	B0	00136	8\$:	MOVW	8(PFD), (R5)	0609	
			04	11	0013A		BRB	10\$	0599		
		65	85	8F	9B	0013C	9\$:	MOVZBW	#133, (R5)	0612	

		08	AE		65	B0	00140	10\$:	MOVW	(R5), USR_USZ	0614
				2C	AE	D4	00144		CLRL	KEYWORDS_SEEN	0621
		30	AE	01FF	8F	AA	00147		BICW2	#511, KEYWORDS_SEEN+4	0622
			58		6C	9A	0014D		MOVZBL	(AP), R8	0623
			58		02	C2	00150		SUBL2	#2, R8	
			54		01	CE	00153		MNEGL	#1, I	
					019F	31	00156	11\$:	BRW	59\$	
			53	08	AC44	D0	00159	12\$:	MOVL	KEYWORDS[I], KEYWD_VALUE	0634
					64	15	0015E		BLEQ	14\$	0635
			28		53	D1	00160		C MPL	KEYWD_VALUE, #40	
					5F	14	00163		BGTR	14\$	
			50	FE29	CF43	9A	00165		MOVZBL	PASS\$AB KEYWD_NAME_TABLE[KEYWD_VALUE], R0	0639
	E6	2C	AE		50	E2	0016B		BBSS	R0, KEYWORDS_SEEN, -11\$	
	27		01		53	CF	00170		CASEL	KEYWD_VALUE, -#1, #39	0641
00A3	0099		0074		0058		00174	13\$:	.WORD	15\$-13\$,-	
00E5	00B7		00AD		009F		0017C			16\$-13\$,-	
010B	0105		00FF		00F0		00184			20\$-13\$,-	
012F	0184		0129		0123		0018C			22\$-13\$,-	
015C	0152		013A		0134		00194			21\$-13\$,-	
0158	014E		0147		0140		0019C			23\$-13\$,-	
0176	0170		0163		0182		001A4			24\$-13\$,-	
0050	0050		0050		017C		001AC			27\$-13\$,-	
0050	0050		0050		0050		001B4			29\$-13\$,-	
011D	0117		0111		0050		001BC			30\$-13\$,-	
										31\$-13\$,-	
										32\$-13\$,-	
										37\$-13\$,-	
										38\$-13\$,-	
										59\$-13\$,-	
										40\$-13\$,-	
										42\$-13\$,-	
										43\$-13\$,-	
										47\$-13\$,-	
										50\$-13\$,-	
										44\$-13\$,-	
										45\$-13\$,-	
										46\$-13\$,-	
										49\$-13\$,-	
										58\$-13\$,-	
										52\$-13\$,-	
										54\$-13\$,-	
										55\$-13\$,-	
										57\$-13\$,-	
										14\$-13\$,-	
										14\$-13\$,-	
										14\$-13\$,-	
										14\$-13\$,-	
										14\$-13\$,-	
										14\$-13\$,-	
										14\$-13\$,-	
										14\$-13\$,-	
										14\$-13\$,-	
										33\$-13\$,-	
										35\$-13\$,-	
										36\$-13\$	
										-(SP)	0861
		7E		00G	7E	D4	001C4	14\$:	CLRL		
					8F	9A	001C6		MOVZBL	#PASSK_INVARGPAS, -(SP)	
					30	11	001CA		BRB	18\$	

		54	D6	001CC	15\$:	INCL	I		0648
	00FF	53	08 AC44	F7 001CE		CVTLW	KEYWORDS[I], FNS		
		8F		53 B1 001D3		CMPW	FNS, #255		0649
				1C 1A 001D8		BGTRU	17\$		
	78	A7		53 90 001DA		MOVB	FNS, 120(RAB)		0652
				54 D6 001DE		INCL	I		0653
	70	A7	08 AC44	D0 001E0		MOVL	KEYWORDS[I], 112(RAB)		
				7A 11 001E6		BRB	28\$		0641
				54 D6 001E8	16\$:	INCL	I		0660
	00FF	53	08 AC44	F7 001EA		CVTLW	KEYWORDS[I], DNS		
		8F		53 B1 001EF		CMPW	DNS, #255		0661
				09 1B 001F4		BLEQU	19\$		
				7E D4 001F6	17\$:	CLRL	-(SP)		0663
	7E		00G	8F 9A 001F8		MOVZBL	#PASSK_INVFILSYN, -(SP)		
				05A2 31 001FC	18\$:	BRW	147\$		
	79	A7		53 90 001FF	19\$:	MOVB	DNS, 121(RAB)		0664
				54 D6 00203		INCL	I		0665
	74	A7	08 AC44	D0 00205		MOVL	KEYWORDS[I], 116(RAB)		
				7C 11 0020B		BRB	34\$		0641
	FC	A7		10 88 0020D	20\$:	BISB2	#16, -4(RAB)		0670
				04 11 00211		BRB	22\$		0680
	4B	A7		02 88 00213	21\$:	BISB2	#2, 75(RAB)		0689
			5A	A7 9E 00217	22\$:	MOVAB	90(RAB), R0		0690
		50		1F 88 0021B		BISB2	#31, (R0)		0694
		60		0080 31 0021E		BRW	39\$		0641
	FC	A7		18 88 00221	23\$:	BISB2	#24, -4(RAB)		0700
				02 88 00225		BISB2	#2, 90(RAB)		0701
	5A	A7		7B 11 00229		BRB	41\$		0641
				54 D6 0022B	24\$:	INCL	I		0706
	0000FFFF	8F	08 AC44	D1 0022D		CMP	KEYWORDS[I], #65535		
				12 1B 00236		BLEQU	26\$		
			08 AC44	DD 00238		PUSHL	KEYWORDS[I]		0708
				01 DD 0023C	25\$:	PUSHL	#1		
			00G	8F 9A 0023E		MOVZBL	#PASSK_INVRECLN, -(SP)		
	00000000G	00		03 FB 00242		CALLS	#3, PASS\$SIGNAL		
				04 00249		RET			
	08	AE	08 AC44	F7 0024A	26\$:	CVTLW	KEYWORDS[I], USR_USZ		0709
		77		69 E9 00250		BLBC	(R9), 48\$		0717
		65	08	AE B0 00253		MOVW	USR_USZ, (R5)		0719
				7C 11 00257		BRB	51\$		0641
	FC	A7		01 88 00259	27\$:	BISB2	#1, -4(RAB)		0724
	48	A7	40	8F 88 0025D		BISB2	#64, 72(RAB)		0725
				7E 11 00262	28\$:	BRB	53\$		0641
	FC	A7		02 88 00264	29\$:	BISB2	#2, -4(RAB)		0730
				02 88 00268		MOVAB	-40(R7), 48(RAB)		0731
	30	A7	D8	A7 9E 00268		BISB2	#16, 4(RAB)		0732
				10 88 0026D		BRB	56\$		0641
	04	A7		7B 11 00271		BRB	56\$		0641
	FC	A7		04 88 00273	30\$:	BISB2	#4, -4(RAB)		0737
				7F 11 00277		BRB	59\$		0641
	63	A7		01 90 00279	31\$:	MOVB	#1, 99(RAB)		0742
				79 11 0027D		BRB	59\$		0641
	63	A7		02 90 0027F	32\$:	MOVB	#2, 99(RAB)		0747
				73 11 00283		BRB	59\$		0641
	63	A7		04 90 00285	33\$:	MOVB	#4, 99(RAB)		0752
				6D 11 00289	34\$:	BRB	59\$		0641
	63	A7		06 90 0028B	35\$:	MOVB	#6, 99(RAB)		0757
				67 11 0028F		BRB	59\$		0641

	63	A7		05	90	00291	36\$:	MOVB	#5, 99(RAB)	0762
				61	11	00295		BRB	59\$	0641
	62	A7		02	88	00297	37\$:	BISB2	#2, 98(RAB)	0767
				5B	11	00298		BRB	59\$	0641
	62	A7		01	88	0029D	38\$:	BISB2	#1, 98(RAB)	0772
				55	11	002A1	39\$:	BRB	59\$	0641
			61	A7	94	002A3	40\$:	CLRB	97(RAB)	0780
				50	11	002A6	41\$:	BRB	59\$	0641
	61	A7		10	90	002A8	42\$:	MOVB	#16, 97(RAB)	0785
				4A	11	002AC		BRB	59\$	0641
	61	A7		20	90	002AE	43\$:	MOVB	#32, 97(RAB)	0790
				44	11	002B2		BRB	59\$	0641
	FC	A7	A0	8F	88	002B4	44\$:	BISB2	#160, -4(RAB)	0806
				3D	11	002B9		BRB	59\$	0641
	FC	A7	80	8F	88	002BB	45\$:	BISB2	#128, -4(RAB)	0811
				0E	11	002C0		BRB	50\$	0812
	FD	A7		01	88	002C2	46\$:	BISB2	#1, -3(RAB)	0817
	FC	A7		20	88	002C6	47\$:	BISB2	#32, -4(RAB)	0818
				2C	11	002CA	48\$:	BRB	59\$	0641
	FD	A7		01	88	002CC	49\$:	BISB2	#1, -3(RAB)	0823
	FC	A7	40	8F	88	002D0	50\$:	BISB2	#64, -4(RAB)	0824
				21	11	002D5	51\$:	BRB	59\$	0641
				54	D6	002D7	52\$:	INCL	I	0829
		52	08	AC44	D0	002D9		MOVL	KEYWORDS[I], USER_ACTION_BPV	
	FD	A7		02	88	002DE		BISB2	#2, -3(RAB)	0830
				14	11	002E2	53\$:	BRB	59\$	0641
	5B	A7		20	88	002E4	54\$:	BISB2	#32, 91(RAB)	0835
				0E	11	002E8		BRB	59\$	0641
	5B	A7		02	88	002EA	55\$:	BISB2	#2, 91(RAB)	0840
				08	11	002EE	56\$:	BRB	59\$	0641
	5B	A7		0F	88	002F0	57\$:	BISB2	#15, 91(RAB)	0848
				02	11	002F4		BRB	59\$	0641
				54	D6	002F6	58\$:	INCL	I	0857
FESB	54	01		58	F1	002F8	59\$:	ACBL	R8, #1, I, 12\$	0623
	03	2C	AE	01	E1	002FE		BBC	#1, KEYWORDS_SEEN, 60\$	0875
				0095	31	00303		BRW	67\$	
			0C	AE	D5	00306	60\$:	TSTL	FILE_TYPE	0886
				69	13	00309		BEQL	65\$	
	FFFFFFF	8F	0C	AE	D1	0030B		CMP	FILE_TYPE, #-1	0895
				11	12	00313		BNEQ	61\$	
	24	AE		09	B0	00315		MOVW	#9, LOGNAM_DSC	0898
	28	AE	FC9F	CF	9E	00319		MOVAB	P.AAA, LOGNAM_DSC+4	0899
		53	FCA2	CF	9E	0031F		MOVAB	P.AAB, SUBSTITUTE_NAME	0900
				0F	11	00324		BRB	62\$	0895
	24	AE		0A	B0	00326	61\$:	MOVW	#10, LOGNAM_DSC	0904
	28	AE	FCA0	CF	9E	0032A		MOVAB	P.AAC, LOGNAM_DSC+4	0905
		53	FCA4	CF	9E	00330		MOVAB	P.AAD, SUBSTITUTE_NAME	0906
	26	AE	010E	8F	B0	00335	62\$:	MOVW	#270, LOGNAM_DSC+2	0909
	1C	AE	010E00FF	8F	D0	0033B		MOVL	#17694975, RSLNAM_DSC	0912
	20	AE	34	AE	9E	00343		MOVAB	RESULT_NAME_STRING, RSLNAM_DSC+4	0913
				7E	7C	00348		CLRQ	-(SP)	0914
				7E	D4	0034A		CLRL	-(SP)	
			28	AE	9F	0034C		PUSHAB	RSLNAM_DSC	
				7E	D4	0034F		CLRL	-(SP)	
			38	AE	9F	00351		PUSHAB	LOGNAM_DSC	
0000000G	00			06	FB	00354		CALLS	#6, SYS\$TRNLOG	
00000629	8F			50	D1	0035B		CMPL	R0, #1577	

			06	12	00362	BNEQ	63\$		
	70	A7	53	D0	00364	MOVL	SUBSTITUTE_NAME, 112(RAB)	0920	
			05	11	00368	BRB	64\$	0921	
	70	A7	28	AE	D0 0036A	63\$:	MOVL	LOGNAM_DSC+4, 112(RAB)	0928
	78	A7	24	AE	90 0036F	64\$:	MOVVB	LOGNAM_DSC, 120(RAB)	0929
			70	A7	D5 00374	65\$:	TSTL	112(RAB)	0933
			22	12	00377	BNEQ	67\$		
10		69	05	E1	00379	BBC	#5, (R9), 66\$	0935	
		50	E4	A7	D0 0037D	MOVL	-28(RAB), PFD	0940	
	78	A7	OC	A0	90 00381	MOVVB	12(PFD), 120(RAB)	0941	
	70	A7	OD	A0	9E 00386	MOVAB	13(R0), 112(RAB)	0942	
			OE	11	0038B	BRB	67\$	0935	
	48	A7	10	88	0038D	66\$:	BISB2	#16, 72(RAB)	0946
	78	A7	OD	90	00391	MOVVB	#13, 120(RAB)	0947	
	70	A7	FC49	CF	9E 00395	MOVAB	P.AAE, 112(RAB)	0948	
OE	2C	AE	02	E0	0039B	67\$:	BBS	#2, KEYWORDS_SEEN, 68\$	0956
OA		69	05	E1	003A0	BBC	#5, (R9), 68\$		
	79	A7	04	90	003A4	MOVVB	#4, 121(RAB)	0959	
	74	A7	FC43	CF	9E 003A8	MOVAB	P.AAF, 116(RAB)	0960	
07	2C	AE	03	E0	003AE	68\$:	BBS	#3, KEYWORDS_SEEN, 69\$	0967
		50	5A	A7	9E 003B3	MOVAB	90(RAB), R0	0973	
		60	1F	88	003B7	BISB2	#31, (R0)	0977	
	FC	OB	2D	AE	E8 003BA	69\$:	BLBS	KEYWORDS_SEEN+1, 70\$	0984
	48	A7	01	88	003BE	BISB2	#1, -4(RAB)	0990	
			40	8F	88 003C2	BISB2	#64, 72(RAB)	0991	
			07	11	003C7	BRB	71\$	0984	
		04	69	E9	003C9	70\$:	BLBC	(R9), 71\$	0993
		4D	FC	A7	E9 003CC	BLBC	-4(RAB), 79\$		
11	2D	AE	03	E0	003D0	71\$:	BBS	#3, KEYWORDS_SEEN+1, 74\$	1001
		04	69	E8	003D5	BLBS	(R9), 72\$	1003	
06		69	02	E1	003D8	BBC	#2, (R9), 73\$		
	63	A7	02	90	003DC	72\$:	MOVVB	#2, 99(RAB)	1005
			04	11	003E0	BRB	74\$		
	63	A7	01	90	003E2	73\$:	MOVVB	#1, 99(RAB)	1007
		54	63	A7	9E 003E6	74\$:	MOVAB	99(RAB), R4	1008
	10	AE	64	90	003EA	MOVVB	(R4), USR RFM		
08	2D	AE	05	E0	003EE	BBS	#5, KEYWORDS_SEEN+1, 76\$	1014	
		04	69	E8	003F3	BLBS	(R9), 75\$	1016	
04		69	02	E1	003F6	BBC	#2, (R9), 76\$		
	62	A7	02	88	003FA	75\$:	BISB2	#2, 98(RAB)	1018
09	2E	AE	03	E0	003FE	76\$:	BBS	#3, KEYWORDS_SEEN+2, 77\$	1024
04	48	A7	04	E0	00403	BBS	#4, 72(RAB), 77\$	1026	
	FC	A7	20	88	00408	BISB2	#32, -4(RAB)	1028	
		05	2E	AE	E8 0040C	77\$:	BLBS	KEYWORDS_SEEN+2, 78\$	1034
			61	A7	94 00410	CLRB	97(RAB)	1036	
			10	11	00413	BRB	80\$		
		0D	69	E9	00415	78\$:	BLBC	(R9), 80\$	1037
			61	A7	95 00418	TSTB	97(RAB)		
			08	13	0041B	BEQL	80\$		
			7E	D4	0041D	79\$:	CLRL	-(SP)	1039
		7E	00G	8F	9A 0041F	MOVZBL	#PASSK_TEXREQSEQ, -(SP)		
			7F	11	00423	BRB	90\$		
		53	61	A7	9E 00425	80\$:	MOVAB	97(RAB), R3	1040
	14	AE	63	90	00429	MOVVB	(R3), USR ORG		
		14	2E	AE	E9 0042D	BLBC	KEYWORDS_SEEN+2, 82\$	1051	
			12	12	00431	BNEQ	82\$	1052	
		09	5B	A7	E8 00433	BLBS	91(RAB), 81\$	1054	

		0A	5A	A7	E9	00437		BLBC	90(RAB), 82\$	1055
05	5B	A7		C1	E1	0043B		BBC	#1, 91(RAB), 82\$	
	5B	A7	40	8F	88	00440	81\$:	BISB2	#64, 91(RAB)	1057
0F	2F	AE		03	E0	00445	82\$:	BBS	#3, KEYWORDS_SEEN+3, 84\$	1059
06	FC	A7		03	E1	0044A		BBC	#3, -4(RAB), -83\$	1061
	5B	A7		02	88	0044F		BISB2	#2, 91(RAB)	1063
				04	11	00453		BRB	84\$	
	5B	A7		20	88	00455	83\$:	BISB2	#32, 91(RAB)	1065
21	58		FC	A7	9E	00459	84\$:	MOVAB	-4(RAB), R8	1075
	68			04	E0	0045D		BBS	#4, (R8), 88\$	
0E	1E			68	E8	00461		BLBS	(R8), 38\$	1078
	68			01	E1	00464		BBC	#1, (R8), 85\$	1079
	10			63	91	00468		CMPB	(R3), #16	1080
				15	13	0046B		BEQL	88\$	
				63	95	0046D		TSTB	(R3)	1081
				05	12	0046F		BNEQ	85\$	
		01		64	91	00471		CMPB	(R4), #1	
				0C	13	00474		BEQL	88\$	
03	68			02	E0	00476	85\$:	BBS	#2, (R8), 87\$	1082
				031E	31	0047A	86\$:	BRW	146\$	
	20			63	91	0047D	87\$:	CMPB	(R3), #32	1083
				F8	12	00480		BNEQ	86\$	
	5A		48	A7	9E	00482	88\$:	MOVAB	72(RAB), R10	1091
1C	6A			04	E1	00486		BBC	#4, (R10), 91\$	
10	68			05	E0	0048A		BBS	#5, (R8), 89\$	1092
	0C		01	A8	E8	0048E		BLBS	1(R8), 89\$	
				68	95	00492		TSTB	(R8)	
				08	19	00494		BLSS	89\$	
04	68			04	E0	00496		BBS	#4, (R8), 89\$	1093
08	6A			19	E1	0049A		BBC	#25, (R10), 91\$	
				7E	D4	0049E	89\$:	CLRL	-(SP)	1095
	7E		00G	8F	9A	004A0		MOVZBL	#PASSK_FILNAMREQ, -(SP)	
				18	11	004A4	90\$:	BRB	93\$	
	18			69	E8	004A6	91\$:	BLBS	(R9), 94\$	1102
14	68			04	E0	004A9		BBS	#4, (R8), 94\$	
	01			64	91	004AD		CMPB	(R4), #1	1103
				0F	12	004B0		BNEQ	94\$	
	65		08	AE	B1	004B2		CMPW	USR_USZ, (R5)	1104
				09	13	004B6		BEQL	94\$	
				7E	D4	004B8	92\$:	CLRL	-(SP)	1106
	7E		00G	8F	9A	004BA		MOVZBL	#PASSK_RECLEINIC, -(SP)	
				02E0	31	004BE	93\$:	BRW	147\$	
	50		08	A6	D0	004C1	94\$:	MOVL	8(PFV), PFD	1116
				60	D5	004C5		TSTL	(PFD)	1118
				12	13	004C7		BEQL	95\$	
				F4	AD	9F	004C9	PUSHAB	XABKEY_SIZE	1121
				F8	AD	9F	004CC	PUSHAB	XABKEY_ADDR	
				BC	AD	9F	004CF	PUSHAB	XAB_SUM	
			00	B040	9F	004D2		PUSHAB	@0(PFD)[PFD]	1122
	0000V	CF		04	FB	004D6		CALLS	#4, FILL_KEY_XABS	1121
		6A	0420	8F	A8	004DB	95\$:	BISW2	#1056, (R10)	1133
	05	A7		06	88	004E0		BISB2	#6, 5(RAB)	1136
				63	95	004E4		TSTB	(R3)	1142
				0A	12	004E6		BNEQ	96\$	
		01		64	91	004E8		CMPB	(R4), #1	1143
				05	13	004EB		BEQL	96\$	
			2C	AE	95	004ED		TSTB	KEYWORDS_SEEN	1144

			05	18	004F0	BGEQ	97\$		
	7A	A7	08	AE	B0 004F2	MOVW	USR_USZ, 122(RAB)		1146
14	2F	AE		02	E1 004F7	BBC	#2, KEYWORDS_SEEN+3, 98\$		1158
				56	DD 004FC	PUSHL	PFV		1167
				57	DD 004FE	PUSHL	RAB		
				58	DD 00500	PUSHL	R11		
		51	04	A2	DO 00502	MOVL	4(USER_ACTION_BPV), R1		
	00	B2		03	FB 00506	CALLS	#3, @07USER_ACTION_BPV)		
		52		50	DO 0050A	MOVL	R0, STATUS		
			01	31	0050D	BRW	118\$		1158
08		68		04	E1 00510	BBC	#4, (R8), 99\$		1181
	00000000G	00		58	DD 00514	PUSHL	R11		1183
				01	FB 00516	CALLS	#1, SYSSOPEN		
				09	11 0051D	BRB	100\$		
	00000000G	00		58	DD 0051F	PUSHL	R11		1185
		09		01	FB 00521	CALLS	#1, SYSSCREATE		
	0001825A	8F		50	E8 00528	BLBS	\$\$STATUS, 101\$		1186
				50	D1 0052B	CMPL	\$\$STATUS, #98906		
				DC	13 00532	BEQL	98\$		
		52		50	DO 00534	MOVL	\$\$STATUS, STATUS		1187
		45		52	E8 00537	BLBS	STATUS, 105\$		1195
	0001829A	8F		52	D1 0053A	CMPL	STATUS, #98970		1197
				36	12 00541	BNEQ	104\$		
04		68		04	E0 00543	BBS	#4, (R8), 102\$		1198
2E		6A		19	E1 00547	BBC	#25, (R10), 104\$		
		2A	5A	A7	E9 0054B	BLBC	90(RAB), 104\$		1199
	5A	A7		02	90 0054F	MOVB	#2, 90(RAB)		1207
	5B	A7		20	8A 00553	BICB2	#32, 91(RAB)		1208
	03	AA		01	88 00557	BISB2	#1, 3(R10)		1209
				58	DD 0055B	PUSHL	R11		1214
	00000000G	00		01	FB 0055D	CALLS	#1, SYSSOPEN		
	18	AE		50	DO 00564	MOVL	R0, \$\$STATUS		
			18	AE	9E 00568	MOVAB	\$\$STATUS, R0		1215
	0001825A	8F		50	D1 0056C	CMPL	R0, #98906		
				E6	13 00573	BEQL	103\$		
		52	18	AE	DO 00575	MOVL	\$\$STATUS, STATUS		1216
		03		52	E8 00579	BLBS	STATUS, 105\$		1225
				00C5	31 0057C	BRW	119\$		
		50	0084	C7	9E 0057F	MOVAB	132(RAB), R0		1235
04		60		0E	E1 00584	BBC	#14, (R0), 106\$		
		A8		01	88 00588	BISB2	#1, 3(R8)		1237
	03	03		69	E8 0058C	BLBS	(R9), 108\$		1245
				0094	31 0058F	BRW	116\$		
F9		60		02	E1 00592	BBC	#2, (R0), 107\$		1246
		A8		02	88 00596	BISB2	#2, 3(R8)		1249
F0	03	A7		01	E1 0059A	BBC	#1, 98(RAB), 107\$		1257
	62	A7		01	E1 0059A	BBC	#1, 98(RAB), 107\$		
		EC	5A	A7	E9 0059F	BLBC	90(RAB), 107\$		
7F		60		1B	E1 005A3	BBC	#27, (R0), 116\$		1258
7B		60		06	E0 005A7	BBS	#6, (R0), 116\$		1259
				5B	DD 005AB	PUSHL	R11		1262
	00000000G	00		01	FB 005AD	CALLS	#1, SYSSCLOSE		
		0D		50	E8 005B4	BLBS	\$\$STATUS, 110\$		
	0001825A	8F		50	D1 005B7	CMPL	\$\$STATUS, #98906		
				04	12 005BE	BNEQ	110\$		
		E7	03	A8	E8 005C0	BLBS	3(R8), 109\$		1263
	62	A7		04	90 005C4	MOVB	#4, 98(RAB)		1264
		64		03	90 005C8	MOVB	#3, (R4)		

	00010619	8F	4C	A7	D1	006B1	CMPL	76(RAB), #67097	
				03	13	006B9	BEQL	126\$	
		68		10	88	006BB	BISB2	#16, (R8)	1334
03		68		04	E0	006BE	BBS	#4, (R8), 127\$	1340
				00C7	31	006C2	BRW	145\$	
		0E	2E	AE	E9	006C5	BLBC	KEYWORDS SEEN+2, 128\$	1348
	14	AE		63	91	006C9	CMPB	(R3), USR_ORG	1349
				08	13	006CD	BEQL	128\$	
				7E	D4	006CF	CLRL	-(SP)	1351
		7E	00G	8F	9A	006D1	MOVZBL	#PASSK_ORGSPEINC, -(SP)	
				26	11	006D5	BRB	131\$	
		07		69	E9	006D7	BLBC	(R9), 129\$	1353
				63	95	006DA	TSTB	(R3)	
				03	13	006DC	BEQL	129\$	
				FD3C	31	006DE	BRW	79\$	
1A	2D	AE		03	E1	006E1	BBC	#3, KEYWORDS SEEN+1, 132\$	1361
		64	10	AE	91	006E6	CMPB	USR RFM, (R4)	1364
				14	13	006EA	BEQL	132\$	
		02	10	AE	91	006EC	CMPB	USR RFM, #2	1366
				05	12	006F0	BNEQ	130\$	
		03		64	91	006F2	CMPB	(R4), #3	1367
				09	13	006F5	BEQL	132\$	
				7E	D4	006F7	CLRL	-(SP)	1369
		7E	00G	8F	9A	006F9	MOVZBL	#PASSK_RECTYPINC, -(SP)	
				00A1	31	006FD	BRW	147\$	
11	0087	C7		04	E0	00700	BBS	#4, 135(RAB), 133\$	1380
0B	0084	C7		02	E0	00706	BBS	#2, 132(RAB), 133\$	1381
			7A	A7	B5	0070C	TSTW	122(RAB)	1383
				06	12	0070F	BNEQ	133\$	
	7A	A7	0080	C7	B0	00711	MOVW	128(RAB), 122(RAB)	1385
		52	7A	A7	3C	00717	MOVZWL	122(RAB), R2	1387
				5B	D4	0071B	CLRL	R11	
				52	D5	0071D	TSTL	R2	
				38	13	0071F	BEQL	141\$	
				5B	D6	00721	INCL	R11	
		01		64	91	00723	CMPB	(R4), #1	1389
				18	12	00726	BNEQ	137\$	
		07		69	E9	00728	BLBC	(R9), 134\$	1392
			2C	AE	95	0072B	TSTB	KEYWORDS_SEEN	1395
				29	18	0072E	BGEQ	141\$	
				05	11	00730	BRB	135\$	1396
		52		65	B1	00732	CMPW	(R5), R2	1400
				06	12	00735	BNEQ	136\$	
		52	0B	AE	B1	00737	CMPW	USR USZ, R2	1401
				1C	13	0073B	BEQL	141\$	
				FD78	31	0073D	BRW	92\$	1403
		04		69	E8	00740	BLBS	(R9), 138\$	1407
07		69		02	E1	00743	BBC	#2, (R9), 139\$	
			2C	AE	95	00747	TSTB	KEYWORDS_SEEN	1410
				0D	18	0074A	BGFQ	141\$	
				05	11	0074C	BEQL	140\$	1411
		52		65	B1	0074E	CMPW	(R5), R2	1415
				EA	1A	00751	BGTRU	136\$	
		52	0B	AE	B1	00753	CMPW	USR USZ, R2	1416
				E4	1A	00757	BGTRU	136\$	
		12		69	E9	00759	BLBC	(R9), 143\$	1425
		05		5B	E9	0075C	BLBC	R11, 142\$	1427

		65		52	B0	0075F		MOVW	R2, (R5)		1429	
				0A	11	00762		BRB	143\$			
		65	D2	AD	B1	00764	142\$:	CMPW	XAB FHC+10, (R5)		1432	
				04	1B	00768		BLEQU	143\$			
		65	D2	AD	B0	0076A		MOVW	XAB FHC+10, (R5)		1434	
		1B		68	E8	0076E	143\$:	BLBS	(R8), 145\$		1442	
	OE	68		01	E1	00771		BBC	#1, (R8), 144\$		1443	
		10		63	91	00775		CMPB	(R3), #16		1444	
				12	13	00778		BEQL	145\$			
				63	95	0077A		TSTB	(R3)		1445	
				05	12	0077C		BNEQ	144\$			
		01		64	91	0077E		CMPB	(R4), #1			
				09	13	00781		BEQL	145\$			
	14	68		02	E1	00783	144\$:	BBC	#2, (R8), 146\$		1446	
		20		63	91	00787		CMPB	(R3), #32		1447	
				0F	12	0078A		BNEQ	146\$			
	19	68		01	E1	0078C	145\$:	BBC	#1, (R8), 148\$		1458	
	05	0087		04	E1	00790		BBC	#4, 135(RAB), 146\$		1460	
				0E	C7	00796		BLBC	202(RAB), 148\$		1461	
			00CA	7E	D4	0079B	146\$:	CLRL	-(SP)		1463	
				7E	8F	0079D		MOVZBL	#PASSK ACCMETINC, -(SP)			
		00000000G	00G	02	FB	007A1	147\$:	CALLS	#2, PASS\$SIGNAL			
				04	04	007A8		RET				
				20	63	91	007A9	148\$:	CMPB	(R3), #32	1470	
				0E	12	007AC		BNEQ	149\$			
			F4	AD	9F	007AE		PUSHAB	XABKEY_SIZE		1473	
			F8	AD	9F	007B1		PUSHAB	XABKEY_ADDR			
			BC	AD	9F	007B4		PUSHAB	XAB_SUM			
		0000V	CF	03	FB	007B7		CALLS	#3, -CHECK_KEY_XABS			
		FFFFFFFF	8F	0C	AE	D1	007BC	149\$:	CMPL	FILE_TYPE, #-T	1483	
				1A	12	007C4		BNEQ	150\$			
		00000000'	13	00000000'	EF	E8	007C6		BLBS	PASS\$GV INPUT OPENED, 150\$		
	00	BE	01	00000000'	EF	01	90	007CD	MOVW	#1, PASS\$GV INPUT OPENED	1486	
			01	10	00	FO	0C7D4		INSV	#0, #16, #1, @0(SP)	1487	
				A8	0C	88	007DA		BISB2	#12, 1(R8)	1489	
					42	11	007DE		BRB	152\$	1483	
			01	0C	AE	D1	007E0	150\$:	CMPL	FILE_TYPE, #1	1491	
					2A	12	007E4		BNEQ	151\$		
		00000000'	23	00000000'	EF	E8	007E6		BLBS	PASS\$GV OUTPUT OPENED, 151\$		
			01	00000000'	EF	01	90	007ED	MOVW	#1, PASS\$GV_OUTPUT_OPENED	1494	
	00	BE	01	A8	30	88	007F4		BISB2	#48, 1(R8)	1496	
				12	01	FO	007F8		INSV	#1, #18, #1, @0(SP)	1497	
	00	BE	01	10	01	FO	007FE		INSV	#1, #16, #1, @0(SP)	1498	
	00	BE	01	11	00	FO	00804		INSV	#0, #17, #1, @0(SP)	1499	
			04	A7	02	88	0080A		BISB2	#2, 4(RAB)	1500	
					12	11	0080E		BRB	152\$	1491	
	00	BE	01	12	00	FO	00810	151\$:	INSV	#0, #18, #1, @0(SP)	1507	
	00	BE	01	10	01	FO	00816		INSV	#1, #16, #1, @0(SP)	1508	
	00	BE	01	11	00	FO	0081C		INSV	#0, #17, #1, @0(SP)	1509	
				52	24	A7	9E	00822	152\$:	MOVAB	36(RAB), R2	1519
				13	69	E9	00826		BLBC	(R9), 153\$	1516	
				7E	65	3C	00829		MOVZWL	(R5), -(SP)	1519	
		00000000G	00	01	FB	0082C		CALLS	#1, PASS\$GET_VM			
				62	50	D0	00833		MOVL	R0, (R2)		
			02	A8	08	88	00836		BISB2	#8, 2(R8)	1520	
					0D	11	0083A		BRB	155\$	1516	
	06			69	02	E1	0083C	153\$:	BBC	#2, (R9), 154\$	1522	

62		66	02	C1	00840	ADDL3	#2, (PFV), (R2)	1524	
			03	11	00844	BRB	155\$	1526	
		62	66	D0	00846	154\$:	MOVL (PFV), (R2)	1532	
	E8	A7	62	D0	00849	155\$:	MOVL (R2), -24(RAB)	1533	
	EC	A7	62	D0	0084D		MOVL (R2), -20(RAB)	1534	
	F4	A7	65	3C	00851		MOVZWL (R5), -12(RAB)	1535	
FO	A7	EC	A7	C1	00855		ADDL3 -12(RAB), -20(RAB), -16(RAB)	1546	
			08	9E	0085C		MOVAB 151(RAB), R2	1542	
				E1	00861		BBC #4, (R10), 156\$	1545	
				94	00865		CLRB 159(RAB)	1546	
				94	00869		CLRB (R2)	1542	
				1E	11	0086B	BRB 157\$	1557	
				9A	0086D	156\$:	MOVZBL (R2), -(SP)	1558	
	00000000G	7E	01	FB	00870		CALLS #1, PASS\$GET_VM		
	04	0C	50	D0	00877		MOVL R0, @4(SP)		
		BE	62	9A	0087B		MOVZBL (R2), R0		
		50	BE	D0	0087E		MOVL @4(SP), R6		
66	34	AE	50	28	00882		MOV C3 R0, RESULT_NAME_STRING, (R6)		
	02	AB	04	88	00887		BISB2 #4, 2(R8)	1559	
	E0	A7	01	CE	0088B	157\$:	MNEGL #1, -32(RAB)	1566	
		4C	69	E9	0088F		BLBC (R9), 158\$	1568	
	24	AE	8F	D0	00892		MOVL #17694733, LOGNAM_DSC	1583	
	28	AE	CF	9E	0089A		MOVAB P.AAG, LOGNAM_DSC+4	1584	
	1C	AE	8F	D0	008A0		MOVL #17694975, LIMIT_DSC	1587	
	20	AE	AE	9E	008A8		MOVAB RESULT_NAME_STRING, LIMIT_DSC+4	1588	
			7E	7C	008AD		CLRB -(SP)	1592	
			7E	D4	008AF		CLRL -(SP)		
			28	AE	9F	008B1	PUSHAB LIMIT_DSC		
			2C	AE	9F	008B4	PUSHAB LIMIT_DSC		
			38	AE	9F	008B7	PUSHAB LOGNAM_DSC		
	00000000G	00	06	FB	008BA		CALLS #6, SYS\$STRNLOG		
		01	50	D1	008C1		CML R0, #1	1593	
			18	12	008C4		BNEQ 158\$		
			01	DD	008C6		PUSHL #1	1602	
			04	DD	008C8		PUSHL #4		
			E0	A7	9F	008CA	PUSHAB -32(RAB)		
			28	AE	9F	008CD	PUSHAB LIMIT_DSC		
	00000000G	00	04	FB	008D0		CALLS #4, OTS\$CVT_TI_L		
		04	50	E8	008D7		BLBS R0, 158\$		
	E0	A7	01	CE	008DA		MNEGL #1, -32(RAB)	1606	
			68	A7	D4	008DE	158\$:	CLRL 104(RAB)	1614
			F8	AD	D5	008E1		TSTL XABKEY_ADDR	1620
			0D	13	008E4		BEQL 159\$		
			F8	AD	9F	008E6		PUSHAB XABKEY_ADDR	1622
			F4	AD	DD	008E9		PUSHL XABKEY_SIZE	
00	BE		02	FB	008EC		CALLS #2, PASS\$FREE_VM		
		01	01	F0	008F3	159\$:	INSV #1, #29, #1, @0(SP)	1628	
			57	DD	008F9		PUSHL RAB	1634	
	00000000G	00	01	FB	008FB		CALLS #1, PASS\$ADD_FILE		
			04	00902			RET	1638	
			0000	00903	160\$:		.WORD Save nothing	0452	
		50	08	AC	D0	00905		MOVL 8(AP), R0	
		50	04	A0	D0	00909		MOVL 4(R0), R0	
			F4	A0	9F	0090D		PUSHAB XABKEY_SIZE	
			F8	A0	9F	00910		PUSHAB XABKEY_ADDR	
			FC	A0	9F	00913		PUSHAB PFV_ADDR	
			03	DD	00916		PUSHL #3		

PASSOPEN2
1-015

OPEN procedure
PASS\$OPEN - Open a file

D 14
16-Sep-1984 01:46:15
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742
[PASRTL.SRC]PASCOPEN2.B32;1

Page 47
(10)

			5E	DD	00918	PUSHL	SP		
			AC	7D	0091A	MOVQ	4(AP), -(SP)		
0000V	7E	04	03	FB	0091E	CALLS	#3, OPEN_HANDLER		
	CF		04	00923	RET				

; Routine Size: 2340 bytes, Routine Base: _PASS\$CODE + 00E5

; 1585 1639 1
; 1586 1640 1 !<BLF/PAGE>

P
1

: 1588
: 1589
: 1590
: 1591
: 1592
: 1593
: 1594
: 1595
: 1596
: 1597
: 1598
: 1599
: 1600
: 1601
: 1602
: 1603
: 1604
: 1605
: 1606
: 1607
: 1608
: 1609
: 1610
: 1611
: 1612
: 1613
: 1614
: 1615
: 1616
: 1617
: 1618
: 1619
: 1620
: 1621
: 1622
: 1623
: 1624
: 1625
: 1626
: 1627
: 1628
: 1629
: 1630
: 1631
: 1632
: 1633
: 1634
: 1635
: 1636
: 1637
: 1638
: 1639
: 1640
: 1641
: 1642
: 1643
: 1644

1641 1
1642 1
1643 1
1644 1
1645 1
1646 1
1647 1
1648 1
1649 1
1650 1
1651 1
1652 1
1653 1
1654 1
1655 1
1656 1
1657 1
1658 1
1659 1
1660 1
1661 1
1662 1
1663 1
1664 1
1665 1
1666 1
1667 1
1668 1
1669 1
1670 1
1671 1
1672 1
1673 1
1674 1
1675 1
1676 1
1677 1
1678 1
1679 1
1680 1
1681 1
1682 1
1683 1
1684 1
1685 1
1686 1
1687 1
1688 1
1689 1
1690 1
1691 1
1692 1
1693 2
1694 2
1695 2
1696 2
1697 2

```

%SBTTL 'PASS$OPEN IMPLICIT - Open implicitly opened files'
GLOBAL ROUTINE PASS$OPEN_IMPLICIT (
    PFV: REF $PASS$PFV FILE VARIABLE,      ! File variable
    IN_FCB: REF $PASS$FCB CONTROL_BLOCK;    ! Control block
    FCB: REF $PASS$FCB CONTROL_BLOCK       ! Output FCB
): JSB_OPEN_IMPLICIT NOVALUE =

++
FUNCTIONAL DESCRIPTION:

    This procedure is called from procedures which do not
    accept textfiles, but have been called with a file that
    has its PFV$V_VALID bit clear. So that a proper error
    message can be given, this procedure opens the file INPUT
    or OUTPUT, if that is the current file and if that file
    has not already been opened.

CALLING SEQUENCE:

    JSB_OPEN_IMPLICIT PASS$OPEN_IMPLICIT (PFV.mr.r, IN_FCB.mr.r;
                                           FCB.mr.r)

FORMAL PARAMETERS:

    PFV          - The Pascal File Variable (PFV) for the file.
    IN_FCB       - The File Control Block (FCB) for the file.
    FCB          - The result FCB for the file.

IMPLICIT INPUTS:

    It is assumed that the caller has verified the PFV and has locked
    it.

IMPLICIT OUTPUTS:

    NONE

COMPLETION STATUS:

    NONE

SIDE EFFECTS:

    NONE

SIGNALLED ERRORS:

--
BEGIN
    FCB = IN_FCB [FCB$R_FCB];      ! Set output FCB
!+

```

```

: 1645      1698  2      ! See if the file is open.  If not, but is INPUT or OUTPUT, open it.
: 1646      1699  2      ! Otherwise, return.
: 1647      1700  2      !-
: 1648      1701  2
: 1649      1702  2      IF NOT .PFV [PFV$V_OPEN] ! Not open
: 1650      1703  2      THEN
: 1651      1704  3      BEGIN
: 1652      1705  3      IF (PFV [PFV$R_PFV] EQLA PASS$V_INPUT) AND NOT .PASS$GV_INPUT_OPENED
: 1653      1706  3      THEN
: 1654      1707  3      PASS$OPEN (PFV [PFV$R_PFV], PASS$K_HISTORY_READONLY; FCB)
: 1655      1708  3      ELSE IF (PFV [PFV$R_PFV] EQLA PASS$V_OUTPUT) AND NOT .PASS$GV_OUTPUT_OPENED
: 1656      1709  3      THEN
: 1657      1710  3      PASS$OPEN (PFV [PFV$R_PFV], PASS$K_HISTORY_NEW; FCB)
: 1658      1711  3      ELSE
: 1659      1712  4      BEGIN
: 1660      1713  4      !
: 1661      1714  4      ! If PFD or buffer address is self-relative, resolve it.
: 1662      1715  4      !-
: 1663      1716  4
: 1664      1717  4      IF .PFV [PFV$V_RELPFD]
: 1665      1718  4      THEN
: 1666      1719  5      BEGIN
: 1667      1720  5      PFV [PFV$A_PFD] = .PFV [PFV$A_PFD] + PFV [PFV$R_PFV];
: 1668      1721  5      PFV [PFV$V_RELPFD] = 0;
: 1669      1722  4      END;
: 1670      1723  4
: 1671      1724  4      IF .PFV [PFV$V_RELBUF]
: 1672      1725  4      THEN
: 1673      1726  5      BEGIN
: 1674      1727  5      PFV [PFV$A_BUFFER] = .PFV [PFV$A_BUFFER] + PFV [PFV$R_PFV];
: 1675      1728  5      PFV [PFV$V_RELBUF] = 0;
: 1676      1729  4      END;
: 1677      1730  4
: 1678      1731  3      END;
: 1679      1732  2      END;
: 1680      1733  2
: 1681      1734  2      RETURN;
: 1682      1735  2
: 1683      1736  1      END;

```

! End of routine PASS\$OPEN_IMPLICIT

	52	04	A6	9E	00000	PASS\$OPEN_IMPLICIT::		
						MOVAB	4(PFV), R2	: 1702
4B	62		1D	E0	00004	BBS	#29, (R2), 5\$: 1705
	50	00000000G	00	9E	00008	MOVAB	PASS\$V_INPUT, R0	
	50		56	D1	0000F	CMPL	PFV, R0	
			0B	12	00012	BNEQ	1\$	
	04	00000000'	EF	E8	00014	BLBS	PASS\$GV_INPUT_OPENED, 1\$	
			06	DD	0001B	PUSHL	#6	: 1707
			15	11	0001D	BRB	2\$	
	50	00000000G	00	9E	0001F	MOVAB	PASS\$V_OUTPUT, R0	: 1708
	50		56	D1	00026	CMPL	PFV, R0	
			11	12	00029	BNEQ	3\$	
	0A	00000000'	EF	E8	0002B	BLBS	PASS\$GV_OUTPUT_OPENED, 3\$: 1711

PASSOPEN2
1-015

OPEN procedure
PASS\$OPEN_IMPLICIT - Open implicitly opened

G 14
16-Sep-1984 01:46 15
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742
[PASRTL.SRC]PASOPEN2.B32;1

Page 50
(11)

P
1

			04	DD	00032		PUSHL	#4				
			56	DD	00034	2\$:	PUSHL	PFV			:	1710
	F6A1	CF	02	FB	00036		CALLS	#2,	PASS\$OPEN		:	
				05	00038		RSB				:	
08		62	1C	E1	0003C	3\$:	BBC	#28,	(R2), 4\$:	1717
	08	A6	56	C0	00040		ADDL2	PFV,	8(PFV)		:	1720
	03	A2	10	8A	00044		BICB2	#16,	3(R2)		:	1721
07		62	1B	E1	00048	4\$:	BBC	#27,	(R2), 5\$:	1724
		66	56	C0	0004C		ADDL2	PFV,	(PFV)		:	1727
	03	A2	08	8A	0004F		BICB2	#8,	3(R2)		:	1728
			05	00053	5\$:		RSB				:	1736

; Routine Size: 84 bytes, Routine Base: _PASS\$CODE + 0A09

```

: 1685 1737 1 %SBTTL 'FILL_KEY_XABS - Fill in key XABs for indexed file'
: 1686 1738 1 ROUTINE FILL_KEY_XABS (
: 1687 1739 1   PFV: REF $PASS$PFV_FILE VARIABLE,      . File variable
: 1688 1740 1   FCB: REF $PASS$FCB_CONTROL_BLOCK,   . File control block
: 1689 1741 1   KDB: REF $PASS$KDB_KEY_DESCRIPTOR, . Key descriptor
: 1690 1742 1   XAB_SUM: REF $XABSUM_DECL,       . Summary XAB
: 1691 1743 1   XABKEY_ADDR: REF VECTOR [, LONG], . Address of KEY XABs
: 1692 1744 1   XABKEY_SIZE: REF VECTOR [, LONG] . Length of KEY XABs
: 1693 1745 1 ) : CALC_FILL_KEY_XABS NOVALUE =
: 1694 1746 1
: 1695 1747 1 ++
: 1696 1748 1 FUNCTIONAL DESCRIPTION:
: 1697 1749 1
: 1698 1750 1     This procedure is called by PASS$OPEN to allocate and fill in
: 1699 1751 1     the KEY XABs for an indexed file.
: 1700 1752 1
: 1701 1753 1 CALLING SEQUENCE:
: 1702 1754 1
: 1703 1755 1     CALL_FILL_KEY_XABS FILL_KEY_XABS (PFV.r.r.r, FCB.mr.r,
: 1704 1756 1     KDB.r.r.r, XAB_SUM.wr.r, XABKEY_ADDR.wa.r, XABKEY_SIZE.wl.r)
: 1705 1757 1
: 1706 1758 1 FORMAL PARAMETERS:
: 1707 1759 1
: 1708 1760 1     PFV           - Pascal File Variable
: 1709 1761 1
: 1710 1762 1     FCB           - File control Block
: 1711 1763 1
: 1712 1764 1     KDB           - Key descriptor block. The structure is:
: 1713 1765 1
: 1714 1766 1     +-----+-----+
: 1715 1767 1     |           MBZ           | Count | <-- KDB
: 1716 1768 1     +-----+-----+
: 1717 1769 1     | MBZ | Size | Dtype | Key # |
: 1718 1770 1     +-----+-----+
: 1719 1771 1     |           Byte offset into record           |
: 1720 1772 1     +-----+-----+
: 1721 1773 1     | MBZ | Size | Dtype | Key # |
: 1722 1774 1     +-----+-----+
: 1723 1775 1     |           Byte offset into record           |
: 1724 1776 1     +-----+-----+
: 1725 1777 1     |           ...           |
: 1726 1778 1
: 1727 1779 1     XAB_SUM      - The summary XAB allocated by our caller. The
: 1728 1780 1     key XABs will be chained to it.
: 1729 1781 1
: 1730 1782 1     XABKEY_ADDR  - Place to store address of allocated KEY XABs
: 1731 1783 1     so that they may be deallocated in case of an
: 1732 1784 1     error.
: 1733 1785 1
: 1734 1786 1     XABKEY_SIZE  - Place to store length of allocated KEY XABs.
: 1735 1787 1
: 1736 1788 1 IMPLICIT INPUTS:
: 1737 1789 1
: 1738 1790 1     NONE
: 1739 1791 1
: 1740 1792 1 IMPLICIT OUTPUTS:
: 1741 1793 1

```

```

: 1742      1794  1  |      NONE
: 1743      1795  1  |
: 1744      1796  1  | COMPLETION STATUS:
: 1745      1797  1  |
: 1746      1798  1  |      NONE
: 1747      1799  1  |
: 1748      1800  1  | SIDE EFFECTS:
: 1749      1801  1  |
: 1750      1802  1  |      NONE
: 1751      1803  1  |
: 1752      1804  1  | SIGNALLED ERRORS:
: 1753      1805  1  |
: 1754      1806  1  |
: 1755      1807  1  | --
: 1756      1808  1  |
: 1757      1809  2  | BEGIN
: 1758      1810  2  |
: 1759      1811  2  | LOCAL
: 1760      1812  2  |     NEXT_KEY: REF $PASSKDB KEY_DESCRIPTOR,  ! Descriptor of next key
: 1761      1813  2  |     XAB_KEY: REF $XABKEY_DECL,             ! Key XAB
: 1762      1814  2  |     LAST_XAB: REF $XABKEY_DECL,           ! Previous key XAB
: 1763      1815  2  |     LAST_KEY: SIGNED;                     ! Last key number
: 1764      1816  2  |
: 1765      1817  2  | NEXT_KEY = .KDB;                          ! Get KDB address
: 1766      1818  2  |
: 1767      1819  2  | !+
: 1768      1820  2  | ! Allocate all the KEY XABs we need to describe the
: 1769      1821  2  | ! keys.
: 1770      1822  2  | !-
: 1771      1823  2  |
: 1772      1824  2  | FCB [FCB$N_KEYS] = .NEXT_KEY [KDB$B_COUNT];
: 1773      1825  2  | XAB_SUM [XAB$B_NOK] = .NEXT_KEY [KDB$B_COUNT]; ! RMS doesn't set on create
: 1774      1826  2  | XABKEY_SIZE [0] = .FCB [FCB$N_KEYS] * XAB$C_KEYLEN;
: 1775      1827  2  | IF .XABKEY_SIZE [0] NEQ 0
: 1776      1828  2  | THEN
: 1777      1829  3  |     BEGIN
: 1778      1830  3  |     XAB_KEY = PASS$GET_VM (PFV [PFV$R_PFV], .XABKEY_SIZE [0]);
: 1779      1831  3  |     XABKEY_ADDR [0] = .XAB_KEY;
: 1780      1832  2  |     END;
: 1781      1833  2  | LAST_XAB = XAB_SUM [0,0,0,0];
: 1782      1834  2  |
: 1783      1835  2  | NEXT_KEY = NEXT_KEY [4,0,0,0]; ! Advance pointer over count longword.
: 1784      1836  2  | LAST_KEY = -1; ! Initialize for comparison
: 1785      1837  2  |
: 1786      1838  2  | !+
: 1787      1839  2  | ! For each key, fill in the KEY XAB.
: 1788      1840  2  | !-
: 1789      1841  2  |
: 1790      1842  2  | INCR KEY_NUM FROM 0 TO .FCB [FCB$N_KEYS] - 1 DO
: 1791      1843  3  |     BEGIN
: 1792      1844  3  |
: 1793      1845  3  |     XAB_KEY [XAB$B_COD] = XAB$C_KEY;
: 1794      1846  3  |     XAB_KEY [XAB$B_BLN] = XAB$C_KEYLEN;
: 1795      1847  3  |     XAB_KEY [XAB$B_REF] = .NEXT_KEY [KDB$B_KEY_NUMBER];
: 1796      1848  3  |
: 1797      1849  3  |
: 1798      1850  3  | !+

```

```

1799 1851 3      ! Ensure that the keys are in ascending order.
1800 1852 3      !
1801 1853 3      IF .XAB_KEY [XAB$B_REF] LEQ .LAST_KEY
1802 1854 3      THEN
1803 1855 3          $PASSIO_ERROR (PASS_INVARGPAS,0);
1804 1856 3
1805 1857 3      !
1806 1858 3      ! If this is possibly a $CREATE, make sure that the keys
1807 1859 3      ! are in order and dense.
1808 1860 3      !
1809 1861 3      IF NOT .FCB [FCB$V_OLD_FILE]
1810 1862 3      THEN
1811 1863 3          IF .XAB_KEY [XAB$B_REF] NEQ .KEY_NUM
1812 1864 3          THEN
1813 1865 3              $PASSIO_ERROR (PASS_INVKEYDEF,0);
1814 1866 3
1815 1867 3      IF .XAB_KEY [XAB$B_REF] NEQ 0      ! Not primary key?
1816 1868 3      THEN
1817 1869 3          XAB_KEY [XAB$B_FLG] = XAB$M_CHG+XAB$M_DUP;      ! Allow changes and duplicates
1818 1870 3
1819 1871 3      IF .NEXT_KEY [KDB$B_DTYPE] EQL DSC$K_DTYPE_T      ! String?
1820 1872 3      THEN
1821 1873 4          BEGIN
1822 1874 4              XAB_KEY [XAB$B_DTP] = XAB$C_STG;
1823 1875 4              XAB_KEY [XAB$B_SIZ0] = .NEXT_KEY [KDB$B_SIZE];
1824 1876 4          END
1825 1877 3      ELSE IF .NEXT_KEY [KDB$B_DTYPE] EQL DSC$K_DTYPE_BU      ! Enumerated?
1826 1878 3      THEN
1827 1879 4          BEGIN
1828 1880 4              !
1829 1881 4              ! Since RMS doesn't have an unsigned byte type,
1830 1882 4              ! call it a 1-byte string, which is equivalent.
1831 1883 4              !
1832 1884 4              XAB_KEY [XAB$B_DTP] = XAB$C_STG;
1833 1885 4              XAB_KEY [XAB$B_SIZ0] = 1;
1834 1886 4          END
1835 1887 3      ELSE CASE .NEXT_KEY [KDB$B_DTYPE] FROM DSC$K_DTYPE_WU TO DSC$K_DTYPE_L OF
1836 1888 3          SET
1837 1889 3              [DSC$K_DTYPE_WU]:
1838 1890 4                  BEGIN
1839 1891 4                      XAB_KEY [XAB$B_DTP] = XAB$C_BN2;
1840 1892 4                      XAB_KEY [XAB$B_SIZ0] = 2;
1841 1893 4                  END;
1842 1894 3              [DSC$K_DTYPE_LU]:
1843 1895 4                  BEGIN
1844 1896 4                      XAB_KEY [XAB$B_DTP] = XAB$C_BN4;
1845 1897 4                      XAB_KEY [XAB$B_SIZ0] = 4;
1846 1898 4                  END;
1847 1899 3              [DSC$K_DTYPE_W]:
1848 1900 4                  BEGIN
1849 1901 4                      XAB_KEY [XAB$B_DTP] = XAB$C_IN2;
1850 1902 4                      XAB_KEY [XAB$B_SIZ0] = 2;
1851 1903 4                  END;
1852 1904 3              [DSC$K_DTYPE_L]:
1853 1905 4                  BEGIN
1854 1906 4                      XAB_KEY [XAB$B_DTP] = XAB$C_IN4;
1855 1907 4                      XAB_KEY [XAB$B_SIZ0] = 4;

```


			17	A2	95	00061	5\$:	TSTB	23(XAB_KEY)	1867
				04	13	00064		BEQL	6\$	
	12	A2		03	90	00066		MOVB	#3, 18(XAB_KEY)	1869
		0E	01	A4	91	0006A	6\$:	CMPB	1(NEXT_KEY), #14	1871
				0A	12	0006E		BNEQ	7\$	
	2E	A2	13	A2	94	00070		CLRB	19(XAB_KEY)	1874
			02	A4	90	00073		MOVB	2(NEXT_KEY), 46(XAB_KEY)	1875
				40	11	00078		BRB	16\$	1871
		02	01	A4	91	0007A	7\$:	CMPB	1(NEXT_KEY), #2	1877
				09	12	0007E		BNEQ	8\$	
	2E	A2	13	A2	94	00080		CLRB	19(XAB_KEY)	1884
				01	90	00083		MOVB	#1, 46(XAB_KEY)	1885
				31	11	00087		BRB	16\$	1877
FFBE		05	01	A4	8F	00089	8\$:	CASEB	1(NEXT_KEY), #3, #5	1887
	FFBE	03		000E		0008E	9\$:	.WORD	10\$-9\$,-	
		0014		001A		00096			11\$-9\$,-	
		0024							3\$-9\$,-	
									3\$-9\$,-	
									12\$-9\$,-	
									14\$-9\$	
				B0	11	0009A		BRB	3\$	1910
	13	A2		02	90	0009C	10\$:	MOVB	#2, 19(XAB_KEY)	1891
				0A	11	000A0		BRB	13\$	1892
	13	A2		04	90	000A2	11\$:	MOVB	#4, 19(XAB_KEY)	1896
				0E	11	000A6		BRB	15\$	1897
	13	A2		01	90	000A8	12\$:	MOVB	#1, 19(XAB_KEY)	1901
	2E	A2		02	90	000AC	13\$:	MOVB	#2, 46(XAB_KEY)	1902
				08	11	000B0		BRB	16\$	1887
	13	A2		03	90	000B2	14\$:	MOVB	#3, 19(XAB_KEY)	1906
	2E	A2		04	90	000B6	15\$:	MOVB	#4, 46(XAB_KEY)	1907
	16	A2	2E	A2	90	000BA	16\$:	MOVB	46(XAB_KEY), 22(XAB_KEY)	1917
	0000FFFF	8F	04	A4	D1	000BF		CMPB	4(NEXT_KEY), #65535	1919
				0E	1B	000C7		BLEQU	19\$	
				7E	D4	000C9	17\$:	CLRL	-(SP)	1921
		7E	00G	8F	9A	000CB		MOVZBL	#PASSK_INVKEYDEF, -(SP)	
	00000000G	00		02	FB	000CF	18\$:	CALLS	#2, PASS\$SIGNAL	
				04	04	000D6		RET		
	1E	A2	04	A4	B0	000D7	19\$:	MOVW	4(NEXT_KEY), 30(XAB_KEY)	1922
	04	A3		52	D0	000DC		MOVL	XAB_KEY, 4(LAST_XAB)	1924
		53		82	7E	000E0		MOVAQ	(XAB_KEY)+, LAST_XAB	1925
		58	0F	A2	9A	000E3		MOVZBL	15(XAB_KEY), LAST_KEY	1926
		52	44	A2	9E	000E7		MOVAB	68(R2), XAB_KEY	1927
		54		08	C0	000EB		ADDL2	#8, NEXT_KEY	1928
	01	55	D0	A7	F2	000EE	20\$:	AOBLS	-48(FCB), KEY_NUM, 21\$	1842
					04	000F3		RET		1933
				FF44	31	000F4	21\$:	BRW	2\$	1842

; Routine Size: 247 bytes, Routine Base: _PASSCODE + 0A5D

```

: 1883 1934 1 %SBTTL 'CHECK_KEY_XABS - Check key XABs for indexed file'
: 1884 1935 1 ROUTINE CHECK_KEY_XABS (
: 1885 1936 1     PFV: REF $PASSPFV_FILE VARIABLE,           ! Pascal File Variable
: 1886 1937 1     FCB: REF $PASSFCB_CONTROL_BLOCK,       ! File Control Block
: 1887 1938 1     XAB_SUM: REF $XABSUM_DECL,           ! Summary XAB
: 1888 1939 1     XABKEY_ADDR: REF VECTOR [, LONG],    ! Address of KEY XABs
: 1889 1940 1     XABKEY_SIZE: REF VECTOR [, LONG],    ! Size of KEY XABs
: 1890 1941 1     ): CALL_CHECK_KEY_XABS NOVALUE =
: 1891 1942 1
: 1892 1943 1
: 1893 1944 1     ++
: 1894 1945 1     FUNCTIONAL DESCRIPTION:
: 1895 1946 1         This procedure is called by PASS$OPEN to compare the keys that
: 1896 1947 1         are declared in the KDB (if any) against those actually defined
: 1897 1948 1         for the indexed file.
: 1898 1949 1
: 1899 1950 1     CALLING SEQUENCE:
: 1900 1951 1
: 1901 1952 1         CALL_CHECK_KEY_XABS (PFV.rr.r, FCB.mr.r, XAB_SUM.rr.r,
: 1902 1953 1             XABKEY_ADDR.ma.r, XABKEY_SIZE.ml.r)
: 1903 1954 1
: 1904 1955 1     FORMAL PARAMETERS:
: 1905 1956 1
: 1906 1957 1         PFV           - The Pascal File Variable (PFV) for the file.
: 1907 1958 1
: 1908 1959 1         FCB           - The result FCB for the file.
: 1909 1960 1
: 1910 1961 1         XAB_SUM        - The summary XAB. If new KEY XABs are allocated
: 1911 1962 1             by this procedure, they are linked to XAB_SUM.
: 1912 1963 1
: 1913 1964 1         XABKEY_ADDR     - The address of the previously allocated KEY XABs.
: 1914 1965 1
: 1915 1966 1         XABKEY_SIZE     - The size of the previously allocated KEY XABs.
: 1916 1967 1
: 1917 1968 1     IMPLICIT INPUTS:
: 1918 1969 1
: 1919 1970 1         NONE
: 1920 1971 1
: 1921 1972 1     IMPLICIT OUTPUTS:
: 1922 1973 1
: 1923 1974 1         NONE
: 1924 1975 1
: 1925 1976 1     COMPLETION STATUS:
: 1926 1977 1
: 1927 1978 1         NONE
: 1928 1979 1
: 1929 1980 1     SIDE EFFECTS:
: 1930 1981 1
: 1931 1982 1         NONE
: 1932 1983 1
: 1933 1984 1     SIGNALLED ERRORS:
: 1934 1985 1
: 1935 1986 1
: 1936 1987 1     --
: 1937 1988 1
: 1938 1989 2     BEGIN
: 1939 1990 2

```

```

1940 1991 2 LOCAL
1941 1992 2 PFD: REF $PASSPFD_FILE_DESCRIPTOR, ! File descriptor
1942 1993 2 KDB: REF $PASSKDB_KEY_DESCRIPTOR, ! Key descriptor block
1943 1994 2 XAB_KEY: REF $XABKEY_DECL, ! Pointer to KEY XAB
1944 1995 2 KDB_NKEYS, ! Number of keys in KDB
1945 1996 2 KEYPYPES: REF VECTOR [, WORD]; ! Vector of allowable key datatypes and lengths
1946 1997
1947 1998
1948 1999 BIND
1949 2000 FAB = FCB: REF $PASSFAB_FCB_STRUCT;
1950 2001 FCB [FCB$V_INDEXED] = 1; ! Indicate indexed organization
1951 2002
1952 2003 !+
1953 2004 ! Get number of keys defined for file.
1954 2005 !-
1955 2006
1956 2007 FCB [FCB$L_NKEYS] = .XAB_SUM [XAB$B_NOK];
1957 2008
1958 2009 !+
1959 2010 ! Get number of user-defined keys from KDB (if any).
1960 2011 !-
1961 2012
1962 2013 PFD = .PFV [PFV$A_PFD];
1963 2014 IF .PFD [PFD$A_KDB] NEQA 0 ! Is there a KDB?
1964 2015 THEN
1965 2016 BEGIN
1966 2017 KDB = .PFD [PFD$A_KDB] + PFD [PFD$R_PFD]; ! Get address of KDB
1967 2018 KDB_NKEYS = .KDB [KDB$B_COUNT]; ! Get count
1968 2019 KDB = KDB [4,0,0,0]; . Advance to first key description
1969 2020 END
1970 2021 ELSE
1971 2022 KDB_NKEYS = 0;
1972 2023
1973 2024 !+
1974 2025 ! If the number of keys specified in the KDB doesn't match the
1975 2026 ! actual number in the file, we have to throw away the KEY XABs
1976 2027 ! we allocated, allocate the correct number, and do a $DISPLAY
1977 2028 ! to fill them in.
1978 2029 !-
1979 2030
1980 2031 IF .KDB_NKEYS NEQ .FCB [FCB$L_NKEYS]
1981 2032 THEN
1982 2033 BEGIN
1983 2034 LOCAL
1984 2035 LAST_XAB: REF BLOCK [, BYTE]; ! Previous XAB
1985 2036
1986 2037 !+
1987 2038 ! Deallocate the old KEY XABs.
1988 2039 !-
1989 2040
1990 2041 IF .XABKEY_ADDR [0] NEQA 0
1991 2042 THEN
1992 2043 PASS$FREE_VM (.XABKEY_SIZE [0], XABKEY_ADDR [0]);
1993 2044
1994 2045 !+
1995 2046 ! Allocate enough new KEY XABs.
1996 2047 !-

```

```

: 1997 2048 3
: 1998 2049 3
: 1999 2050 3
: 2000 2051 3
: 2001 2052 3
: 2002 2053 3
: 2003 2054 3
: 2004 2055 3
: 2005 2056 3
: 2006 2057 3
: 2007 2058 3
: 2008 2059 3
: 2009 2060 4
: 2010 2061 4
: 2011 2062 4
: 2012 2063 4
: 2013 2064 4
: 2014 2065 4
: 2015 2066 4
: 2016 2067 3
: 2017 2068 3
: 2018 2069 3
: 2019 2070 3
: 2020 2071 3
: 2021 2072 3
: 2022 2073 3
: 2023 2074 3
: 2024 2075 3
: 2025 2076 3
: 2026 2077 3
: 2027 2078 3
: 2028 2079 4
: 2029 2080 3
: 2030 2081 3
: 2031 2082 2
: 2032 2083 2
: 2033 2084 2
: 2034 2085 2
: 2035 2086 2
: 2036 2087 2
: 2037 2088 2
: 2038 2089 2
: 2039 2090 2
: 2040 2091 2
: 2041 2092 2
: 2042 2093 2
: 2043 2094 2
: 2044 2095 2
: 2045 2096 2
: 2046 2097 2
: 2047 2098 2
: 2048 2099 2
: 2049 2100 2
: 2050 2101 2
: 2051 2102 2
: 2052 2103 2
: 2053 2104 2

```

```

XABKEY_SIZE [0] = XAB$C_KEYLEN * .FCB [FCB$L_NKEYS];
XAB_KEY = PASS$GET_VM (PFV [PFV$R_PFV], .XABKEY_SIZE [0]);
XABKEY_ADDR [0] = .XAB_KEY;

!+
! Fill in each KEY XAB and link it into the chain.
!-

LAST_XAB = .XAB_SUM;      ! Link KEY XABs after SUM XAB

INCR KEY_NUM FROM 0 TO .FCB [FCB$L_NKEYS] - 1 DO
  BEGIN
    XAB_KEY [XAB$B_COD] = XAB$C_KEY;
    XAB_KEY [XAB$B_BLN] = XAB$C_KEYLEN;
    XAB_KEY [XAB$B_REF] = .KEY_NUM;
    LAST_XAB [XAB$C_NXT] = .XAB_KEY;      ! Link in XAB
    LAST_XAB = .XAB_KEY;
    XAB_KEY = .XAB_KEY + XAB$C_KEYLEN;
  END;

!+
! Make sure that XAB_SUM is linked to the FAB.  A USER_ACTION
! routine may have unlinked it.
!-

FAB [FAB$L_XAB] = .XAB_SUM;

! Ask RMS to fill in our new XABs.
!-

IF NOT $PASSRMS_OP ($DISPLAY (FAB = FAB [0,0,0,0]))
  THEN
    $PASSIO_ERROR (PASS_ERRDUROPE);
  END;

!+
! Allocate a vector of longwords , one for each key, which will
! contain a bit mask that indicates what key expression datatypes
! may be used to reference that key, and the key size.  The mask
! occupies the first word and the length, the second word.
!-

KEYTYPES = PASS$GET_VM (PFV [PFV$R_PFV], .FCB [FCB$L_NKEYS] * 4);
FCB [FCB$A_KEY_TYPES] = .KEYTYPES;

!+
! For each key in the file, if the KDB describes the key, make sure
! it matches.  Set the KEYTYPES mask appropriately.  This mask has
! 16 bits, corresponding to standard datatype codes 0-15.  (Codes
! greater than 15 are not currently supported by Pascal for keys.)
! If a bit for a particular datatype is set, then a key expression
! of that type may be used to reference this key.  Some combinations
! require a range check of the key expression value.  This is done
! by PASS$FINDK.  Then set the key size in KEYTYPES.
!-

! The following table shows the allowable combinations of key value

```

```

: 2054 2105 2
: 2055 2106 2
: 2056 2107 2
: 2057 2108 2
: 2058 2109 2
: 2059 2110 2
: 2060 2111 2
: 2061 2112 2
: 2062 2113 2
: 2063 2114 2
: 2064 2115 2
: 2065 2116 2
: 2066 2117 2
: 2067 2118 2
: 2068 2119 2
: 2069 2120 2
: 2070 2121 2
: 2071 2122 2
: 2072 2123 2
: 2073 2124 2
: 2074 2125 2
: 2075 2126 2
: 2076 2127 2
: 2077 2128 2
: 2078 2129 2
: 2079 2130 2
: 2080 2131 2
: 2081 2132 2
: 2082 2133 2
: 2083 2134 2
: 2084 2135 2
: 2085 2136 2
: 2086 2137 2
: 2087 2138 2
: 2088 2139 2
: 2089 2140 2
: 2090 2141 2
: 2091 2142 2
: 2092 2143 2
: 2093 2144 2
: 2094 2145 2
: 2095 2146 2
: 2096 2147 2
: 2097 2148 2
: 2098 2149 2
: 2099 2150 2
: 2100 2151 2
: 2101 2152 2
: 2102 2153 2
: 2103 2154 2
: 2104 2155 2
: 2105 2156 2
: 2106 2157 2
: 2107 2158 2
: 2108 2159 2
: 2109 2160 2
: 2110 2161 2

```

```

: types and file key types. (ST1 refers to a 1-byte STG key)
:
:   STG  ST1  BN2  BN4  IN2  IN4  other
: BU    no   (1)  ok   ok   no   no   no
: WU    no   (1)  ok   ok   no   no   no
: LU    no   (1)  (2)  ok   no   no   no
: L     no   (1)  (2)  ck   (2)  ok   no
: T,Z   ok   (3)  no   no   no   no   (4)
:
: Notes:
: 1. Only if key not declared as string. Value must
:    be in range of key type.
: 2. Value must be in range of key type.
: 3. Only if key not declared as BU.
: 4. Can't be declared in key descriptor. Otherwise,
:    treated as string.
:
: XAB_KEY = .XABKEY_ADDR [0]; ! Get first KEY XAB
: INCR KEY_NUM FROM 0 TO .FCB [FCB$L_NKEYS] - 1 DO
: BEGIN
:   LOCAL
:     KEY_DTYPE: BYTE, ! Datatype of key in DSC$ codes
:     USR_KEY:      ! 1 if user defined key, otherwise 0
:
:   !+
:   ! Define literal masks for the datatypes.
:   !-
:   LITERAL
:     M_Z = 1^DSC$K_DTYPE_Z,
:     M_BU = 1^DSC$K_DTYPE_BU,
:     M_WU = 1^DSC$K_DTYPE_WU,
:     M_LU = 1^DSC$K_DTYPE_LU,
:     M_L = 1^DSC$K_DTYPE_L,
:     M_T = 1^DSC$K_DTYPE_T;
:
:   !+
:   ! Set USR_KEY if this key is in the KDB.
:   !-
:   USR_KEY = 0;
:   IF .KDB_NKEYS NEQ 0
:   THEN
:     IF .XAB_KEY [XAB$B_REF] EQL .KDB [KDB$B_KEY_NUMBER]
:     THEN
:       USR_KEY = 1;
:
:   !+
:   ! Find out what datatype the file's key is.
:   !-
:   CASE .XAB_KEY [XAB$B_DTP] FROM 0 TO XAB$C_MAXDTP OF
:   SET
:     [XAB$C_STG]:      ! String

```

```

2111      2162  4      BEGIN
2112      2163  4      IF .USR_KEY
2113      2164  4      THEN
2114      2165  4          IF .KDB [KDB$B_DTYPE] EQL DSC$K_DTYPE_BU
2115      2166  4          THEN
2116      2167  5              BEGIN
2117      2168  5                  IF .XAB_KEY [XAB$B_TKS] NEQ 1
2118      2169  5                  THEN
2119      2170  5                      $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM);
2120      2171  5                      KEYTYPES [0] = M_BU+M_WO+M_LU+M_L;
2121      2172  5                      END
2122      2173  4                  ELSE IF .KDB [KDB$B_DTYPE] EQL DSC$K_DTYPE_T
2123      2174  4                  THEN
2124      2175  5                      BEGIN
2125      2176  5                          IF .KDB [KDB$B_SIZE] NEQ .XAB_KEY [XAB$B_TKS]
2126      2177  5                          THEN
2127      2178  5                              $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM);
2128      2179  5                              KEYTYPES [0] = M_T+M_Z;
2129      2180  5                              END
2130      2181  4                      ELSE
2131      2182  5                          $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM)
2132      2183  4                  ELSE
2133      2184  4                      IF .XAB_KEY [XAB$B_TKS] EQL 1
2134      2185  4                      THEN
2135      2186  4                          KEYTYPES [0] = M_BU+M_L+M_T+M_Z
2136      2187  4                      ELSE
2137      2188  4                          KEYTYPES [0] = M_T+M_Z;
2138      2189  3      END;
2139      2190  3
2140      2191  3      [XAB$C_IN2]:      ! Word integer
2141      2192  4      BEGIN
2142      2193  4      IF .USR_KEY
2143      2194  4      THEN
2144      2195  4          IF .KDB [KDB$B_DTYPE] NEQ DSC$K_DTYPE_W
2145      2196  4          THEN
2146      2197  4              $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM);
2147      2198  4              KEYTYPES [0] = M_L;
2148      2199  3      END;
2149      2200  3
2150      2201  3      [XAB$C_IN4]:      ! Longword integer
2151      2202  4      BEGIN
2152      2203  4      IF .USR_KEY
2153      2204  4      THEN
2154      2205  4          IF .KDB [KDB$B_DTYPE] NEQ DSC$K_DTYPE_L
2155      2206  4          THEN
2156      2207  4              $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM);
2157      2208  4              KEYTYPES [0] = M_L;
2158      2209  3      END;
2159      2210  3
2160      2211  3      [XAB$C_BN2]:      ! Word unsigned
2161      2212  4      BEGIN
2162      2213  4      IF .USR_KEY
2163      2214  4      THEN
2164      2215  4          IF .KDB [KDB$B_DTYPE] NEQ DSC$K_DTYPE_WU
2165      2216  4          THEN
2166      2217  4              $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM);
2167      2218  4              KEYTYPES [0] = M_BU+M_WU+M_CU+M_L;

```

```

2168 2219 3      END;
2169 2220 3
2170 2221 3      [XAB$C BN4]:      ! Longword unsigned
2171 2222 4      BEGIN
2172 2223 4      IF .USR_KEY
2173 2224 4      THEN
2174 2225 4          IF .KDB [KDB$B_DTYPE] NEQ DSC$K_DTYPE_LU
2175 2226 4          THEN
2176 2227 4              $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM);
2177 2228 4          KEYTYPES [0] = M_BU+M_WU+M_U+M_L;
2178 2229 3      END;
2179 2230 3
2180 2231 3      [INRANGE]:      ! Defined by RMS, but not by PASCAL
2181 2232 4      BEGIN
2182 2233 4          +
2183 2234 4          Pascal does not support this key type.
2184 2235 4          However, if the file has one, it is treated as
2185 2236 4          a string.
2186 2237 4          -
2187 2238 4      IF .USR_KEY
2188 2239 4      THEN
2189 2240 4          $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM);
2190 2241 4      KEYTYPES [0] = M_T+M_Z;
2191 2242 3      END;
2192 2243 3
2193 2244 3      [OUTRANGE]:
2194 2245 3          $PASSBUGCHECK (BUG_BADKEYDTP);
2195 2246 3
2196 2247 3      TES;
2197 2248 3
2198 2249 3      +
2199 2250 3      Check offset of key.
2200 2251 3      -
2201 2252 3
2202 2253 3      IF .USR_KEY
2203 2254 3      THEN
2204 2255 3          IF .KDB [KDB$L_OFFSET] NEQ .XAB_KEY [XAB$W_POS0]
2205 2256 3          THEN
2206 2257 3              $PASSIO_ERROR (PASS_KEYDEFINC,1,.KEY_NUM);
2207 2258 3
2208 2259 3      +
2209 2260 3      Set size of key.
2210 2261 3      -
2211 2262 3
2212 2263 3      KEYTYPES [1] = .XAB_KEY [XAB$B_TKS];
2213 2264 3
2214 2265 3      +
2215 2266 3      Advance KDB pointer, if any remain.
2216 2267 3      -
2217 2268 3
2218 2269 3      IF .KDB_NKEYS GTR 0
2219 2270 3      THEN
2220 2271 4          BEGIN
2221 2272 4          KDB = KDB [8,0,0,0];
2222 2273 4          KDB_NKEYS = .KDB_NKEYS - 1;
2223 2274 3          END;
2224 2275 3

```

```

: 2225      2276      3
: 2226      2277      3
: 2227      2278      3
: 2228      2279      3
: 2229      2280      3
: 2230      2281      3
: 2231      2282      3
: 2232      2283      2
: 2233      2284      2
: 2234      2285      2
: 2235      2286      2
: 2236      2287      2
: 2237      2288      2
: 2238      2289      2
: 2239      2290      2
: 2240      2291      2
: 2241      2292      2
: 2242      2293      2
: 2243      2294      2
: 2244      2295      1

```

```

      +
      | Advance XAB_KEY and KEYTYPES.
      |
      |
      | XAB_KEY = .XAB_KEY [XAB$NXT];
      | KEYTYPES = KEYTYPES [2];
      |
      | END;      ! End of INCR loop
      |
      | +
      | | If any KDB keys remain, give an error.
      | |
      | |
      | | IF .KDB_NKEYS GTR 0
      | | THEN
      | |   $PASSIO_ERROR (PASS_KEYDEFINC,1,.KDB [KDB$B_KEY_NUMBER]);
      | |
      | | RETURN;
      | |
      | | END;      ! End of routine CHECK_KEY_XABS

```

```

      .EXTRN SYSSDISPLAY, PASSK_KEYDEFINC
      .EXTRN PASS$BUGCHECK

```

OF3C 0000 CHECK_KEY_XABS:

					WORD	Save R2,R3,R4,R5,R8,R9,R10,R11	1935
		5B 00000000G	00	9E 00002	MOVAB	PASS\$SIGNAL, R11	
		5A 00000000G	00	9E 00009	MOVAB	PASS\$GET_VM, R10	
	FE	A7 40	8F	88 00010	BISB2	#64, -2(FCB)	2001
		54 04	AC	D0 00015	MOVL	XAB_SUM, R4	2007
	DO	A7 09	A4	9A 00019	MOVZBL	9(R4), -48(FCB)	
		50 08	A6	D0 0001E	MOVL	8(PFD), PFD	2013
			60	D5 00022	TSTL	(PFD)	2014
			0C	13 00024	BEQL	1\$	
	58	60	50	C1 00026	ADDL3	PFD, (PFD), KDB	2017
		59	88	9A 0002A	MOVZBL	(KDB)+, KDB_NKEYS	2018
		58	03	C0 0002D	ADDL2	#3, KDB	2019
			02	11 00030	BRB	2\$	2014
			59	D4 00032	CLRL	KDB_NKEYS	2022
	DO	A7	59	D1 00034	CMPL	KDB_NKEYS, -48(FCB)	2031
			77	13 00038	BEQL	8\$	
		53 08	AC	D0 0003A	MOVL	XABKEY_ADDR, R3	2041
			63	D5 0003E	TSTL	(R3)	
			0C	13 00040	BEQL	3\$	
			53	DD 00042	PUSHL	R3	2043
			0C	BC DD 00044	PUSHL	@XABKEY_SIZE	
	OC BC	00000000G	00	02 FB 00047	CALLS	#2, PASS\$FREE_VM	
		DO	A7	0000004C	MULL3	#76, -48(FCB), @XABKEY_SIZE	2049
			0C	BC DD 00058	PUSHL	@XABKEY_SIZE	2050
		6A	01	FB 0005B	CALLS	#1, PASS\$GET_VM	
		52	50	D0 0005E	MOVL	R0, XAB_KEY	
		63	52	D0 00061	MOVL	XAB_KEY, (R3)	2051
		50	54	D0 00064	MOVL	R4, LAST_XAB	2057
		51	01	CE 00067	MNEGL	#1, KEY_NUM	2059
			14	11 0006A	BRB	5\$	

		62	4C15	8F	B0	0006C	4\$:	MOVW	#19477, (XAB_KEY)	2061
	17	A2		51	90	00071		MOVW	KEY_NUM, 23(XAB_KEY)	2063
	04	A0		52	D0	00075		MOVL	XAB_KEY, 4(LAST_XAB)	2064
		50		82	7E	00079		MOVAB	(XAB_KEY)+, LAST_XAB	2065
		52	44	A2	9E	0007C		MOVAB	68(R2), XAB_KEY	2066
E7		51	D0	A7	F2	00080	5\$:	AOBLSS	-48(FCB), KEY_NUM, 4\$	2059
	68	A7		54	D0	00085		MOVL	R4, 104(FCB)	2074
		53	44	A7	9E	00089		MOVAB	68(R7), R3	2079
				53	DD	0008D	6\$:	PUSHL	R3	
	00000000G	00		01	FB	0008F		CALLS	#1, SYSS\$DISPLAY	
		18		50	EB	00096		BLBS	\$\$STATUS, 8\$	
	0001825A	8F		50	D1	00099		CMPL	\$\$STATUS, #98906	
				04	12	000A0		BNEQ	7\$	
		E7	FF	A7	EB	000A2		BLBS	-1(FCB), 6\$	
		08		50	EB	000A6	7\$:	BLBS	\$\$STATUS, 8\$	
		7E	00G	8F	9A	000A9		MOVZBL	#PASSK_ERRDUROPE, -(SP)	2081
		6B		01	FB	000AD		CALLS	#1, PASS\$\$SIGNAL	
					04	000B0		RET		
7E	D0	A7		02	78	000B1	8\$:	ASHL	#2, -48(FCB), -(SP)	2091
		6A		01	FB	000B6		CALLS	#1, PASS\$\$GET_VM	
		53		50	D0	000B9		MOVL	R0, KEYTYPES	
		CC		53	D0	000BC		MOVL	KEYTYPES, -52(FCB)	2092
		52	08	BC	D0	000C0		MOVL	@XABKEY_ADDR, XAB_KEY	2123
		55		01	CE	000C4		MNEGL	#1, KEY_NUM	2125
					31	000C7		BRW	29\$	
					54	D4	9\$:	CLRL	USR_KEY	2147
					59	D5		TSTL	KDB_NKEYS	2148
					09	13		BEQL	10\$	
		68	17	A2	91	000D0		CMPB	23(XAB_KEY), (KDB)	2150
				03	12	000D4		BNEQ	10\$	
		54		01	D0	000D6		MOVL	#1, USR_KEY	2152
	07	00	13	A2	8F	000D9	10\$:	CASEB	19(XAB_KEY), #0, #7	2158
0052	0062	0047		001A		000DE	11\$:	.WORD	12\$-11\$,-	
007B	007B	007B		006B		000E6			16\$-11\$,-	
									20\$-11\$,-	
									18\$-11\$,-	
									21\$-11\$,-	
									23\$-11\$,-	
									23\$-11\$,-	
									23\$-11\$	
				05	DD	000EE		PUSHL	#5	2245
	00000000G	00		01	FB	000F0		CALLS	#1, PASS\$\$BUGCHECK	
					04	000F7		RET		
		1D		54	E9	000F8	12\$:	BLBC	USR_KEY, 15\$	2163
		02	01	A8	91	000FB		CMPB	1(KDB), #2	2165
				08	12	000FF		BNEQ	14\$	
		01	16	A2	91	00101		CMPB	22(XAB_KEY), #1	2168
				4B	13	00105	13\$:	BEQL	22\$	
				64	11	00107		BRB	26\$	2170
		0E	01	A8	91	00109	14\$:	CMPB	1(KDB), #14	2173
				5E	12	0010D		BNEQ	26\$	
	16	A2	02	A8	91	0010F		CMPB	2(KDB), 22(XAB_KEY)	2176
				46	13	00114		BEQL	24\$	
				55	11	00116		BRB	26\$	2178
		01	16	A2	91	00118	15\$:	CMPB	22(XAB_KEY), #1	2184
				3E	12	0011C		BNEQ	24\$	
		63	4105	8F	B0	0011E		MOVW	#16645, (KEYTYPES)	2186

				3C 11 00123	BRB	25\$			
				54 E9 00125 16\$:	BLBC	USR_KEY, 19\$			2193
			07	A8 91 00128	CMPB	1(KDB), #7			2195
				0B 13 0012C 17\$:	BEQL	19\$			
				3D 11 0012E	BRB	26\$			2197
			06	54 E9 00130 18\$:	BLBC	USR_KEY, 19\$			2203
			08	A8 91 00133	CMPB	1(KDB), #8			2205
				F3 11 00137	BRB	17\$			
			63	8F B0 00139 19\$:	MOVW	#256, (KEYTYPES)			2208
				21 11 0013E	BRB	25\$			2158
			0F	54 E9 00140 20\$:	BLBC	USR_KEY, 22\$			2213
			03	A8 91 00143	CMPB	1(KDB), #3			2215
				BC 11 00147	BRB	13\$			
			06	54 E9 00149 21\$:	BLBC	USR_KEY, 22\$			2223
			04	A8 91 0014C	CMPB	1(KDB), #4			2225
				B3 11 00150	BRB	13\$			
			63	8F B0 00152 22\$:	MOVW	#284, (KEYTYPES)			2228
				08 11 00157	BRB	25\$			2158
			11	54 E8 00159 23\$:	BLBS	USR_KEY, 26\$			2238
			63	8F B0 0015C 24\$:	MOVW	#16385, (KEYTYPES)			2241
			0D	54 E9 00161 25\$:	BLBC	USR_KEY, 27\$			2253
04	AB	1E	A2	10	00 ED 00164	CMPZV	#0, #16, 30(XAB_KEY), 4(KDB)		2255
					04 13 0016B	BEQL	27\$		
					55 DD 0016D 26\$:	PUSHL	KEY_NUM		2257
					26 11 0016F	BRB	32\$		
			02	A3	16	A2 9B 00171 27\$:	MOVZBW	22(XAB_KEY), 2(KEYTYPES)	2263
						59 D5 00176	TSTL	KDB_NKEYS	2269
						05 15 00178	BLEQ	28\$	
			58			08 C0 0017A	ADDL2	#8, KDB	2272
						59 D7 0017D	DECL	KDB_NKEYS	2273
			52	04	A2 D0 0017F 28\$:	MOVL	4(XAB_KEY), XAB_KEY		2280
			53			04 C0 00183	ADDL2	#4, KEYTYPES	2281
		02				55 D0 A7 F2 00186 29\$:	AOBLSS	-48(FCB), KEY_NUM, 30\$	2125
						03 11 0018B	BRB	31\$	
						FF3A 31 0018D 30\$:	BRW	9\$	
						59 D5 00190 31\$:	TSTL	KDB_NKEYS	2289
						0C 15 00192	BLEQ	33\$	
			7E			68 9A 00194	MOVZBL	(KDB), -(SP)	2291
						01 DD 00197 32\$:	PUSH!	#1	
			7E	00G		8F 9A 00199	MOVZBL	#PASSK_KEYDEFINC, -(SP)	
			6B			03 FB 0019D	CALLS	#3, PASS\$SIGNAL	
						04 001A0 33\$:	RET		2295

; Routine Size: 417 bytes, Routine Base: _PASSCODE + 0B54

```

: 2246 2296 1 %SBTTL 'OPEN_HANDLER - Close file on unwind'
: 2247 2297 1 ROUTINE OPEN_HANDLER (
: 2248 2298 1     SIGNAL_ARGS: REF BLOCK [, BYTE],           ; Signal arguments
: 2249 2299 1     MECHANISM_ARGS: REF BLOCK [, BYTE],      ; Mechanism arguments
: 2250 2300 1     ENABLE_ARGS: REF VECTOR [, LONG]   ; Enable arguments
: 2251 2301 1     ) =
: 2252 2302 1
: 2253 2303 1
: 2254 2304 1 **
: 2255 2305 1 FUNCTIONAL DESCRIPTION:
: 2256 2306 1     OPEN_HANDLER is a condition handler established by PASS$OPEN. It
: 2257 2307 1     serves only to close the file which is currently being opened
: 2258 2308 1     if an unwind occurs.
: 2259 2309 1
: 2260 2310 1 CALLING SEQUENCE:
: 2261 2311 1
: 2262 2312 1     ret_status.wlc.v = OPEN_HANDLER (signal_args.rlu.r, mechanism_args.rlu.r,
: 2263 2313 1     enable_args.rlu.r)
: 2264 2314 1
: 2265 2315 1 FORMAL PARAMETERS:
: 2266 2316 1
: 2267 2317 1     SIGNAL_ARGS           - The signal arguments list
: 2268 2318 1     MECHANISM_ARGS        - The mechanism argument list
: 2269 2319 1     ENABLE_ARGS          - A vector of four longwords, the first of
: 2270 2320 1                       which contains the count of remaining
: 2271 2321 1                       longwords (3). The remaining longwords
: 2272 2322 1                       contain the following information:
: 2273 2323 1                       ENABLE_ARGS [1] - Address of longword which
: 2274 2324 1                       contains the address of the
: 2275 2325 1                       active PFV.
: 2276 2326 1                       ENABLE_ARGS [2] - Address of longword which
: 2277 2327 1                       contains the address of
: 2278 2328 1                       a block of KEY XABs to be
: 2279 2329 1                       deallocated (if non-zero).
: 2280 2330 1                       ENABLE_ARGS [3] - Address of longword which
: 2281 2331 1                       contains the number of bytes
: 2282 2332 1                       of KEY XABs to deallocate.
: 2283 2333 1
: 2284 2334 1 IMPLICIT INPUTS:
: 2285 2335 1
: 2286 2336 1     NONE
: 2287 2337 1
: 2288 2338 1 IMPLICIT OUTPUTS:
: 2289 2339 1
: 2290 2340 1     NONE
: 2291 2341 1
: 2292 2342 1 COMPLETION STATUS:
: 2293 2343 1
: 2294 2344 1     $$$_RESIGNAL
: 2295 2345 1
: 2296 2346 1 SIDE EFFECTS:
: 2297 2347 1
: 2298 2348 1     Closes the current file upon an unwind.
: 2299 2349 1     Deallocates KEY XABs.
: 2300 2350 1
: 2301 2351 1 SIGNALLED ERRORS:
: 2302 2352 1

```

```

2303 2353 1 | NONE
2304 2354 1 | |
2305 2355 1 | |
2306 2356 1 | |
2307 2357 2 | BEGIN
2308 2358 2 | |
2309 2359 2 | |
2310 2360 2 | | Check for unwinding.
2311 2361 2 | |
2312 2362 2 | |
2313 2363 2 | IF .SIGNAL_ARGS [CHF$SIG_NAME] EQL S$$UNWIND
2314 2364 2 | THEN
2315 2365 2 | BEGIN
2316 2366 2 | LOCAL
2317 2367 2 | PFV: REF $PASSPFV_FILE_VARIABLE; ! Pascal File Variable
2318 2368 2 |
2319 2369 2 |
2320 2370 2 | |
2321 2371 2 | | If the enable argument for the PFV address is non-zero,
2322 2372 2 | | and if the FCB is valid, close the file and deallocate the FCB.
2323 2373 2 | |
2324 2374 2 | |
2325 2375 2 | PFV = ..ENABLE_ARGS [1];
2326 2376 2 | IF PFV [PFV$R_PFV] NEQA 0
2327 2377 2 | THEN
2328 2378 2 | IF .PFV [PFV$V_FCB_VALID]
2329 2379 2 | THEN
2330 2380 2 | BEGIN
2331 2381 2 | PASS$CLOSE (PFV [PFV$R_PFV]); ! Close the file
2332 2382 2 | PASS$REMOVE_FILE (.PFV [PFV$A_FCB]); ! Deallocate FCB
2333 2383 2 | END;
2334 2384 2 |
2335 2385 2 | |
2336 2386 2 | | If there are KEY XABs to deallocate, deallocate them.
2337 2387 2 | |
2338 2388 2 | |
2339 2389 2 | IF ..ENABLE_ARGS [2] NEQ 0
2340 2390 2 | THEN
2341 2391 2 | PASS$FREE_VM (..ENABLE_ARGS [3], .ENABLE_ARGS [2]);
2342 2392 2 | END;
2343 2393 2 |
2344 2394 2 | RETURN S$$RESIGNAL;
2345 2395 2 |
2346 2396 1 | END; ! End of routine OPEN_HANDLER

```

.EXTRN PASS\$CLOSE, PASS\$REMOVE_FILE

```

0000920 50 04 AC D0 00002 .WORD Save R2,R3 : 2297
8F 04 A0 D1 00006 MOVL SIGNAL_ARGS, R0 : 2363
34 12 0000E CMPL 4(R0), #2336 :
52 0C AC D0 00010 BNEQ 2$ : 2375
53 04 B2 D0 00014 MOVL ENABLE_ARGS, R2 :
18 13 00018 MOVL @4(R2), PFV :
BEQL 1$ : 2376

```

PASS\$OPEN2
1-015

OPEN procedure
OPEN_HANDLER - Close file on unwind

K 15
16-Sep-1984 01:46:15
14-Sep-1984 12:51:41

VAX-11 Bliss-32 V4.0-742
[PASRTL.SRC]PASOPEN2.B32;1

Page 67
(14)

13	07	A3	06	E1	0001A	BBC	#6, 7(PFV), 1\$:	2378
			53	DD	0001F	PUSHL	PFV	:	2381
00000000G	00		01	FB	00021	CALLS	#1, PASS\$CLOSE	:	
			0C	A3	DD 00028	PUSHL	12(PFV)	:	2382
00000000G	00		01	FB	0002B	CALLS	#1, PASS\$REMOVE_FILE	:	
			08	B2	D5 00032	TSTL	@8(R2)	:	2389
				0D	13 00035	BEQL	2\$:	
			08	A2	DD 00037	PUSHL	8(R2)	:	2391
			0C	B2	DD 0003A	PUSHL	@12(R2)	:	
00000000G	00		02	FB	0003D	CALLS	#2, PASS\$FREE_VM	:	
	50		0918	8F	3C 00044	MOVZWL	#2328, R0	:	2394
				04	00049	RET		:	2396

; Routine Size: 74 bytes, Routine Base: _PASS\$CODE + 0CF5

; 2347 2397 1
; 2348 2398 1 !<BLF/PAGE>

```

: 2350 2399 1 %SBTTL 'EXIT_HANDLER - Exit handler for file system'
: 2351 2400 1 ROUTINE EXIT_HANDLER (
: 2352 2401 1     EXIT_REASON: REF VECTOR [, LONG]           ! Exit reason
: 2353 2402 1     ): NOVALUE =
: 2354 2403 1
: 2355 2404 1 !++
: 2356 2405 1 ! FUNCTIONAL DESCRIPTION:
: 2357 2406 1 !
: 2358 2407 1 !     This is the exit handler for the file system. It is declared
: 2359 2408 1 !     by PASS$OPEN, and serves to close all open files in an orderly
: 2360 2409 1 !     fashion upon image exit.
: 2361 2410 1 !
: 2362 2411 1 ! CALLING SEQUENCE:
: 2363 2412 1 !
: 2364 2413 1 !     CALL EXIT_HANDLER (EXIT_REASON.rlc.r)
: 2365 2414 1 !     (Called by VMS upon image exit.)
: 2366 2415 1 !
: 2367 2416 1 ! FORMAL PARAMETERS:
: 2368 2417 1 !
: 2369 2418 1 !     EXIT_REASON     - The reason for the exit. This parameter is
: 2370 2419 1 !                   not used here.
: 2371 2420 1 !
: 2372 2421 1 ! IMPLICIT INPUTS:
: 2373 2422 1 !
: 2374 2423 1 !     NONE
: 2375 2424 1 !
: 2376 2425 1 ! IMPLICIT OUTPUTS:
: 2377 2426 1 !
: 2378 2427 1 !     NONE
: 2379 2428 1 !
: 2380 2429 1 ! COMPLETION STATUS:
: 2381 2430 1 !
: 2382 2431 1 !     NONE
: 2383 2432 1 !
: 2384 2433 1 ! SIDE EFFECTS:
: 2385 2434 1 !
: 2386 2435 1 !     Closes all open files.
: 2387 2436 1 !
: 2388 2437 1 ! SIGNALLED ERRORS:
: 2389 2438 1 !
: 2390 2439 1 !     NONE
: 2391 2440 1 !
: 2392 2441 1 ! --
: 2393 2442 1 !
: 2394 2443 2 ! BEGIN
: 2395 2444 2 !
: 2396 2445 2 ! !+
: 2397 2446 2 ! ! Clear EXITH DECLARED so that if a user exit handler opens more
: 2398 2447 2 ! ! files, another handler will be declared.
: 2399 2448 2 ! !-
: 2400 2449 2 !
: 2401 2450 2 ! EXITH_DECLARED = 0;
: 2402 2451 2 !
: 2403 2452 2 ! !+
: 2404 2453 2 ! ! Call PASS$CLOSE_ALL to close all the files.
: 2405 2454 2 ! !-
: 2406 2455 2

```


