


```
BBBBBBBBB      AAAAAA      SSSSSSSS      000000      PPPPPPPP      EEEEEEEEEEE      NN      NN
BBBBBBBBB      AAAAAA      SSSSSSSS      000000      PPPPPPPP      EEEEEEEEEEE      NN      NN
BB      BB      AA      AA      SS      00      00      PP      PP      EE      NN      NN
BB      BB      AA      AA      SS      00      00      PP      PP      EE      NN      NN
BB      BB      AA      AA      SS      00      00      PP      PP      EE      NNNN      NN
BB      BB      AA      AA      SS      00      00      PP      PP      EE      NNNN      NN
BBBBBBBBB      AA      AA      SSSSSS      00      00      PPPPPPP      EEEEEEEEE      NN      NN
BBBBBBBBB      AA      AA      SSSSSS      00      00      PPPPPPP      EEEEEEEEE      NN      NN
BB      BB      AAAAAAAAAA      SS      00      00      PP      EE      NN      NNNN
BB      BB      AAAAAAAAAA      SS      00      00      PP      EE      NN      NNNN
BB      BB      AA      AA      SS      00      00      PP      EE      NN      NN
BB      BB      AA      AA      SS      00      00      PP      EE      NN      NN
BBBBBBBBB      AA      AA      SSSSSSSS      000000      PP      EEEEEEEEEEE      NN      NN
BBBBBBBBB      AA      AA      SSSSSSSS      000000      PP      EEEEEEEEEEE      NN      NN
                                     ....
                                     ....
                                     ....
                                     ....
```

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



```

1 0001 0
2 0002 0 MODULE BASSOPEN (
3 0003 0 IDENT = '1-113'
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1 *****
8 0008 1 *
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1983, 1984 BY
10 0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
11 0011 1 * ALL RIGHTS RESERVED.
12 0012 1 *
13 0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
14 0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
15 0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
16 0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
17 0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
18 0018 1 * TRANSFERRED.
19 0019 1 *
20 0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
21 0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
22 0022 1 * CORPORATION.
23 0023 1 *
24 0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
25 0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
26 0026 1 *
27 0027 1 *
28 0028 1 *****
29 0029 1
30 0030 1
31 0031 1 ++
32 0032 1 FACILITY: VAX-11 BASIC I/O Processing
33 0033 1
34 0034 1 ABSTRACT:
35 0035 1
36 0036 1 This module contains the BASSOPEN routine, which opens a file
37 0037 1 for a VAX-11 BASIC program.
38 0038 1
39 0039 1 ENVIRONMENT: VAX-11 User Mode
40 0040 1
41 0041 1 AUTHOR: John Sauter, CREATION DATE: 30-NOV-78
42 0042 1
43 0043 1 MODIFIED BY:
44 0044 1
45 0045 1 1-001 - Original. JBS 30-NOV-78
46 0046 1 1-002 - Change REQUIRE file name from FOR... to OTS... JBS 06-DEC-78
47 0047 1 1-003 - Change OPEN$K symbols to LUB$K. JBS 08-DEC-78
48 0048 1 1-004 - REQUIRE BASOPN to get default record length. JBS 12-DEC-78
49 0049 1 1-005 - Call BASSCB_PUSH and POP instead of FOR$. JBS 29-DEC-78
50 0050 1 1-006 - Update this module to use the new OPEN calling sequence
51 0051 1 and semantics decided on in the 12-FEB-1979 meeting.
52 0052 1 JBS 14-FEB-1979
53 0053 1 1-007 - Set LUB$V_USER_RBUF if the record buffer is taken from the
54 0054 1 MAP parameter instead of allocated from virtual storage.
55 0055 1 JBS 16-FEB-1979
56 0056 1 1-008 - Clear the LOCATE bit in the RAB if the user provides his own
57 0057 1 buffer, so the data will always be placed in it. JBS 19-FEB-1979

```


58	0058	1	1-009	- Use I/O error codes from BASIOERR.REQ. JBS 20-FEB-1979
59	0059	1	1-010	- If the device being opened is a terminal, set the right
60	0060	1		margin based on the BLS field of the FAB. JBS 22-FEB-1979
61	0061	1	1-011	- Support the ALLOW SCRATCH clause; it implies no sharing.
62	0062	1		JBS 01-MAR-1979
63	0063	1	1-012	- Add ORGANIZATION INDEXED. JBS 08-MAR-1979
64	0064	1	1-013	- Don't use summary XABs until I can learn how. JBS 08-MAR-1979
65	0065	1	1-014	- Set up the key buffer and size fields of the RAB. JBS 26-MAR-1979
66	0066	1	1-015	- If the user provides a MAP, null it. JBS 04-APR-1979
67	0067	1	1-016	- Accept key data types, if the user provides them.
68	0068	1		JBS 06-APR-1979
69	0069	1	1-017	- Set up LUB\$B_ORGAN on a new file. JBS 06-APR-1979
70	0070	1	1-018	- Verify keys properly for an existing indexed file. JBS 06-APR-1979
71	0071	1	1-019	- If we allocate a prompt buffer, set its current length
72	0072	1		to zero. JBS 09-APR-1979
73	0073	1	1-020	- OPEN on channel 0 gives an error message, and on any other
74	0074	1		open channel closes the channel first. JBS 12-APR-1979
75	0075	1	1-021	- The default record format for VIRTUAL files is FIXED.
76	0076	1		JBS 19-APR-1979
77	0077	1	1-022	- Implement STREAM, since it is in the compiler. Note: the
78	0078	1		rest of the RTL does not implement STREAM. JBS 19-APR-1979
79	0079	1	1-023	- Add LUB\$B_RAT, for the FSP\$ function. JBS 19-APR-1979
80	0080	1	1-024	- Add STATUS. JBS 19-APR-1979
81	0081	1	1-025	- Set LUB\$V_TERM FOR if this is a terminal format file.
82	0082	1		JBS 14-MAY-1979
83	0083	1	1-026	- If RECORDSIZE is specified on a terminal format file, set
84	0084	1		the MARGIN and DEFAULT MARGIN to it. JBS 18-MAY-1979
85	0085	1	1-027	- Don't allow a recordsize of 0 on an old file.
86	0086	1		JBS 22-MAY-1979
87	0087	1	1-028	- Set MRS to the computed record size, in case it defaulted.
88	0088	1		JBS 24-MAY-1979
89	0089	1	1-029	- Correct an obvious bug in STATUS. It has not been tested.
90	0090	1		JBS 24-MAY-1979
91	0091	1	1-030	- Set LUB\$V_FORCIBLE if the device is a terminal.
92	0092	1		JBS 24-MAY-1979
93	0093	1	1-031	- Set RAB\$V_UIF for virtual files. JBS 25-MAY-1979
94	0094	1	1-032	- Change from LUB\$V_NO KEYS to LUB\$V_KEYED. JBS 30-MAY-1979
95	0095	1	1-033	- Change margin to 16 bits. JBS 30-MAY-1979
96	0096	1	1-034	- Add BASS\$STATU_INIT. JBS 04-JUN-1979
97	0097	1	1-035	- Set the language byte in the LUB, so only BASIC I/O
98	0098	1		statements can be used on files opened in BASIC.
99	0099	1		This restriction may be relaxed in some future
100	0100	1		release. JBS 30-JUN-1979
101	0101	1	1-036	- If the device is a terminal, change to PRN format so it
102	0102	1		is forcible. JBS 10-JUL-1979
103	0103	1	1-037	- Implement STREAM for real (see edit 022). JBS 12-JUL-1979
104	0104	1		PRN files must be VFC format. JBS 17-JUL-1979
105	0105	1	1-038	- Add the unit number as the third argument to USEROPEN.
106	0106	1		JBS 25-JUL-1979
107	0107	1	1-040	- Set up ISB\$A_USER_FP. JBS 25-JUL-1979
108	0108	1	1-041	- Make LUB\$Q_BFA_QUEUE empty. JBS 30-JUL-1979
109	0109	1	1-042	- Don't allow ACCESS READ to create a file. JBS 30-JUL-1979
110	0110	1	1-043	- Make sure the LUB/ISB/RAB gets deallocated if we must
111	0111	1		"bail out". JBS 31-JUL-1979
112	0112	1	1-044	- Don't fool with LUB\$B_RSL and LUB\$A_RSN until we are sure
113	0113	1		the LUN is not open. Any error messages should show the
114	0114	1		previous file name. JBS 08-AUG-1979

115	0115	1	1-045 - Set NOTSEQORG for all but terminal and sequential files,
116	0116	1	so BAS\$\$IO BEG can use it. JBS 08-AUG-1979
117	0117	1	1-046 - If ACCESS READ, assume FOR INPUT. JBS 08-AUG-1979
118	0118	1	1-047 - Use the BASIC-specific exit handler. JBS 17-AUG-1979
119	0119	1	1-048 - Be sure to RMS close the file if an error is detected after
120	0120	1	the RMS OPEN is successful. JBS 23-AUG-1979
121	0121	1	1-049 - Rearrange MARGIN and RECORDSIZE defaults for compatability
122	0122	1	with the PDP-11. Now, terminal format files on disk have
123	0123	1	margin of 72, whereas terminal format files on terminals
124	0124	1	have infinite margin. JBS 24-AUG-1979
125	0125	1	1-050 - Correct a typo in the computation of bucket size. JBS 04-SEP-1979
126	0126	1	1-051 - Disable locate mode until we get a chance to debug it.
127	0127	1	JBS 12-SEP-1979
128	0128	1	1-052 - Remove STREAM and add record attributes. JBS 13-SEP-1979
129	0129	1	1-053 - Re-enable locate mode. JBS 13-SEP-1979
130	0130	1	1-054 - Default record attribute for an old virtual file may be NONE or CR.
131	0131	1	JBS 15-SEP-1979
132	0132	1	1-055 - Give an error message for the CONNECT clause until it is implemented.
133	0133	1	JBS 19-SEP-1979
134	0134	1	1-056 - Remove references to LUB\$Q_BFA_QUEUE, no longer needed.
135	0135	1	JBS 19-SEP-1979
136	0136	1	1-057 - Implement CONNECT for indexed files. JBS 30-SEP-1979
137	0137	1	1-058 - Improve the error message for mismatch of record attributes.
138	0138	1	JBS 03-OCT-1979
139	0139	1	1-059 - Don't demand any particular record format if the organization is
140	0140	1	is UNDEFINED. JBS 12-OCT-1979
141	0141	1	1-060 - Allow any record format, even UNDEFINED, if the organization is
142	0142	1	undefined. JBS 12-OCT-1979
143	0143	1	1-061 - If this is a VFC file, make sure the VFC field size is right.
144	0144	1	JBS 15-OCT-1979
145	0145	1	1-062 - If the argument list says FOR OUTPUT, set the FAB\$V_SUP bit, so
146	0146	1	that an explicit version number in the file name will delete
147	0147	1	and recreate an existing file. QAR N11-02971 JBS 22-OCT-1979
148	0148	1	1-063 - Round the block size up to a multiple of 4 bytes. JBS 25-OCT-1979
149	0149	1	1-064 - Fix an error message. JBS 07-NOV-1979
150	0150	1	1-065 - Improve the interface to USEROPEN. JBS 07-NOV-1979
151	0151	1	1-066 - Change virtual arrays to automatic record locking. JBS 09-NOV-1979
152	0152	1	1-067 - Set up LIB\$A_UBF. JBS 13-NOV-1979
153	0153	1	1-068 - Make sure the record buffer is at least as large as the user's
154	0154	1	declared "buffer size", or the default for the organization,
155	0155	1	if none was declared. JBS 27-NOV-1979
156	0156	1	1-069 - Don't ever turn on RMS Locate Mode. DGP 29-Nov-79
157	0157	1	1-070 - Correct an error in edit 1-068. JBS 05-DEC-1979
158	0158	1	1-071 - If the record size test fails on an existing file, give error
159	0159	1	BAD RECORDSIZE VALUE ON OPEN rather than FILE ATTRIBUTES NOT
160	0160	1	MATCHED. This error was found by FEATS. JBS 28-DEC-1979
161	0161	1	1-072 - Change "Run Time Syntax Error" to "Program Lost Sorry". DGP 07-Jan-80
162	0162	1	1-073 - Change OPEN_HANDLER to clean up the I/O data base before signalling
163	0163	1	in the event of a severe error. DGP 08-Jan-80
164	0164	1	1-074 - Complete edit 1-073. JBS 09-JAN-1980
165	0165	1	1-075 - Improve compatability with the PDP-11: round block size up to 20,
166	0166	1	don't set FAB\$V_RWC and don't check FAB\$B_RFM if it is defaulted.
167	0167	1	JBS 14-JAN-1980
168	0168	1	1-076 - Another compatability change: If the ALLOW clause is omitted, let
169	0169	1	RMS default the FAB\$B_SHR field. JBS 15-JAN-1980
170	0170	1	1-077 - The bucket size field is a word, not a byte. JBS 01-FEB-1980
171	0171	1	1-078 - Use 1 for default blocksize in Open. DGP 12-Feb-1980

172	0172	1	1-079	- Always round recordsize for virtual files to next higher multiple of 512. Check file attributes against temp RSZ. DGP 12-Feb-80
173	0173	1		
174	0174	1	1-080	- Complete edit 1-079. JBS 13-FEB-1980
175	0175	1	1-081	- A Virtual file must be sequential and have fixed-length records. The buffer size (record size) must be greater than or equal to 512 bytes. If it is less, round it up unless the user supplied a map, in which case we have an error. Round the record size down if necessary to make it even. JBS 12-JUN-1980
176	0176	1		
177	0177	1		
178	0178	1		
179	0179	1		
180	0180	1	1-082	- Put FAB\$ALQ in LUB\$ALQ and initialize LUB\$REC_MAX. FM 22-SEP-80
181	0181	1	1-083	- In order to be able to OPEN a spooled terminal just like any other terminal, make the \$OPEN macro a \$CREATE macro after we see that the file is a terminal and we close and reopen (reCREATE). FM 4-FEB-81
182	0182	1		
183	0183	1		
184	0184	1	1-084	- If organization is UNDEFINED, and recordsize is 0 then signal BADRECVAL. FM 5-FEB-81
185	0185	1		
186	0186	1	1-085	- Set FAC to null before invoking macro \$FAB_INIT. If this is not done the opener gets read access automatically. PL 20-AUG-81
187	0187	1		
188	0188	1	1-086	- Set FAB\$V_MSE unconditionally if organization is indexed, so a subsequent CONNECT can work. Also, if caller is trying to CONNECT to a child, instead of a parent signal INVFILOPT. FM 20-AUG-81
189	0189	1		
190	0190	1		
191	0191	1	1-087	- LIB\$STOP should be declared EXTERNAL. PLL 20-Nov-81
192	0192	1	1-088	- If the user specifies a recordsize less than the file record size, do not give an error. PLL 13-Jan-82
193	0193	1		
194	0194	1	1-089	- Check the recordsize of a variable length record file only if MRS is set. PLL 22-Feb-82
195	0195	1		
196	0196	1	1-090	- Check 089 should be made only if the file is opened for write access. Rather than checking the recordsize of a variable length record file, just let RMS catch the error. PLL 9-Mar-1982
197	0197	1		
198	0198	1		
199	0199	1	1-091	- Yet another fix for recordsize. RBUF can be larger than the recordsize specified in the OPEN block. PLL 3-May-1982
200	0200	1		
201	0201	1	1-092	- Add support for manual record locking. If the user specifies 'UNLOCK EXPLICIT' in the OPEN statement, set the ULK bit in the RAB ROP. PLL 2-Jun-1982
202	0202	1		
203	0203	1		
204	0204	1	1-093	- Add check for recordsize that exceeds the MAP buffer size. PLL 2-Jun-1982
205	0205	1	1-094	- Add support for segmented keys. PLL 3-Jun-1982
206	0206	1	1-095	- Fix bug in setting RAB ROP ULK bit. PLL 8-Jun-1982
207	0207	1	1-096	- For V2, if the RECORDSIZE is 0, use the MAP size. PLL 9-Jun-1982
208	0208	1	1-097	- For segmented keys, the lengths and the positions of all segments specified by the user must be checked. PLL 9-Jun-1982
209	0209	1		
210	0210	1	1-098	- For V2, if there is no MAP don't compare the MAP size to the recordsize. PLL 11-Jun-1982
211	0211	1		
212	0212	1	1-099	- Fix problem of two files being queued when output file is a spooled terminal. Instead of CLOSEing and re-CREATEing after initial CREATE tells us the device is a spooled terminal, simply do a PARSE to see if it is, and fix up the FAB and RAB accordingly prior to the CREATE. MDL 16-Jun-1982
213	0213	1		
214	0214	1		
215	0215	1		
216	0216	1		
217	0217	1	1-100	- Add check for printer as output "file" on OPEN statement; set default and right margins to printer width if it is. MDL 17-Jun-1982
218	0218	1		
219	0219	1	1-101	- Move recordsize/map size check so that it includes new files also. PLL 2-Jul-1982
220	0220	1		
221	0221	1	1-102	- when CONNECT is specified but indexed is not, allow check of parent file attributes (to see if its indexed & therefore OK) before giving an error. MDL 19-Jul-1982
222	0222	1		
223	0223	1		
224	0224	1	1-103	- when UNLOCK EXPLICIT is specified for a non-512 byte fixed-length sequential file, issue an error. MDL 20-Jul-1982
225	0225	1		
226	0226	1	1-104	- only check for UNLOCK EXPLICIT on V2 programs. MDL 21-Sep-1982
227	0227	1	1-105	- enabled commented-out code on summary XABs. Added check of declared key type vs. existing key type for indexed files.
228	0228	1		


```
: 229      0229 1  MDL 28-Sep-1982
: 230      0230 1  1-106 - for V2 programs, if there is not a recordsize but there is a map
: 231      0231 1  size, then the map size should be used to as the default right
: 232      0232 1  margin rather than the BASIC default. MDL 29-Sep-1982
: 233      0233 1  1-107 - for indexed files, only give an error if the number of keys the
: 234      0234 1  user specifies is MORE than the number of keys on the file, rather
: 235      0235 1  than looking for an exact match. MDL 15-Oct-1982
: 236      0236 1  1-108 - update UNWIND CCB when we have to drop R11 and pick it up again,
: 237      0237 1  in the case of opening a channel that's already open somewhere
: 238      0238 1  else (and therefore needs to be closed first). MDL 5-Jan-1983
: 239      0239 1  1-109 - for V2 programs, signal BADRECVAL if the user attempts to open
: 240      0240 1  a file with MRS with a MAP that is too small. MDL 6-Jan-1983
: 241      0241 1  1-110 - don't be quite so picky about key data types matching existing
: 242      0242 1  indexed files - Basic has no unsigned data type, but indexed files
: 243      0243 1  can have unsigned keys. MDL 29-Aug-1983
: 244      0244 1  1-111 - allow OPEN of SYSS$INPUT with ORG=UNDEFINED. also, look at FAB$W BLS
: 245      0245 1  to determine actual recordsize, along with MRS and LRL. MDL 22-Feb-1984
: 246      0246 1  1-112 - take recordsize from the file if it isn't specified. MDL 3-May-1984
: 247      0247 1  1-113 - Fix 1-112. Use recordsize from the file only if the recordsize
: 248      0248 1  was not specified and the maximum recordsize (FAB$W_MRS) is
: 249      0249 1  not zero. KC and MDL and AKS 07-Sep-1984.
: 250      0250 1  --
: 251      0251 1
: 252      0252 1  !<BLF/PAGE>
```



```
254 0253 1 |
255 0254 1 | SWITCHES:
256 0255 1 |
257 0256 1 |
258 0257 1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
259 0258 1 |
260 0259 1 |
261 0260 1 | LINKAGES:
262 0261 1 |
263 0262 1 |
264 0263 1 REQUIRE 'RTLIN:OTSLNK';           ! define linkages
265 0692 1 |
266 0693 1 |
267 0694 1 | TABLE OF CONTENTS:
268 0695 1 |
269 0696 1 |
270 0697 1 FORWARD ROUTINE
271 0698 1     BASSOPEN : NOVALUE,           ! Open a file
272 0699 1     OPEN_HANDLER,               ! Condition handler for OPEN
273 0700 1     BASSSTATUS,                 ! Status of last file opened
274 0701 1     BASS$STATU_INIT : NOVALUE;   ! Initialize status for RUN command
275 0702 1 |
276 0703 1 |
277 0704 1 | INCLUDE FILES:
278 0705 1 |
279 0706 1 |
280 0707 1 REQUIRE 'RTLIN:RTLPSECT';         ! Macros for defining psects
281 0802 1 |
282 0803 1 REQUIRE 'RTLML:OTSLUB';           ! Logical Unit Block definitions
283 0943 1 |
284 0944 1 REQUIRE 'RTLML:OTSISB';           ! ISB definitions
285 1112 1 |
286 1113 1 REQUIRE 'RTLIN:BASOPN';           ! OPEN literals
287 1161 1 |
288 1162 1 REQUIRE 'RTLIN:BASIOERR';         ! I/O error codes
289 1215 1 |
290 1216 1 LIBRARY 'RTLSTARLE';              ! system definitions
291 1217 1 |
292 1218 1 |
293 1219 1 | MACROS:
294 1220 1 |
295 1221 1 |     NONE
296 1222 1 |
297 1223 1 | EQUATED SYMBOLS:
298 1224 1 |
299 1225 1 | +
300 1226 1 | The following symbols refer to arguments of BASSOPEN.
301 1227 1 | -
302 1228 1 |
303 1229 1 LITERAL
304 1230 1     OPNSK_ORG_TERMI = 0,           ! no organization specified
305 1231 1     OPNSK_ORG_VIRTU = 1,           ! ORGANIZATION VIRTUAL specified
306 1232 1     OPNSK_ORG_SEQUE = 2,           ! ORGANIZATION SEQUENTIAL specified
307 1233 1     OPNSK_ORG_RELAT = 3,           ! ORGANIZATION RELATIVE specified
308 1234 1     OPNSK_ORG_INDEX = 4,           ! ORGANIZATION INDEXED specified
309 1235 1     OPNSK_ORG_UNDEF = 5,           ! ORGANIZATION UNDEFINED specified
310 1236 1     OPNSK_ACC_DEFAU = 0,           ! no access specified ( = MODIFY)
```



```
311 1237 1 OPNSK_ACC_MODIF = 1,      ! ACCESS MODIFY specified
312 1238 1 OPNSK_ACC_WRITE = 2,     ! ACCESS WRITE specified
313 1239 1 OPNSK_ACC_READ = 3,      ! ACCESS READ specified
314 1240 1 OPNSK_ACC_SCRAT = 4,     ! ACCESS SCRATCH specified
315 1241 1 OPNSK_ACC_APPEN = 5,     ! ACCESS APPEND specified
316 1242 1 OPNSK_RFM_DEFAU = 0,     ! record format not specified
317 1243 1 OPNSK_RFM_FIXED = 1,     ! FIXED specified
318 1244 1 OPNSK_RFM_VARIA = 2,     ! VARIABLE specified
319 1245 1 OPNSK_RFM_VFC = 3,       ! VFC specified
320 1246 1 OPNSK_RFM_STREA = 4,     ! STREAM specified
321 1247 1 OPNSK_ALL_DEFAU = 0,     ! no ALLOW clause
322 1248 1 OPNSK_ALL_MODIF = 1,     ! ALLOW MODIFY specified
323 1249 1 OPNSK_ALL_WRITE = 2,     ! ALLOW WRITE specified
324 1250 1 OPNSK_ALL_READ = 3,      ! ALLOW READ specified
325 1251 1 OPNSK_ALL_SCRAT = 4,     ! ALLOW SCRATCH specified
326 1252 1 OPNSK_ALL_NONE = 6,     ! ALLOW NONE specified
327 1253 1 OPNSK_RAT_DEFAU = 0,     ! no RECORDATTR clause
328 1254 1 OPNSK_RAT_FORTR = 1,     ! RECORDATTR FORTRAN specified
329 1255 1 OPNSK_RAT_CRLF = 2,     ! RECORDATTR CRLF specified
330 1256 1 OPNSK_RAT_NONE = 3,     ! RECORDATTR NONE specified
331 1257 1 OPNSK_RAT_PRINT = 4,     ! RECORDATTR PRINT specified
332 1258 1 OPNSK_RAT_ANY = 5;       ! RECORDATTR ANY specified
333 1259 1
334 1260 1 !+
335 1261 1 ! The following codes specify how OPEN_HANDLER is to act on an UNWIND.
336 1262 1 !-
337 1263 1
338 1264 1 LITERAL
339 1265 1 UNWIND_MIN = 0,           ! Minimum UNWIND code
340 1266 1 UNWIND_NOP = 0,          ! Do nothing
341 1267 1 UNWIND_POP = 1,          ! POP the CCB
342 1268 1 UNWIND_DEALLOC = 2,     ! Deallocate (and POP) the CCB
343 1269 1 UNWIND_CLOSE = 3,       ! RMS CLOSE the CCB, then deallocate and POP
344 1270 1 UNWIND_MAX = 3;         ! Maximum UNWIND code
345 1271 1
346 1272 1 LITERAL
347 1273 1 K_V1_BLK_SIZE = 52;     ! Size of V1 OPEN argument block
348 1274 1
349 1275 1
350 1276 1 PSECTS:
351 1277 1
352 1278 1 DECLARE_PSECTS (BAS);    ! Declare psects for BAS$ facility
353 1279 1
354 1280 1 OWN STORAGE:
355 1281 1
356 1282 1
357 1283 1 OWN
358 1284 1 L_STATUS : INITIAL (0);  ! Status of last file opened.
359 1285 1
360 1286 1
361 1287 1 EXTERNAL REFERENCES:
362 1288 1
363 1289 1
364 1290 1 EXTERNAL
365 1291 1 BAS$$_L_XIT_LOCK;         ! Once-only flag for the exit handler.
366 1292 1
367 1293 1 EXTERNAL ROUTINE
```



```

: 368      1294 1      LIB$STOP : NOVALUE,      | Signal fatal error
: 369      1295 1      BAS$$STOP : NOVALUE,      | Signals fatal error
: 370      1296 1      BAS$$STOP IO : NOVALUE,      | Signals fatal I/O error
: 371      1297 1      BAS$$CB_PUSH : JSB_CB_PUSH NOVALUE, | loads register CCB
: 372      1298 1      BAS$$CB_POP : JSB_CB_POP NOVALUE, | completes LUN processing
: 373      1299 1      LIB$GET_VM,      | get virtual storage
: 374      1300 1      LIB$FREE_VM,      | free virtual storage
: 375      1301 1      BAS$$DECC_EXITH : NOVALUE,      | Declare exit handler
: 376      1302 1      OTS$$TAKE_LUN,      | Sign out a logical unit number
: 377      1303 1      OTS$$CLOSE_FILE,      | RMS Close a file
: 378      1304 1      LIB$MATCH_COND;      | Match a condition code
: 379      1305 1
: 380      1306 1
: 381      1307 1      !+ The following are the error codes used in this module.
: 382      1308 1      !-
: 383      1309 1
: 384      1310 1      EXTERNAL LITERAL
: 385      1311 1      BAS$K_RECOVERMAP : UNSIGNED (8),      | RECORDSIZE overflows MAP buffer
: 386      1312 1      BAS$K_PROLOSSOR : UNSIGNED (8),      | Program lost, sorry
: 387      1313 1      BAS$K_ILLILLACC : UNSIGNED (8),      | Illegal or illogical access
: 388      1314 1      BAS$K_ILLIO_CHA : UNSIGNED (8),      | Illegal I/O channel
: 389      1315 1      BAS$K_IO_CHAALR : UNSIGNED (8),      | I/O channel already open
: 390      1316 1      BAS$K_FATSYSIO : UNSIGNED (8),      | Fatal system I/O error
: 391      1317 1      BAS$K_FILATTNOT : UNSIGNED (8),      | File attributes not matched
: 392      1318 1      BAS$K_RECATTNOT : UNSIGNED (8),      | Record attributes not matched
: 393      1319 1      BAS$K_MAXMEMEXC : UNSIGNED (8),      | Maximum memory exceeded
: 394      1320 1      BAS$K_BADRECVAL : UNSIGNED (8),      | Bad recordsize value on OPEN
: 395      1321 1      BAS$K_NOTIMP : UNSIGNED (8),      | Not implemented
: 396      1322 1      BAS$K_INVFILOPT : UNSIGNED (8),      | Invalid file option
: 397      1323 1      BAS$K_IO_CHANOT : UNSIGNED (8),      | I/O Channel not open
: 398      1324 1      BAS$K_REQRECSIZ : UNSIGNED (8);      | unlock explicit requires recordsize 512
: 399      1325 1
: 400      1326 1      !<BLF/PAGE>
```



```
402 1327 1 !+
403 1328 1 ! The following FIELD describes the OPEN_ARG_BLK argument of
404 1329 1 ! BAS$OPEN.
405 1330 1 !-
406 1331 1
407 1332 1 FIELD
408 1333 1     OPN$ARG_BLOCK =
409 1334 1     SET
410 1335 1         OPN$B_CNT = [0, 0, 8, 0],      ! number of remaining bytes
411 1336 1         OPN$B_ORG = [1, 0, 8, 1],      ! ORGANIZATION clause
412 1337 1         OPN$B_ACCESS = [2, 0, 8, 1],  ! ACCESS clause
413 1338 1         OPN$B_RFM = [3, 0, 8, 1],      ! record format
414 1339 1         OPN$B_ALLOW = [4, 0, 8, 1],     ! ALLOW clause
415 1340 1         OPN$V_NOSPAN = [5, 0, 1, 0], ! NO SPAN specified
416 1341 1         OPN$V_CONTIGUOUS = [5, 1, 1, 0], ! CONTIGUOUS specified
417 1342 1         OPN$V_NO_REWIND = [5, 2, 1, 0], ! NO REWIND specified
418 1343 1         OPN$V_TEMPORARY = [5, 3, 1, 0], ! TEMPORARY specified
419 1344 1         OPN$V_FOR_OUTPUT = [5, 4, 1, 0], ! FOR OUTPUT specified
420 1345 1         OPN$V_FOR_INPUT = [5, 5, 1, 0], ! FOR INPUT specified
421 1346 1         OPN$V_CHAN_REF = [5, 6, 1, 0], ! OPN$L_CHANNEL points to channel number
422 1347 1         OPN$V_CHAN_WORD = [5, 7, 1, 0], ! Channel is 16 bits (ref only)
423 1348 1         OPN$B_RAT = [6, 0, 8, 1],     ! RECORDATTR clause
424 1349 1         OPN$B_FSZ = [7, 0, 8, 0],      ! Size of VFC buffer
425 1350 1         OPN$T_FILE_SPEC = [8, 0, %BPADDR, 0], ! descriptor for file name string
426 1351 1         OPN$W_RECORDSIZ = [12, 0, 16, 0], ! size of record buffer in bytes
427 1352 1         OPN$W_BLOCKSIZE = [14, 0, 16, 0], ! size of magtape block in bytes
428 1353 1         OPN$A_MAP = [16, 0, %BPADDR, 0], ! address of user record buffer
429 1354 1         OPN$L_FILESIZE = [20, 0, %BPVAL, 1], ! size of file in blocks
430 1355 1         OPN$W_EXTENDSIZ = [24, 0, 16, 1], ! extend size
431 1356 1         OPN$W_BUCKETSIZ = [26, 0, 16, 1], ! bucket size in records
432 1357 1         OPN$B_WINDOWSIZ = [28, 0, 8, 1], ! number of retrieval pointers
433 1358 1         OPN$W_CONNECT = [30, 0, 16, 1], ! channel (LUN) to connect to
434 1359 1         OPN$B_MULTIBUFF = [32, 0, 8, 1], ! number of I/O buffers
435 1360 1         OPN$T_DEFAULTNA = [36, 0, %BPADDR, 0], ! default name string descriptor
436 1361 1         OPN$A_USEROPEN = [40, 0, %BPADDR, 0], ! address of user open procedure
437 1362 1         OPN$L_CHANNEL = [44, 0, %BPVAL, 0], ! Channel number, or its address
438 1363 1         OPN$A_VFC = [48, 0, %BPADDR, 0], ! Address of VFC buffer
439 1364 1         OPN$W_MAP_SIZE = [52, 0, 16, 0], ! Size of MAP buffer
440 1365 1         OPN$V_UNLOCK = [54, 0, 1, 0], ! set for manual record locking
441 1366 1     TES;
442 1367 1
443 1368 1 !+
444 1369 1 ! The following two fields describe the header on the KEY_INFO_BLK parameter
445 1370 1 ! and each key, respectively.
446 1371 1 !-
447 1372 1
448 1373 1 FIELD
449 1374 1     OPN$KEYH_BLOCK =
450 1375 1     SET
451 1376 1         KEYH$B_KEYNUM = [0, 0, 8, 0],      ! Number of keys in this block
452 1377 1         KEYH$B_LEN = [1, 0, 8, 0],        ! Length of each KEY field
453 1378 1     TES;
454 1379 1
455 1380 1 LITERAL
456 1381 1     KEYH$K_LENGTH = 4;      ! Length of the key header
457 1382 1
458 1383 1 FIELD
```



```

: 459      1384 1      OPNSKEY_BLOCK =
: 460      1385 1      SET
: 461      1386 1      KEY$B_LEN = [0, 0, 8, 0],
: 462      1387 1      KEY$B_FLAGS = [1, 0, 8, 0],
: 463      1388 1      KEY$V_DUP = [1, 0, 1, 0],
: 464      1389 1      KEY$V_CHG = [1, 1, 1, 0],
: 465      1390 1      KEY$W_OFFSET = [2, 0, 16, 0],
: 466      1391 1      KEY$B_DTYPE = [4, 0, 8, 0],
: 467      1392 1      KEY$B_NUM_SEG = [5, 0, 8, 0],
: 468      1393 1      KEY$B_LEN1 = [6, 0, 8, 0],
: 469      1394 1      KEY$W_OFFSET1 = [7, 0, 16, 0],
: 470      1395 1      KEY$B_LEN2 = [9, 0, 8, 0],
: 471      1396 1      KEY$W_OFFSET2 = [10, 0, 16, 0],
: 472      1397 1      KEY$B_LEN3 = [12, 0, 8, 0],
: 473      1398 1      KEY$W_OFFSET3 = [13, 0, 16, 0],
: 474      1399 1      KEY$B_LEN4 = [15, 0, 8, 0],
: 475      1400 1      KEY$W_OFFSET4 = [16, 0, 16, 0],
: 476      1401 1      KEY$B_LEN5 = [18, 0, 8, 0],
: 477      1402 1      KEY$W_OFFSET5 = [19, 0, 16, 0],
: 478      1403 1      KEY$B_LEN6 = [21, 0, 8, 0],
: 479      1404 1      KEY$W_OFFSET6 = [22, 0, 16, 0],
: 480      1405 1      KEY$B_LEN7 = [24, 0, 8, 0],
: 481      1406 1      KEY$W_OFFSET7 = [25, 0, 16, 0]
: 482      1407 1      TES;
: 483      1408 1

```

Length of this key
Flags, as follows:
If set, duplicates are allowed
If set, changes are allowed
Offset of this key in the record
Data type of key
Additional # of segments
length segment 1
position segment 1
length segment 2
position segment 2
length segment 3
position segment 3
length segment 4
position segment 4
length segment 5
position segment 5
length segment 6
position segment 6
length segment 7
position segment 7


```

485 1409 1 GLOBAL ROUTINE BASSOPEN (
486 1410 1     OPEN_ARG_BLK,
487 1411 1     KEY_INFO_BLK
488 1412 1 ) : NOVALUE =
489 1413 1
490 1414 1 ++
491 1415 1 FUNCTIONAL DESCRIPTION:
492 1416 1
493 1417 1     Open a file for BASIC. This will always be an RMS file.
494 1418 1     Not all combinations of input options are valid.
495 1419 1
496 1420 1 FORMAL PARAMETERS:
497 1421 1
498 1422 1     OPEN_ARG_BLK.mz.r      A long list of OPEN options.
499 1423 1     KEY_INFO_BLK.rl.ra     The keys, for ISAM opens.
500 1424 1
501 1425 1 IMPLICIT INPUTS:
502 1426 1
503 1427 1     NONE
504 1428 1
505 1429 1 IMPLICIT OUTPUTS:
506 1430 1
507 1431 1     A lot of fields in the LUB.
508 1432 1
509 1433 1 ROUTINE VALUE:
510 1434 1 COMPLETION CODES:
511 1435 1
512 1436 1     NONE
513 1437 1
514 1438 1 SIDE EFFECTS:
515 1439 1
516 1440 1     Either opens a file, thus permitting use of the channel number
517 1441 1     by BASIC I/O statements, or calls BAS$$STOP, thus not returning
518 1442 1     to its caller.
519 1443 1
520 1444 1 --
521 1445 1
522 1446 2 BEGIN
523 1447 2
524 1448 2 BUILTIN
525 1449 2     ACTUALCOUNT;
526 1450 2
527 1451 2 MAP
528 1452 2     OPEN_ARG_BLK : REF BLOCK [0, BYTE] FIELD (OPN$ARG_BLOCK),
529 1453 2     KEY_INFO_BLK : REF BLOCK [0, BYTE] FIELD (OPN$KEYR_BLOCK);
530 1454 2
531 1455 2 LOCAL
532 1456 2     RSZ,
533 1457 2     BKS,
534 1458 2     BLS,
535 1459 2     NO_MAP_REC_SPECIFIED,
536 1460 2
537 1461 2
538 1462 2     OPEN_STATUS,
539 1463 2     CONNECT_STATUS,
540 1464 2     CHANNEL,
541 1465 2     FAB_BLOCK : $FAB_DECL,
```

! Open a file
! Argument list
! Keys for ISAM

! Computed record size
! Computed bucket size
! Computed blocksize
! Flag for W RBUF SIZE
! computation. Set in RSZ
! computation
! RMS status returned by \$OPEN
! RMS status returned by \$CONNECT
! The channel number
! Local FAB


```

: 542      1466 2      FAB : REF $FAB_DECL,      ! pointer to FAB
: 543      1467 2      NAM_BLOCK : $NAM_DECL,    ! local NAM block
: 544      1468 2      XABFHC : $XABFHC_DECL,    ! local XABFHC block
: 545      1469 2      XABSUM : $XABSUM_DECL,    ! local XABSUM block
: 546      1470 2      FILE_NAME_DESC : REF BLOCK [0, BYTE], ! descriptor for file name
: 547      1471 2      FILE_NAME : BLOCK [NAM$C_MAXRSS, BYTE], ! text for file name
: 548      1472 2      UNWIND_ACTION : VOLATILE,  ! What to do on UNWIND
: 549      1473 2      UNWIND_CCB : VOLATILE;    ! What CCB to do it to
: 550      1474 2
: 551      1475 2      GLOBAL REGISTER
: 552      1476 2      CCB = K_CCB_REG : REF BLOCK [0, BYTE]; ! points to LUB
: 553      1477 2
: 554      1478 2      +
: 555      1479 2      Maintenance note: When detecting an error, the best routine to call is
: 556      1480 2      BAS$$STOP_IO, since it will print the channel number and file name.
: 557      1481 2      However, it should not be called until after the file name and channel
: 558      1482 2      number are stored in the LUB and FAB. The best time to call BAS$$STOP_IO
: 559      1483 2      is after the RMS $OPEN, since the file name will then be the resultant or
: 560      1484 2      expanded name from RMS, which will have defaults merged and logical names
: 561      1485 2      translated. Thus, deferring recognizing errors until after the RMS $OPEN
: 562      1486 2      is worth while. Of course, some errors cannot reasonably be deferred to
: 563      1487 2      that point, such as a channel number being out of range, and for these
: 564      1488 2      the BAS$$STOP routine is used to signal errors.
: 565      1489 2      -
: 566      1490 2      +
: 567      1491 2      Set up condition handler for UNWINDs
: 568      1492 2      -
: 569      1493 2
: 570      1494 2      ENABLE
: 571      1495 2      OPEN_HANDLER (UNWIND_ACTION, UNWIND_CCB);
: 572      1496 2
: 573      1497 2      +
: 574      1498 2      Initial action is NOP
: 575      1499 2      -
: 576      1500 2      UNWIND_ACTION = UNWIND_NOP;
: 577      1501 2      +
: 578      1502 2      Fetch the channel number from the user. Later we may need the
: 579      1503 2      address.
: 580      1504 2      -
: 581      1505 2      CHANNEL = .OPEN_ARG_BLK [OPN$L_CHANNEL];
: 582      1506 2
: 583      1507 3      IF (.OPEN_ARG_BLK [OPN$V_CHAN_REF])
: 584      1508 2      THEN
: 585      1509 2
: 586      1510 3      IF (.OPEN_ARG_BLK [OPN$V_CHAN_WORD])
: 587      1511 2      THEN
: 588      1512 2      CHANNEL = .BLOCK [.CHANNEL, 0, 0, 16, 1]
: 589      1513 2      ELSE
: 590      1514 2      CHANNEL = .BLOCK [.CHANNEL, 0, 0, %BPVAL, 1];
: 591      1515 2
: 592      1516 2      +
: 593      1517 2      If the channel number is negative or too large, we have a fatal error.
: 594      1518 2      -
: 595      1519 2
: 596      1520 2      IF ((.CHANNEL LSS LUB$K_LUN_MIN) OR (.CHANNEL GTR LUB$K_LUN_MAX)) THEN BAS$$STOP (BAS$K_ILLIO_CHA);
: 597      1521 2
: 598      1522 2      !+
```



```
599 1523 2 | If the channel number is zero, the OPEN fails.
600 1524 2 | -
601 1525 2 |
602 1526 2 | IF (.CHANNEL EQL 0) THEN BASS$STOP (BASS$_ILLIO_CHA);
603 1527 2 |
604 1528 2 | +
605 1529 2 | Compute some auxiliary variables which we will need later.
606 1530 2 | -
607 1531 2 | NO MAP_REC SPECIFIED = 0;
608 1532 2 | RSZ = .OPEN_ARG_BLK [OPN$_RECORDSIZE];
609 1533 2 |
610 1534 2 | IF (.RSZ EQL 0)
611 1535 2 | THEN
612 1536 2 | BEGIN
613 1537 2 | +
614 1538 2 | Provide a reasonable default value for the record size, when we
615 1539 2 | can determine one. For V2 programs, the MAP size can be used if
616 1540 2 | no RECORDSIZE was specified.
617 1541 2 | -
618 1542 2 | IF .OPEN_ARG_BLK [OPN$_CNT] GTR K_V1_BLK_SIZE AND
619 1543 2 | .OPEN_ARG_BLK [OPN$_MAP_SIZE] NEQ 0
620 1544 2 | THEN
621 1545 2 | RSZ = .OPEN_ARG_BLK [OPN$_MAP_SIZE]
622 1546 2 | ELSE
623 1547 2 | BEGIN
624 1548 2 | NO MAP_REC SPECIFIED = 1;
625 1549 2 | RSZ = (CASE .OPEN_ARG_BLK [OPN$_ORG] FROM OPN$_ORG_TERM1 TO OPN$_ORG_UNDEF OF
626 1550 2 | SET
627 1551 2 | [OPN$_ORG_TERM1, OPN$_ORG_SEQUE] : BASS$_DEF_RECLE;
628 1552 2 | [OPN$_ORG_VIRTU] : 512;
629 1553 2 | [INRANGE, OUTRANGE] : 0;
630 1554 2 | TES);
631 1555 2 | END
632 1556 2 | END
633 1557 2 | ELSE
634 1558 2 | +
635 1559 2 | For virtual files, round the record size down if necessary to make
636 1560 2 | it even.
637 1561 2 | -
638 1562 2 |
639 1563 2 | IF (.OPEN_ARG_BLK [OPN$_ORG] EQL OPN$_ORG_VIRTU)
640 1564 2 | THEN
641 1565 2 | BEGIN
642 1566 2 | RSZ = (.RSZ AND -2);
643 1567 2 | +
644 1568 2 | If there is no map, round the record size up to 512 if it is less.
645 1569 2 | We don't want to give an error message here because the file name is
646 1570 2 | not yet set up and the error message needs to include the file name,
647 1571 2 | so we don't check here for having a map less than
648 1572 2 | 512 bytes long; instead a check is made after the RMS $OPEN for the
649 1573 2 | record size still being less than 512 bytes.
650 1574 2 | -
651 1575 2 |
652 1576 2 | IF (.OPEN_ARG_BLK [OPN$_MAP] EQL 0) THEN RSZ = MAX (.RSZ, 512);
653 1577 2 |
654 1578 2 | END;
655 1579 2 |
```



```

: 656      1580 2      BKS = .OPEN_ARG_BLK [OPNSW_BUCKETSIZ];
: 657      1581 2
: 658      1582 3      IF (.BKS NEQ 0)
: 659      1583 2      THEN
: 660      1584 2      +
: 661      1585 2      - Compute the bucket size in RMS terms
: 662      1586 2
: 663      1587 5      BKS = (511 + (.BKS*(.RSZ +
: 664      1588 6      BEGIN
: 665      1589 6
: 666      1590 6      IF (.OPEN_ARG_BLK [OPNSB_RFM] EQL OPNSK_RFM_VARIA) THEN 2 ELSE 0
: 667      1591 6
: 668      1592 6      END
: 669      1593 5      +
: 670      1594 6      BEGIN
: 671      1595 6
: 672      1596 6      IF (.OPEN_ARG_BLK [OPNSB_ORG] EQL OPNSK_ORG_INDEX) THEN 7 ELSE 1
: 673      1597 6
: 674      1598 6      END
: 675      1599 3      )) +
: 676      1600 4      BEGIN
: 677      1601 4
: 678      1602 4      IF (.OPEN_ARG_BLK [OPNSB_ORG] EQL OPNSK_ORG_INDEX) THEN 15 ELSE 0
: 679      1603 4
: 680      1604 4      END
: 681      1605 2      )/512;
: 682      1606 2
: 683      1607 2      +
: 684      1608 2      - Compute the block size in RMS terms in case this is a new file on mag tape.
: 685      1609 2      The size is rounded up to the nearest 4 bytes for compatability with the
: 686      1610 2      PDP-11.
: 687      1611 2      -
: 688      1612 5      BLS = MAX (((MAX (1, .OPEN_ARG_BLK [OPNSW_BLOCKSIZE])*
: 689      1613 6      BEGIN
: 690      1614 6
: 691      1615 6      IF (.OPEN_ARG_BLK [OPNSB_RFM] EQL OPNSK_RFM_VARIA) THEN .RSZ + 4 ELSE .RSZ
: 692      1616 6
: 693      1617 6      END
: 694      1618 6      ) + 3) AND ( NOT 3)), 20);
: 695      1619 2      +
: 696      1620 2      - Allocate the LUB, ISB and RAB if they have not already been
: 697      1621 2      allocated for this LUN. Push down if there is already an I/O
: 698      1622 2      statement in progress on another unit. On return, CCB points
: 699      1623 2      to the current control block.
: 700      1624 2      -
: 701      1625 2      BASS$CB_PUSH (.CHANNEL, LUB$K_LUN_MIN);
: 702      1626 2      +
: 703      1627 2      - Arrange to POP the CCB if we get an error.
: 704      1628 2
: 705      1629 2      UNWIND_CCB = .CCB;
: 706      1630 2      UNWIND_ACTION = UNWIND_POP;
: 707      1631 2      +
: 708      1632 2      - If the LUN is already open, close it. The call to BASS$STOP IO here
: 709      1633 2      will print the former file name (that is, the name of the file that
: 710      1634 2      could not be closed).
: 711      1635 2
: 712      1636 2
```



```

713 1637 3 IF (.CCB [LUB$V_OPENED])
714 1638 2 THEN
715 1639 2 BEGIN
716 1640 2
717 1641 2 IF ( NOT OTS$$CLOSE_FILE () ) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
718 1642 2
719 1643 2 !+
720 1644 2 We must drop R11 and pick it back up so that the CLOSE can complete.
721 1645 2 Otherwise the CLOSE will 'hang fire' until the recursive I/O is
722 1646 2 done.
723 1647 2 !-
724 1648 2 BAS$$CB_POP ();
725 1649 2 BAS$$CB_PUSH (.CHANNEL, LUB$K_LUN_MIN);
726 1650 2 !+
727 1651 2 update which CCB to unwind in case of future errors
728 1652 2 !-
729 1653 2 UNWIND_CCB = .CCB;
730 1654 2 END;
731 1655 2
732 1656 2 !+
733 1657 2 If the channel is still open (possibly due to recursive I/O), fail.
734 1658 2 Also fail if there is a FAB associated with the LUB, which means that
735 1659 2 the FORTRAN compatability routines have been called to provide implicit
736 1660 2 inputs to OPEN. Again, the call to BAS$$STOP_IO will print the former
737 1661 2 file name.
738 1662 2 !-
739 1663 2
740 1664 3 IF (.CCB [LUB$V_OPENED] OR .CCB [LUB$V_DEALLOC] OR (.CCB [LUB$A_FAB] NEQA 0))
741 1665 2 THEN
742 1666 2 BAS$$STOP_IO (BAS$K_IO_CHAALR);
743 1667 2
744 1668 2 !+
745 1669 2 Make sure the file name fields are set up, in case of an error.
746 1670 2 From here to the RMS $OPEN, a call to BAS$$STOP_IO will print the name
747 1671 2 of this file, as supplied in the call.
748 1672 2 !-
749 1673 2 FILE_NAME_DESC = .OPEN_ARG_BLK [OPN$T_FILE_SPEC];
750 1674 2 CCB [LUB$A_RSN] = .FILE_NAME_DESC [DSC$A_POINTER];
751 1675 2 CCB [LUB$B_RSL] = MIN (.FILE_NAME_DESC [DSC$W_LENGTH], 255);
752 1676 2 !+
753 1677 2 Sign out the logical unit number, so an AST won't try to open it.
754 1678 2 !-
755 1679 2
756 1680 2 IF ( NOT OTS$$TAKE_LUN (CHANNEL)) THEN BAS$$STOP_IO (BAS$K_IO_CHAALR);
757 1681 2
758 1682 2 !+
759 1683 2 Now that we are sure the CCB is ours, if we get an error, deallocate it.
760 1684 2 !-
761 1685 2 UNWIND_ACTION = UNWIND_DEALLOC;
762 1686 2 !<BLF/PAGE>
```



```

764 1687 2 !+
765 1688 2 !- Set up the FAB. The fields are set up in alphabetical order.
766 1689 2 !-
767 1690 2 FAB = FAB_BLOCK;
768 1691 2 $FAB_INIT (FAB = .FAB, FAC = );
769 1692 2 FAB [FAB$L_ALQ] = .OPEN_ARG_BLK [OPN$L_FILESIZE]; ! length of file in blocks
770 1693 2
771 1694 2 IF ((.OPEN_ARG_BLK [OPN$B_ORG] EQL OPN$K_ORG_RELAT) OR (.OPEN_ARG_BLK [OPN$B_ORG] EQL OPN$K_ORG_INDEX))
772 1695 2 THEN
773 1696 2 FAB [FAB$B_BKS] = MIN (255, .BKS); ! number of blocks in each bucket
774 1697 2
775 1698 2 IF ( NOT ((.OPEN_ARG_BLK [OPN$B_ORG] EQL OPN$K_ORG_RELAT) !
776 1699 2 OR (.OPEN_ARG_BLK [OPN$B_ORG] EQL OPN$K_ORG_INDEX)))
777 1700 2 THEN
778 1701 2 FAB [FAB$W_BLS] = .BLS; ! mag tape block size
779 1702 2
780 1703 2 FAB [FAB$W_DEQ] = .OPEN_ARG_BLK [OPN$W_EXTENDSIZ]; ! blocks to add when extending file
781 1704 2 !+
782 1705 2 !- Store pointer to default file name, as specified by the user.
783 1706 2 !-
784 1707 2 FILE_NAME_DESC = .OPEN_ARG_BLK [OPN$T_DEFAULTNA];
785 1708 2
786 1709 2 IF (.FILE_NAME_DESC NEQ 0)
787 1710 2 THEN
788 1711 2 BEGIN
789 1712 2 FAB [FAB$L_DNA] = .FILE_NAME_DESC [DSC$A_POINTER];
790 1713 2 FAB [FAB$B_DNS] = MIN (255, .FILE_NAME_DESC [DSC$W_LENGTH]);
791 1714 2 END;
792 1715 2
793 1716 2 !+
794 1717 2 !- Set the FAC field.
795 1718 2 !-
796 1719 2
797 1720 2 SELECT (.OPEN_ARG_BLK [OPN$B_ACCESS]) OF
798 1721 2 SET
799 1722 2
800 1723 2 [OPN$K_ACC_DEFAU, OPN$K_ACC_MODIF] :
801 1724 2 FAB [FAB$V_DEL] = 1; ! allow deletion of records
802 1725 2
803 1726 2 [OPN$K_ACC_DEFAU, OPN$K_ACC_READ, OPN$K_ACC_MODIF, OPN$K_ACC_SCRAT] :
804 1727 2 FAB [FAB$V_GET] = 1; ! allow reading records
805 1728 2
806 1729 2 [OPN$K_ACC_DEFAU, OPN$K_ACC_WRITE, OPN$K_ACC_MODIF, OPN$K_ACC_SCRAT, OPN$K_ACC_APPEN] :
807 1730 2 FAB [FAB$V_PUT] = 1; ! allow writing records
808 1731 2
809 1732 2 [OPN$K_ACC_SCRAT] :
810 1733 2 FAB [FAB$V_TRN] = 1; ! allow truncate (scratch)
811 1734 2
812 1735 2 [OPN$K_ACC_DEFAU, OPN$K_ACC_MODIF, OPN$K_ACC_SCRAT] :
813 1736 2 FAB [FAB$V_UPD] = 1; ! allow updating records
814 1737 2
815 1738 2 [OTHERWISE] :
816 1739 2 BASS$STOP_IO (BASS$K_PROLOSSOR);
817 1740 2 TES;
818 1741 2
819 1742 2 !+
820 1743 2 !- Store pointer to file name, as specified by the user.
```



```

821 1744 2  !-
822 1745 2  FILE_NAME_DESC = .OPEN_ARG_BLK [OPN$T_FILE_SPEC];
823 1746 2  FAB [FAB$C_FNA] = .FILE_NAME_DESC [DSC$A_POINTER];
824 1747 2  FAB [FAB$B_FNS] = MIN (255, .FILE_NAME_DESC [DSC$W_LENGTH]);
825 1748 2  !+
826 1749 2  Set up the FOP field.
827 1750 2  !-
828 1751 2  !+
829 1752 2  If the user specified a file size but did not say CONTIGUOUS, do a
830 1753 2  "best effort" attempt to get contiguous space. This is compatable
831 1754 2  with FORTRAN.
832 1755 2  Note: this code has been disabled pending review. Because a search
833 1756 2  is made for the requested space, this could cause a performance
834 1757 2  problem.
835 1758 2  !-
836 1759 2  % (
C 1760 2  IF ((.OPEN_ARG_BLK [OPN$L_FILESIZE] NEQ 0) AND ( NOT .OPEN_ARG_BLK [OPN$V_CONTIGUOU]))
C 1761 2  THEN
C 1762 2  FAB [FAB$V_CBT] = 1;
840 1763 2  )%
841 1764 2  !+
842 1765 2  If the user specified neither FOR INPUT nor FOR OUTPUT, then set
843 1766 2  the CIF bit, so that the $CREATE system service will use an
844 1767 2  existing file if one is present. (Otherwise it will create a new
845 1768 2  file, as usual.)
846 1769 2  !-
847 1770 2  !-
848 1771 2  IF ( NOT (.OPEN_ARG_BLK [OPN$V_FOR_INPUT] OR .OPEN_ARG_BLK [OPN$V_FOR_OUTPU])) THEN FAB [FAB$V_CIF] = 1;
849 1772 2  !-
850 1773 2  !+
851 1774 2  Set the 'contiguous best try' flag if the user said CONTIGUOUS.
852 1775 2  ( Perhaps we should set CTG? )
853 1776 2  !-
854 1777 2  !-
855 1778 2  IF (.OPEN_ARG_BLK [OPN$V_CONTIGUOU]) THEN FAB [FAB$V_CBT] = 1;
856 1779 2  !-
857 1780 2  !+
858 1781 2  Set "deferred write" unless the ALLOW option implies sharing which
859 1782 2  does not permit it.
860 1783 2  !-
861 1784 2  !-
862 1785 4  IF ((.OPEN_ARG_BLK [OPN$B_ALLOW] NEQ OPN$K_ALL_WRITE) AND (.OPEN_ARG_BLK [OPN$B_ALLOW] NEQ
863 1786 3  OPN$K_ALL_MODIF))
864 1787 2  THEN
865 1788 2  FAB [FAB$V_DFW] = 1;
866 1789 2  !-
867 1790 2  !+
868 1791 2  If the user did not say APPEND, set the NEF flag to prevent
869 1792 2  positioning a mag tape to end-of-file on open.
870 1793 2  !-
871 1794 2  !-
872 1795 2  IF (.OPEN_ARG_BLK [OPN$B_ACCESS] NEQ OPN$K_ACC_APPEN) THEN FAB [FAB$V_NEF] = 1;
873 1796 2  !-
874 1797 2  !+
875 1798 2  If NOREWIND is not specified, cause the tape to rewind on open.
876 1799 2  !-
877 1800 2  !-
```



```
878 1801 2 IF ( NOT .OPEN_ARG_BLK [OPN$V_NO_REWIND]) THEN FAB [FAB$V_RWO] = 1;
879 1802 2
880 1803 2 !+
881 1804 2 In terminal format files, only sequential operations are allowed.
882 1805 2 Tell RMS that we will only be doing sequential operations in case
883 1806 2 it can gain some efficiency by knowing this.
884 1807 2 !-
885 1808 2
886 1809 2 IF (.OPEN_ARG_BLK [OPN$B_ORG] EQL OPN$K_ORG_TERMI) THEN FAB [FAB$V_SQO] = 1;
887 1810 2
888 1811 2 !+
889 1812 2 If the user said 'FOR OUTPUT', set the SUP bit so that the $CREATE will
890 1813 2 supercede any existing file. Note that this will happen only if a version
891 1814 2 number is specified, since the default for the version number on $CREATE is
892 1815 2 the next higher number.
893 1816 2 !-
894 1817 2
895 1818 2 IF (.OPEN_ARG_BLK [OPN$V_FOR_OUTPU]) THEN FAB [FAB$V_SUP] = 1;
896 1819 2
897 1820 2 !+
898 1821 2 If the user said TEMPORARY, mark the file as "temporary, delete when
899 1822 2 closed".
900 1823 2 !-
901 1824 2
902 1825 2 IF (.OPEN_ARG_BLK [OPN$V_TEMPORARY]) THEN FAB [FAB$V_TMD] = 1;
903 1826 2
904 1827 2 FAB [FAB$B_FSZ] = .OPEN_ARG_BLK [OPN$B_FSZ];
905 1828 2 FAB [FAB$W_MRS] = .RSZ;
906 1829 2
907 1830 2 !+
908 1831 2 Set up ORG field.
909 1832 2 !-
910 1833 2
911 1834 2 CASE (.OPEN_ARG_BLK [OPN$B_ORG]) FROM OPN$K_ORG_TERMI TO OPN$K_ORG_UNDEF OF
912 1835 2 SET
913 1836 2
914 1837 2 [OPN$K_ORG_TERMI] : ! Terminal format file
915 1838 2 FAB [FAB$B_ORG] = FAB$C_SEQ;
916 1839 2
917 1840 2 [OPN$K_ORG_VIRTU] : ! Virtual array file
918 1841 2 FAB [FAB$B_ORG] = FAB$C_SEQ;
919 1842 2
920 1843 2 [OPN$K_ORG_SEQUE] : ! Sequential file
921 1844 2 FAB [FAB$B_ORG] = FAB$C_SEQ;
922 1845 2
923 1846 2 [OPN$K_ORG_RELAT] : ! Relative file
924 1847 2 FAB [FAB$B_ORG] = FAB$C_REL;
925 1848 2
926 1849 2 [OPN$K_ORG_INDEX] : ! Indexed file
927 1850 2 FAB [FAB$B_ORG] = FAB$C_IDX;
928 1851 2
929 1852 2 [OPN$K_ORG_UNDEF] : ! Unspecified organization
930 1853 2 FAB [FAB$B_ORG] = FAB$C_SEQ;
931 1854 2
932 1855 2 [OUTRANGE] :
933 1856 2 BAS$$STOP_IO (BAS$K_PROLOSSOR);
934 1857 2 TES;
```



```

935 1858 2
936 1859 2
937 1860 2
938 1861 2
939 1862 2
940 1863 2
941 1864 2
942 1865 2
943 1866 2
944 1867 2
945 1868 2
946 1869 2
947 1870 2
948 1871 2
949 1872 2
950 1873 2
951 1874 2
952 1875 2
953 1876 2
954 1877 2
955 1878 2
956 1879 2
957 1880 2
958 1881 2
959 1882 2
960 1883 2
961 1884 2
962 1885 2
963 1886 2
964 1887 2
965 1888 2
966 1889 2
967 1890 2
968 1891 2
969 1892 2
970 1893 2
971 1894 2
972 1895 2
973 1896 2
974 1897 2
975 1898 2
976 1899 2
977 1900 2
978 1901 2
979 1902 2
980 1903 2
981 1904 2
982 1905 2
983 1906 2
984 1907 2
985 1908 2
986 1909 2
987 1910 2
988 1911 2
989 1912 2
990 1913 2
991 1914 2

!+
!-
Set up RAT field. PRN will be set after OPEN if this is a terminal device and a
terminal format file with default record attributes.

IF (.OPEN_ARG_BLK [OPNSV_NOSPAN]) THEN FAB [FAB$V_BLK] = 1;
CASE .OPEN_ARG_BLK [OPNSB_RAT] FROM OPNSK_RAT_DEFAU TO OPNSK_RAT_ANY OF
SET
  [OPNSK_RAT_DEFAU, OPNSK_RAT_ANY] :
    IF (.OPEN_ARG_BLK [OPNSB_ORG] NEQ OPNSK_ORG_VIRTU) THEN FAB [FAB$V_CR] = 1;
  [OPNSK_RAT_FORTR] :
    FAB [FAB$V_FTN] = 1;
  [OPNSK_RAT_CRLF] :
    FAB [FAB$V_CR] = 1;
  [OPNSK_RAT_NONE] :
    BEGIN
    0
    END;
  [OPNSK_RAT_PRINT] :
    FAB [FAB$V_PRN] = 1;
  [OUTRANGE] :
    BAS$$STOP_IO (BAS$K_PROLOSSOR);
TES;

!+
!-
Set up the RFM field.

CASE (.OPEN_ARG_BLK [OPNSB_RFM]) FROM OPNSK_RFM_DEFAU TO OPNSK_RFM_STREA OF
SET
  [OPNSK_RFM_DEFAU] : ! Default to variable unless VIRTUAL
    FAB [FAB$B_RFM] =
    BEGIN
      IF (.OPEN_ARG_BLK [OPNSB_ORG] EQL OPNSK_ORG_VIRTU) THEN FAB$C_FIX ELSE FAB$C_VAR
    END;
  [OPNSK_RFM_FIXED] :
    FAB [FAB$B_RFM] = FAB$C_FIX; ! fixed-length record format
  [OPNSK_RFM_VARIA] :
    FAB [FAB$B_RFM] = FAB$C_VAR; ! variable-length record format
  [OPNSK_RFM_VFC] :
    FAB [FAB$B_RFM] = FAB$C_VFC; ! variable-length with fixed control record format
```



```

: 992      1915 2      [OPN$K_RFM_STREA, OUTRANGE] :
: 993      1916 2      BASS$STOP_IO (BASS$K_PROLOSSOR);
: 994      1917 2      TES;
: 995      1918 2
: 996      1919 2      FAB [FAB$B_RTV] = .OPEN_ARG_BLK [OPN$B_WINDOWSIZE];
: 997      1920 2
: 998      1921 2      !+ Set the SHR field. Note that no bits are set in the default case, which causes
: 999      1922 2      RMS to default sharing based on how the file is to be used: read sharing if we
1000      1923 2      are only reading the file, no sharing otherwise.
1001      1924 2      !-
1002      1925 2
1003      1926 2      SELECT (.OPEN_ARG_BLK [OPN$B_ALLOW]) OF
1004      1927 2      SET
1005      1928 2
1006      1929 2      [OPN$K_ALL_MODIF] :
1007      1930 2      FAB [FAB$V_SHRDEL] = 1;          ! allow sharers to delete
1008      1931 2
1009      1932 2      [OPN$K_ALL_READ, OPN$K_ALL_MODIF] :
1010      1933 2      FAB [FAB$V_SHRGET] = 1;          ! allow sharers to read
1011      1934 2
1012      1935 2      [OPN$K_ALL_NONE, OPN$K_ALL_SCRAT] :
1013      1936 2      FAB [FAB$V_NIL] = 1;          ! forbid sharing
1014      1937 2
1015      1938 2      [OPN$K_ALL_MODIF, OPN$K_ALL_WRITE] :
1016      1939 2      FAB [FAB$V_SHRPUT] = 1;          ! allow sharers to write
1017      1940 2
1018      1941 2      [OPN$K_ALL_MODIF] :
1019      1942 2      FAB [FAB$V_SHRUPD] = 1;          ! allow sharers to update
1020      1943 2
1021      1944 2      [OPN$K_ALL_DEFAU] :
1022      1945 2      BEGIN
1023      1946 2      0
1024      1947 2      END;
1025      1948 2
1026      1949 2      [OTHERWISE] :
1027      1950 2      BASS$STOP_IO (BASS$K_PROLOSSOR);
1028      1951 2      TES;
1029      1952 2
1030      1953 2      !+
1031      1954 2      ! If organization is indexed then set FAB$V_MSE unconditionally.
1032      1955 2      !-
1033      1956 2      IF .OPEN_ARG_BLK [OPN$B_ORG] EQL OPN$K_ORG_INDEX THEN FAB [FAB$V_MSE] = 1;
1034      1957 2
1035      1958 2      !<BLF/PAGE>
```



```
1037 1959 2 1+
1038 1960 2 1- Set up the LUB.
1039 1961 2 1-
1040 1962 2 1+ CCB [LUB$A_FAB] = .FAB; ! store pointer to FAB in LUB
1041 1963 2 1-
1042 1964 2 1+ Store pointer to user-supplied file name in LUB for error processing
1043 1965 2 1-
1044 1966 2 1+ CCB [LUB$A_RSN] = .FAB [FAB$L_FNA];
1045 1967 2 1+ CCB [LUB$B_RSL] = .FAB [FAB$B_FNS];
1046 1968 2 1+
1047 1969 2 1- Set the READ ONLY, SCRATCH and APPEND bits in the LUB based on the
1048 1970 2 1- access. If READ ONLY is set, set OLD FILE, since there is no point
1049 1971 2 1- in creating a file which can only be read.
1050 1972 2 1-
1051 1973 2 1-
1052 1974 2 1- CASE (.OPEN_ARG_BLK [OPN$B_ACCESS]) FROM OPN$K_ACC_DEFAU TO OPN$K_ACC_APPEN OF
1053 1975 2 1- SET
1054 1976 2 1-
1055 1977 2 1- [OPN$K_ACC_DEFAU, OPN$K_ACC_MODIF, OPN$K_ACC_WRITE] :
1056 1978 2 1- BEGIN ! set none of the three bits
1057 1979 2 1- CCB [LUB$V_READ_ONLY] = 0;
1058 1980 2 1- CCB [LUB$V_SCRATCH] = 0;
1059 1981 2 1- CCB [LUB$V_APPEND] = 0;
1060 1982 2 1- END;
1061 1983 2 1-
1062 1984 2 1- [OPN$K_ACC_READ] :
1063 1985 2 1- BEGIN ! set READ_ONLY
1064 1986 2 1- CCB [LUB$V_OLD_FILE] = 1; ! Do not create file
1065 1987 2 1- CCB [LUB$V_READ_ONLY] = 1;
1066 1988 2 1- CCB [LUB$V_SCRATCH] = 0;
1067 1989 2 1- CCB [LUB$V_APPEND] = 0;
1068 1990 2 1- END;
1069 1991 2 1-
1070 1992 2 1- [OPN$K_ACC_SCRAT] :
1071 1993 2 1- BEGIN ! set SCRATCH
1072 1994 2 1- CCB [LUB$V_READ_ONLY] = 0;
1073 1995 2 1- CCB [LUB$V_SCRATCH] = 1;
1074 1996 2 1- CCB [LUB$V_APPEND] = 0;
1075 1997 2 1- END;
1076 1998 2 1-
1077 1999 2 1- [OPN$K_ACC_APPEN] :
1078 2000 2 1- BEGIN ! set APPEND. It will be cleared if the file is NEW
1079 2001 2 1- CCB [LUB$V_READ_ONLY] = 0;
1080 2002 2 1- CCB [LUB$V_SCRATCH] = 0;
1081 2003 2 1- CCB [LUB$V_APPEND] = 1;
1082 2004 2 1- END;
1083 2005 2 1- TES;
1084 2006 2 1-
1085 2007 2 1+
1086 2008 2 1- Set the "not sequential" flag in the LUB based on the file organization.
1087 2009 2 1-
1088 2010 2 1-
1089 2011 2 1- CASE (.OPEN_ARG_BLK [OPN$B_ORG]) FROM OPN$K_ORG_TERM1 TO OPN$K_ORG_UNDEF OF
1090 2012 2 1- SET
1091 2013 2 1-
1092 2014 2 1- [OPN$K_ORG_TERM1, OPN$K_ORG_SEQUE] :
1093 2015 2 1- CCB [LUB$V_NOTSEQORG] = 0;
```



```

: 1094      2016      2
: 1095      2017      2
: 1096      2018      2      [OPNSK_ORG_RELAT, OPNSK_ORG_INDEX, OPNSK_ORG_VIRTU, OPNSK_ORG_UNDEF] :
: 1097      2019      2      CCB [LUB$V_NOTSEQORG] = -1;
: 1098      2020      2      TES;
: 1099      2021      2
: 1100      2022      2      !+ Set the 'formatted' and 'unformatted' bits in the LUB.
: 1101      2023      2      These bits are used only by FORTRAN I/O statements.
: 1102      2024      2      !-
: 1103      2025      2
: 1104      2026      2      CASE (.OPEN_ARG_BLK [OPNSB_ORG]) FROM OPNSK_ORG_TERM1 TO OPNSK_ORG_UNDEF OF
: 1105      2027      2      SET
: 1106      2028      2
: 1107      2029      2      [OPNSK_ORG_TERM1, OPNSK_ORG_SEQUE, OPNSK_ORG_RELAT, OPNSK_ORG_INDEX] :
: 1108      2030      2      CCB [LUB$V_FORMATTED] = -1;
: 1109      2031      2
: 1110      2032      2      [OPNSK_ORG_VIRTU, OPNSK_ORG_UNDEF] :
: 1111      2033      2      CCB [LUB$V_UNFORMAT] = T;
: 1112      2034      2      TES;
: 1113      2035      2
: 1114      2036      2      !+
: 1115      2037      2      Set the 'fixed' flag in the LUB.
: 1116      2038      2      !-
: 1117      2039      2
: 1118      2040      2      IF (.FAB [FAB$B_RFM] EQL FAB$C_FIX) THEN CCB [LUB$V_FIXED] = 1;
: 1119      2041      2
: 1120      2042      2      !+
: 1121      2043      2      Set the 'old file' flag in the LUB.
: 1122      2044      2      It will also be set after the $OPEN if the file turns out to already
: 1123      2045      2      exist.
: 1124      2046      2      !-
: 1125      2047      2
: 1126      2048      2      IF (.OPEN_ARG_BLK [OPNSV_FOR_INPUT]) THEN CCB [LUB$V_OLD_FILE] = 1;
: 1127      2049      2
: 1128      2050      2      !+
: 1129      2051      2      Set the 'direct' flag in the LUB. This is used only by FORTRAN I/O.
: 1130      2052      2      !-
: 1131      2053      2
: 1132      2054      2      CASE (.OPEN_ARG_BLK [OPNSB_ORG]) FROM OPNSK_ORG_TERM1 TO OPNSK_ORG_UNDEF OF
: 1133      2055      2      SET
: 1134      2056      2
: 1135      2057      2      [OPNSK_ORG_TERM1, OPNSK_ORG_SEQUE, OPNSK_ORG_RELAT, OPNSK_ORG_INDEX] :
: 1136      2058      2      CCB [LUB$V_DIRECT] = 0;          ! not direct access
: 1137      2059      2
: 1138      2060      2      [OPNSK_ORG_VIRTU, OPNSK_ORG_UNDEF] :
: 1139      2061      2      CCB [LUB$V_DIRECT] = 1;          ! direct access. Assigning UNDEF here is arbitrary.
: 1140      2062      2      TES;
: 1141      2063      2
: 1142      2064      2      !+
: 1143      2065      2      Set up the right margin and default right margin.
: 1144      2066      2      !-
: 1145      2067      2
: 1146      2068      2      CASE (.OPEN_ARG_BLK [OPNSB_ORG]) FROM OPNSK_ORG_TERM1 TO OPNSK_ORG_UNDEF OF
: 1147      2069      2      SET
: 1148      2070      2
: 1149      2071      2      [OPNSK_ORG_TERM1] :
: 1150      2072      2      BEGIN
```



```
1151 2073 3 !+
1152 2074 3 !- Terminal format file. We assume that this will be a disk file. If it turns out
1153 2075 3 to be a terminal, the margin will be reset to "infinite" after the OPEN.
1154 2076 3 !-
1155 2077 3 CCB [LUB$W_D_MARGIN] =
1156 2078 4 BEGIN
1157 2079 4
1158 2080 5 IF (.OPEN_ARG_BLK [OPN$W_RECORDSIZE] NEQ 0)
1159 2081 4 THEN
1160 2082 4 .OPEN_ARG_BLK [OPN$W_RECORDSIZE]
1161 2083 4 ELSE
1162 2084 4 !+
1163 2085 4 for V2 programs, use the MAP size if it was specified.
1164 2086 4 (and there was no recordsize)
1165 2087 4 !-
1166 2088 5 IF ( (.OPEN_ARG_BLK [OPN$B_CNT] GTR K V1_BLK_SIZE) AND
1167 2089 5 (.OPEN_ARG_BLK [OPN$W_MAP_SIZE] NEQ 0) )
1168 2090 4 THEN .OPEN_ARG_BLK [OPN$W_MAP_SIZE]
1169 2091 4 ELSE LUB$K_D_MARGIN
1170 2092 4
1171 2093 3 END;
1172 2094 3 CCB [LUB$W_R_MARGIN] = .CCB [LUB$W_D_MARGIN];
1173 2095 3 CCB [LUB$V_NOMARGIN] = 0;
1174 2096 2 END;
1175 2097 2
1176 2098 2 [INRANGE] :
1177 2099 2 BEGIN
1178 2100 3 !+
1179 2101 3 This is not a terminal format file. The margin is only important for sequential files,
1180 2102 3 so we will set up the margin for sequential files.
1181 2103 3 !-
1182 2104 3 CCB [LUB$W_D_MARGIN] =
1183 2105 4 BEGIN
1184 2106 4
1185 2107 5 IF (.OPEN_ARG_BLK [OPN$W_RECORDSIZE] NEQ 0)
1186 2108 4 THEN
1187 2109 4 .OPEN_ARG_BLK [OPN$W_RECORDSIZE]
1188 2110 4 ELSE
1189 2111 4 !+
1190 2112 4 for V2 programs, use the MAP size if it was specified.
1191 2113 4 (and there was no recordsize)
1192 2114 4 !-
1193 2115 5 IF ( (.OPEN_ARG_BLK [OPN$B_CNT] GTR K V1_BLK_SIZE) AND
1194 2116 5 (.OPEN_ARG_BLK [OPN$W_MAP_SIZE] NEQ 0) )
1195 2117 4 THEN .OPEN_ARG_BLK [OPN$W_MAP_SIZE]
1196 2118 4 ELSE LUB$K_D_MARGIN
1197 2119 4
1198 2120 3 END;
1199 2121 3 CCB [LUB$W_R_MARGIN] = 0;
1200 2122 3 CCB [LUB$V_NOMARGIN] = 1;
1201 2123 2 END;
1202 2124 2
1203 2125 2 [OUTRANGE] :
1204 2126 2 BAS$$STOP_IO (BAS$K_PROLOSSOR);
1205 2127 2 TES;
1206 2128 2
1207 2129 2 !+
```


BAS\$OPEN
1-113

M 2
16-Sep-1984 00:52:31
14-Sep-1984 11:55:24

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASOPEN.B32;91

Page 24
(6)

```
: 1208      2130 2 ! Set a flag in the LUB if this file was opened with keys. This
: 1209      2131 2 ! is important for processing indexed files.
: 1210      2132 2 !-
: 1211      2133 2 CCB [LUB$V_KEYED] = (ACTUALCOUNT () GEQ 2);
: 1212      2134 2 !+
: 1213      2135 2 ! Set a flag in the LUB if this is a multi-stream connect.
: 1214      2136 2 !-
: 1215      2137 2
: 1216      2138 3 IF (.OPEN_ARG_BLK [OPN$W_CONNECT] NEQ 0)
: 1217      2139 2 THEN
: 1218      2140 2 CCB [LUB$V_M_STREAM] = 1
: 1219      2141 2 ELSE
: 1220      2142 2 CCB [LUB$V_M_STREAM] = 0;
: 1221      2143 2
: 1222      2144 2 !+
: 1223      2145 2 ! Set the terminal format file bit in the LUB if this is a terminal
: 1224      2146 2 ! format file. This is used to bias RECOUNT.
: 1225      2147 2 !-
: 1226      2148 2
: 1227      2149 2 IF (.OPEN_ARG_BLK [OPN$B_ORG] EQL OPN$K_ORG_TERMI) THEN CCB [LUB$V_TERM_FOR] = 1;
: 1228      2150 2
: 1229      2151 2 !<BLF/PAGE>
```



```
1231 2152 2  !+
1232 2153 2  ! Set up the RAB.
1233 2154 2  !-
1234 2155 2  CCB [RAB$B_BID] = RAB$C_BID;          ! this is a RAB
1235 2156 2  CCB [RAB$B_BLN] = RAB$C_BLN;          ! length of a RAB
1236 2157 2  CCB [RAB$L_FAB] = .FAB;                ! pointer to File Access Block
1237 2158 2  !+
1238 2159 2  ! Set up the key buffer and size fields. If this is a terminal device
1239 2160 2  ! these will be overwritten by the prompt buffer and size fields,
1240 2161 2  ! but this is OK, since you cannot do keyed access to a terminal device.
1241 2162 2  !-
1242 2163 2  CCB [RAB$L_KBF] = CCB [LUB$L_LOG_RECNO];
1243 2164 2  CCB [RAB$B_KSZ] = 4;
1244 2165 2  CCB [RAB$B_MBF] = .OPEN_ARG_BLK [OPN$B_MULTIBUFF]; ! number of buffers
1245 2166 2
1246 2167 2  IF (.OPEN_ARG_BLK [OPN$B_FSZ] NEQ 0) THEN CCB [RAB$L_RHB] = .OPEN_ARG_BLK [OPN$A_VFC];
1247 2168 2
1248 2169 2  !+
1249 2170 2  ! Set up the ROP field.
1250 2171 2  !-
1251 2172 2
1252 2173 2  !+
1253 2174 2  ! for V2 programs, enable manual record locking if specified.
1254 2175 2  !-
1255 2176 3  IF (.OPEN_ARG_BLK [OPN$B_CNT] GTR K_V1_BLK_SIZE)
1256 2177 3  THEN
1257 2178 3  IF (.OPEN_ARG_BLK [OPN$V_UNLOCK] NEQ 0)
1258 2179 3  THEN
1259 2180 3  BEGIN
1260 2181 3
1261 2182 3  !+
1262 2183 3  ! if this is a sequential file, and the recordsize is not 512,
1263 2184 3  ! manual record locking is not allowed; otherwise, it is.
1264 2185 3  !-
1265 2186 4  IF (.OPEN_ARG_BLK [OPN$B_ORG] EQL OPN$K_ORG_SEQUE AND
1266 2187 4  .OPEN_ARG_BLK [OPN$W_RECORDSIZ] NEQ 512)
1267 2188 3  THEN BAS$$STOP_IO (BAS$K_REQRECSIZ)
1268 2189 3
1269 2190 3  ELSE CCB [RAB$V_ULK] = 1;
1270 2191 3  ! enable manual record locking
1271 2192 2  END;
1272 2193 2
1273 2194 2  !+
1274 2195 2  ! If we are to append to the file, position to EOF when opening it.
1275 2196 2  !-
1276 2197 2
1277 2198 2  IF (.OPEN_ARG_BLK [OPN$B_ACCESS] EQL OPN$K_ACC_APPEN) THEN CCB [RAB$V_EOF] = 1;
1278 2199 2
1279 2200 2  !+
1280 2201 2  ! Always turn off Locate mode. This is done because GETs of variable length
1281 2202 2  ! records are to null fill the remainder of the buffer. This cannot be done
1282 2203 2  ! if the data is in an RMS buffer in system space.
1283 2204 2  !-
1284 2205 2  CCB [RAB$V_LOC] = 0;
1285 2206 2  !+
1286 2207 2  ! Don't truncate the file on a PUT to its middle; give an error message
1287 2208 2  ! instead.
```



```

1288 2209 2 !-
1289 2210 2 CCB [RAB$V_TPT] = 0;
1290 2211 2
1291 2212 2 !+ Don't convert PUT to UPDATE except on VIRTUAL files. If the user wants
1292 2213 2 to do an UPDATE to any other kind of file, he must issue an explicit
1293 2214 2 UPDATE statement.
1294 2215 2 !-
1295 2216 2 CCB [RAB$V_UIF] = (.OPEN_ARG_BLK [OPNSB_ORG] EQL OPNSK_ORG_VIRTU);
1296 2217 2 !+
1297 2218 2 Initialize the local NAM block and point the FAB to it so that
1298 2219 2 RMS will return the full name of the file being accessed. This
1299 2220 2 will improve error messages.
1300 2221 2 !-
1301 2222 2 $NAM_INIT (NAM = NAM_BLOCK);
1302 2223 2 NAM_BLOCK [NAM$S_RSA] = NAM_BLOCK [NAM$S_ESA] = FILE_NAME;
1303 2224 2 NAM_BLOCK [NAM$S_RSS] = NAM_BLOCK [NAM$S_ESS] = NAM$C_MAXRSS;
1304 2225 2 FAB [FAB$S_NAM] = NAM_BLOCK;
1305 2226 2 !+
1306 2227 2 Initialize the local FHC XAB and point the FAB to it. This is
1307 2228 2 used to get the size of the longest record in an old file, so
1308 2229 2 that the buffer size can be defaulted properly.
1309 2230 2 Also initialize the SUMMARY XAB and add it into the XAB chain.
1310 2231 2 This will be used to set total number of keys for indexed files.
1311 2232 2 !-
1312 2233 2 $XABFHC_INIT (XAB = XABFHC, NXT = XABSUM);
1313 2234 2 $XABSUM_INIT (XAB = XABSUM);
1314 2235 2 XABSUM [XAB$S_NXT] = 0; ! make sure SUM is end of chain
1315 2236 2 FAB [FAB$S_XAB] = XABFHC; ! Link FAB to XAB
1316 2237 2 !+
1317 2238 2 Initialize the KEY XABs if any keys are provided in this call to OPEN.
1318 2239 2 !-
1319 2240 2
1320 2241 2 IF (ACTUALCOUNT () GEQ 2)
1321 2242 2 THEN
1322 2243 2 BEGIN
1323 2244 2
1324 2245 2 LOCAL
1325 2246 2 KEY_PTR : REF BLOCK [, BYTE] FIELD (OPNSKEY_BLOCK),
1326 2247 2 XABKEY : REF $XABKEY_DECL;
1327 2248 2
1328 2249 2 !+
1329 2250 2 The keys are arranged in order by key number.
1330 2251 2 They are allocated in reverse order because RMS-32 requires them to
1331 2252 2 be chained in forward order.
1332 2253 2 !-
1333 2254 2
1334 2255 2 DECR KEYNO FROM .KEY_INFO_BLK [KEYH$B_KEYNUM] - 1 TO 0 DO
1335 2256 2 BEGIN
1336 2257 2 KEY_PTR = (.KEYNO*.KEY_INFO_BLK [KEYH$B_LEN]) + KEYH$K_LENGTH + .KEY_INFO_BLK;
1337 2258 2 BEGIN
1338 2259 2
1339 2260 2 LOCAL
1340 2261 2 GET_VM_STATUS;
1341 2262 2
1342 2263 2 GET_VM_STATUS = LIB$GET_VM (%REF (XAB$C_KEYLEN), XABKEY);
1343 2264 2
1344 2265 2 IF ( NOT .GET_VM_STATUS) THEN BAS$$STOP_IO (BAS$K_MAXMEMEXC);
```



```
: 1345      2266  5
: 1346      2267  4      END;
: 1347      2268  4      $XABKEY_INIT (XAB = .XABKEY);
: 1348      2269  4      !+
: 1349      2270  4      ! Fill in the key XAB from the parameter.
: 1350      2271  4      !-
: 1351      2272  4      XABKEY [XAB$B_REF] = .KEYNO;          ! Key of reference
: 1352      2273  4      XABKEY [XAB$W_POS0] = .KEY_PTR [KEY$W_OFFSET]; ! Position in record
: 1353      2274  4      XABKEY [XAB$B_SIZE0] = .KEY_PTR [KEY$B_LEN];  ! Total key length
: 1354      2275  4      !                                     if NOT segmented;
: 1355      2276  4      !                                     otherwise length of
: 1356      2277  4      !                                     first segment.
: 1357      2278  4      XABKEY [XAB$V_CHG] = .KEY_PTR [KEY$V_CHG];    ! Key can be changed
: 1358      2279  4      XABKEY [XAB$V_DUP] = .KEY_PTR [KEY$V_DUP];    ! Key can be duplicated
: 1359      2280  4
: 1360      2281  4      !+
: 1361      2282  4      ! also fill in the summary XAB.
: 1362      2283  4      !-
: 1363      2284  4      XABSUM [XAB$B_NOK] = .KEY_INFO_BLK [KEYH$B_KEYNUM];
: 1364      2285  4
: 1365      2286  4      !+
: 1366      2287  4      ! The V2 compiler supports segmented keys, the V1 compiler does not. If this
: 1367      2288  4      ! is a V2 program, check for multiple key segments.
: 1368      2289  4      !-
: 1369      2290  5      IF (.OPEN_ARG_BLK [OPN$B_CNT] GTR K_V1_BLK_SIZE)
: 1370      2291  4      THEN
: 1371      2292  5      BEGIN
: 1372      2293  5      IF .KEY_PTR [KEY$B_NUM_SEG] GTR 0
: 1373      2294  5      THEN
: 1374      2295  6      BEGIN
: 1375      2296  6      INCR KEY_NUM FROM 1 TO .KEY_PTR [KEY$B_NUM_SEG] DO
: 1376      2297  7      BEGIN
: 1377      2298  7      CASE .KEY_NUM FROM 1 TO 7 OF
: 1378      2299  7      SET
: 1379      2300  7      [1]:
: 1380      2301  8      BEGIN
: 1381      2302  8      XABKEY [XAB$B_SIZE1] = .KEY_PTR [KEY$B_LEN1];
: 1382      2303  8      XABKEY [XAB$W_POS1] = .KEY_PTR [KEY$W_OFFSET1];
: 1383      2304  7      END;
: 1384      2305  7
: 1385      2306  7      [2]:
: 1386      2307  8      BEGIN
: 1387      2308  8      XABKEY [XAB$B_SIZE2] = .KEY_PTR [KEY$B_LEN2];
: 1388      2309  8      XABKEY [XAB$W_POS2] = .KEY_PTR [KEY$W_OFFSET2];
: 1389      2310  7      END;
: 1390      2311  7
: 1391      2312  7      [3]:
: 1392      2313  8      BEGIN
: 1393      2314  8      XABKEY [XAB$B_SIZE3] = .KEY_PTR [KEY$B_LEN3];
: 1394      2315  8      XABKEY [XAB$W_POS3] = .KEY_PTR [KEY$W_OFFSET3];
: 1395      2316  7      END;
: 1396      2317  7
: 1397      2318  7      [4]:
: 1398      2319  8      BEGIN
: 1399      2320  8      XABKEY [XAB$B_SIZE4] = .KEY_PTR [KEY$B_LEN4];
: 1400      2321  8      XABKEY [XAB$W_POS4] = .KEY_PTR [KEY$W_OFFSET4];
: 1401      2322  7      END;
```



```

1402      2323  7
1403      2324  7
1404      2325  8
1405      2326  8
1406      2327  8
1407      2328  7
1408      2329  7
1409      2330  7
1410      2331  8
1411      2332  8
1412      2333  8
1413      2334  7
1414      2335  7
1415      2336  7
1416      2337  8
1417      2338  8
1418      2339  8
1419      2340  7
1420      2341  7
1421      2342  7
1422      2343  7
1423      2344  7
1424      2345  7
1425      2346  6
1426      2347  5
1427      2348  4
1428      2349  4
1429      2350  4
1430      2351  4
1431      2352  4
1432      2353  4
1433      2354  4
1434      2355  4
1435      2356  5
1436      2357  5
1437      2358  5
1438      2359  5
1439      2360  5
1440      2361  5
1441      2362  5
1442      2363  5
1443      2364  5
1444      2365  5
1445      2366  5
1446      2367  5
1447      2368  5
1448      2369  5
1449      2370  5
1450      2371  5
1451      2372  5
1452      2373  5
1453      2374  5
1454      2375  5
1455      2376  5
1456      2377  5
1457      2378  5
1458      2379  5

[5]:
BEGIN
XABKEY [XAB$B_SIZ5] = .KEY_PTR [KEY$B_LEN5];
XABKEY [XAB$W_POS5] = .KEY_PTR [KEY$W_OFFSET5];
END;

[6]:
BEGIN
XABKEY [XAB$B_SIZ6] = .KEY_PTR [KEY$B_LEN6];
XABKEY [XAB$W_POS6] = .KEY_PTR [KEY$W_OFFSET6];
END;

[7]:
BEGIN
XABKEY [XAB$B_SIZ7] = .KEY_PTR [KEY$B_LEN7];
XABKEY [XAB$W_POS7] = .KEY_PTR [KEY$W_OFFSET7];
END;

[OUTRANGE]:
BAS$$STOP_IO (BAS$K_NOTIMP);

TES;
END;
END;

!+ Fill in the data type field. The compiler deals in VAX standard
!- codes for the data types, so we must translate them to the RMS
!- codes. VAX codes which do not translate are treated as strings.

XABKEY [XAB$B_DTP] =
BEGIN
CASE .KEY_PTR [KEY$B_DTYPE] FROM DSC$K_DTYPE_WU TO DSC$K_DTYPE_P OF
SET
[DSC$K_DTYPE_WU] :
XAB$C_BN2;
[DSC$K_DTYPE_LU] :
XAB$C_BN4;
[DSC$K_DTYPE_W] :
XAB$C_IN2;
[DSC$K_DTYPE_L] :
XAB$C_IN4;
[DSC$K_DTYPE_T] :
XAB$C_STG;
[DSC$K_DTYPE_P] :
XAB$C_PAT;
[INRANGE, OUTRANGE] :
```



```
: 1459      2380  5      XAB$C_STG;
: 1460      2381  5      TES
: 1461      2382  5
: 1462      2383  4      END;
: 1463      2384  4      !+
: 1464      2385  4      ! define the total key size of this key XAB, since RMS
: 1465      2386  4      ! doesn't set it for us in the case of a $CREATE.
: 1466      2387  4      !-
: 1467      2388  5      XABKEY [XAB$B_TKS] = (IF .KEY_PTR [KEY$B_NUM_SEG] EQL 0
: 1468      2389  5      THEN
: 1469      2390  5      .XABKEY [XAB$B_SIZ0]
: 1470      2391  5      ELSE
: 1471      2392  5      .XABKEY [XAB$B_SIZ0] + .XABKEY [XAB$B_SIZ1] +
: 1472      2393  5      .XABKEY [XAB$B_SIZ2] + .XABKEY [XAB$B_SIZ3] +
: 1473      2394  5      .XABKEY [XAB$B_SIZ4] + .XABKEY [XAB$B_SIZ5] +
: 1474      2395  4      .XABKEY [XAB$B_SIZ6] + .XABKEY [XAB$B_SIZ7]);
: 1475      2396  4
: 1476      2397  4      !+
: 1477      2398  4      ! Link this key XAB to the FAB.
: 1478      2399  4      ! The XAB chain is set up so that key XABs are placed, in order,
: 1479      2400  4      ! in front of any other existing XABs on the file.
: 1480      2401  4      !-
: 1481      2402  4      XABKEY [XAB$L_NXT] = .FAB [FAB$L_XAB];
: 1482      2403  4      FAB [FAB$L_XAB] = .XABKEY;
: 1483      2404  3      END;
: 1484      2405  3
: 1485      2406  2      END;
: 1486      2407  2
: 1487      2408  2      !<BLF/PAGE>
```



```

1489      2409  2  !+
1490      2410  2  ! Set up the FAB from the parent file if this is a multi-stream connect.
1491      2411  2  ! This is needed because we will not be doing an OPEN.
1492      2412  2  !-
1493      2413  2
1494      2414  2      IF (.CCB [LUB$V_M_STREAM])
1495      2415  2      THEN
1496      2416  2          BEGIN
1497      2417  2
1498      2418  2          LOCAL
1499      2419  2              PARENT_IFI,
1500      2420  2              PARENT_ORG,
1501      2421  2              PARENT_MRS,
1502      2422  2              PARENT_RAT,
1503      2423  2              PARENT_RFM,
1504      2424  2              PARENT_BKS,
1505      2425  2              PARENT_BLS,
1506      2426  2              CONNECTED,
1507      2427  2              OUR_CCB : REF BLOCK [, BYTE];
1508      2428  2
1509      2429  2      OUR_CCB = .CCB;
1510      2430  2      BAS$$CB PUSH (.OPEN ARG BLK [OPNSW CONNECT], LUB$K_LUN_MIN);
1511      2431  2      PARENT_IFI = (IF (.CCB [LUB$V_OPENED]) THEN .CCB [LUB$W_IFI] ELSE 0);
1512      2432  2      PARENT_ORG = .CCB [LUB$B_ORG];
1513      2433  2      PARENT_MRS = .CCB [LUB$W_RBUF_SIZE];
1514      2434  2      PARENT_RAT = .CCB [LUB$B_RAT];
1515      2435  2      PARENT_RFM = .CCB [LUB$B_RFM];
1516      2436  2      PARENT_BKS = .CCB [LUB$B_BKS];
1517      2437  2      PARENT_BLS = .CCB [LUB$W_BLS];
1518      2438  2
1519      2439  2  !+
1520      2440  2  ! Mark that there may be a connect to this file, for CLOSE.
1521      2441  2  ! If this is already connected; a child instead of a parent; then remember to
1522      2442  2  ! complain later.
1523      2443  2  !-
1524      2444  2      IF .CCB [LUB$V_M_STREAM] EQL 1
1525      2445  2      THEN
1526      2446  2          CONNECTED = 1
1527      2447  2      ELSE
1528      2448  2          BEGIN
1529      2449  2              CONNECTED = 0;
1530      2450  2              CCB [LUB$V_M_STR_C] = 1;
1531      2451  2          END;
1532      2452  2
1533      2453  2      BAS$$CB POP ();
1534      2454  2      CCB = .OUR_CCB;
1535      2455  2
1536      2456  2      IF (.CONNECTED) THEN BAS$$STOP_IO (BAS$K_INVFILOPT);
1537      2457  2
1538      2458  2      IF (.PARENT_IFI EQL 0) THEN BAS$$STOP_IO (BAS$K_IO_CHANOT);
1539      2459  2
1540      2460  2      IF (.PARENT_ORG NEQ LUB$K_ORG_INDEX) THEN BAS$$STOP_IO (BAS$K_FILATTNOT);
1541      2461  2
1542      2462  2      FAB [FAB$W_IFI] = .PARENT_IFI;
1543      2463  2      FAB [FAB$W_MRS] = .PARENT_MRS;
1544      2464  2      FAB [FAB$B_RAT] = .PARENT_RAT;
1545      2465  2      FAB [FAB$B_RFM] = .PARENT_RFM;
1546      2466  2      FAB [FAB$B_BKS] = .PARENT_BKS;
```


BAS\$OPEN
1-113

6 3
16-Sep-1984 00:52:31
14-Sep-1984 11:55:24

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASOPEN.B32;91

Page 31
(8)

```
: 1546      2466 3      FAB [FAB$W_BLS] = .PARENT_BLS;  
: 1547      2467 2      END;  
: 1548      2468 2  
: 1549      2469 2 !<BLF/PAGE>
```



```

: 1551      2470 2  !+
: 1552      2471 2  ! If the USEROPEN value is non-zero, call the user-supplied procedure
: 1553      2472 2  ! to do the $OPEN and $CONNECT. It will return an RMS status code
: 1554      2473 2  ! as its value. Otherwise we do the $OPEN and $CONNECT ourselves.
: 1555      2474 2  ! If we call USEROPEN, set a flag in the LUB to help software support
: 1556      2475 2  ! if they get an SPR.
: 1557      2476 2  !-
: 1558      2477 2  !
: 1559      2478 3      IF (.OPEN_ARG_BLK [OPN$A_USEROPEN] NEQA 0)
: 1560      2479 3      THEN
: 1561      2480 3          BEGIN
: 1562      2481 3              CCB [LUB$V_USEROPEN] = 1;
: 1563      2482 3              OPEN_STATUS = (.OPEN_ARG_BLK [OPN$A_USEROPEN]) (.FAB, .CCB, %REF (.CCB [LUB$W_LUN]));
: 1564      2483 3              CONNECT_STATUS = $$$_NORMAL;
: 1565      2484 3
: 1566      2485 4              IF ( NOT .OPEN_STATUS)
: 1567      2486 3              THEN
: 1568      2487 4                  BEGIN
: 1569      2488 4  !+
: 1570      2489 4  ! Try to construct the correct values for OPEN_STATUS and CONNECT_STATUS.
: 1571      2490 4  !-
: 1572      2491 4
: 1573      2492 5              IF (.FAB [FAB$L_STS])
: 1574      2493 4              THEN
: 1575      2494 5                  BEGIN
: 1576      2495 5  !+
: 1577      2496 5  ! The $OPEN was OK, how about the $CONNECT.
: 1578      2497 5  !-
: 1579      2498 5
: 1580      2499 6                  IF ( NOT .CCB [RAB$L_STS])
: 1581      2500 5                  THEN
: 1582      2501 6                      BEGIN
: 1583      2502 6  !+
: 1584      2503 6  ! The $OPEN succeeded but the $CONNECT failed.
: 1585      2504 6  !-
: 1586      2505 6                      OPEN_STATUS = .FAB [FAB$L_STS];
: 1587      2506 6                      CONNECT_STATUS = .CCB [RAB$L_STS];
: 1588      2507 6                      END
: 1589      2508 5                  ELSE
: 1590      2509 5  !+
: 1591      2510 5  ! Both the RMS values look ok, just signal the error.
: 1592      2511 5  !-
: 1593      2512 5                      LIB$STOP (.OPEN_STATUS);
: 1594      2513 5
: 1595      2514 4                  END;
: 1596      2515 4
: 1597      2516 3                  END;
: 1598      2517 3
: 1599      2518 3                      END
: 1600      2519 3      ELSE
: 1601      2520 3          BEGIN
: 1602      2521 3  !+
: 1603      2522 3  ! Not USEROPEN. If an old file is explicitly wanted
: 1604      2523 3  ! (user said FOR INPUT) do a $OPEN. Otherwise do a $CREATE.
: 1605      2524 3  ! However, if this is just a CONNECT, do a $DISPLAY to set up the XABs.
: 1606      2525 3  !-
: 1607      2526 3
```



```
: 1608      2527  4      IF (.CCB [LUB$V_M_STREAM])
: 1609      2528  3      THEN
: 1610      2529  4          OPEN_STATUS = $DISPLAY (FAB = .FAB)
: 1611      2530  4
: 1612      2531  3      ELSE
: 1613      2532  4          BEGIN
: 1614      2533  4
: 1615      2534  4      !+
: 1616      2535  4      ! check for a terminal format file on a terminal device,
: 1617      2536  4      ! and change to PRN format if so.
: 1618      2537  4      ! This is so that the terminal is forcible.
: 1619      2538  4      !-
: 1620      2539  4          OPEN_STATUS = $PARSE (FAB = .FAB);
: 1621      2540  4
: 1622      2541  5      IF (.OPEN_STATUS)
: 1623      2542  4      THEN
: 1624      2543  5          BEGIN
: 1625      2544  5
: 1626      2545  7          IF (((.FAB [FAB$L_DEV] AND DEV$M_TRM) NEQ 0) !
: 1627      2546  7              AND (.OPEN_ARG_BLK [OPN$B_ORG] EQL OPN$K_ORG_TERM) !
: 1628      2547  7              AND (.OPEN_ARG_BLK [OPN$B_RFM] EQL OPN$K_RFM_DEFAU) !
: 1629      2548  6              AND (.OPEN_ARG_BLK [OPN$B_RAT] EQL OPN$K_RAT_DEFAU))
: 1630      2549  5          THEN
: 1631      2550  6              BEGIN
: 1632      2551  6      !+
: 1633      2552  6      ! Turn off CR and turn on PRN.
: 1634      2553  6      !-
: 1635      2554  6          FAB [FAB$V_CR] = 0;
: 1636      2555  6          FAB [FAB$V_PRN] = 1;
: 1637      2556  6      !+
: 1638      2557  6      ! Change from VAR to VFC, so we can support PRN.
: 1639      2558  6      !-
: 1640      2559  6          FAB [FAB$B_RFM] = FAB$C_VFC;
: 1641      2560  6          FAB [FAB$B_FSZ] = 2;
: 1642      2561  6      !+
: 1643      2562  6      ! Any VFC field provided by the user is ignored.
: 1644      2563  6      !-
: 1645      2564  6          CCB [RAB$L_RHB] = CCB [LUB$W_BAS_VFC];
: 1646      2565  6      !+
: 1647      2566  6      ! make the terminal forcible.
: 1648      2567  6      !-
: 1649      2568  6          CCB [LUB$V_FORCIBLE] = 1;
: 1650      2569  5          END;
: 1651      2570  5
: 1652      2571  4      END;
: 1653      2572  4
: 1654      2573  5      IF (.CCB [LUB$V_OLD_FILE])
: 1655      2574  4      THEN
: 1656      2575  5          OPEN_STATUS = $OPEN (FAB = .FAB)
: 1657      2576  4      ELSE
: 1658      2577  4          OPEN_STATUS = $CREATE (FAB = .FAB);
: 1659      2578  3      END;
: 1660      2579  3
: 1661      2580  3      IF (.OPEN_STATUS) THEN CONNECT_STATUS = $CONNECT (RAB = .CCB);
: 1662      2581  3
: 1663      2582  2      END;
: 1664      2583  2
```



```

1665 2584 2  !+
1666 2585 2  ! If FAB$V_CIF is set, set the 'old file' bit in the LUB based on
1667 2586 2  ! whether or not an existing file was found. If this is a CONNECT, always
1668 2587 2  ! set 'old file', for error checking.
1669 2588 2  !-
1670 2589 2  !-
1671 2590 3  IF (.CCB [LUB$V_M_STREAM] OR (.FAB [FAB$V_CIF] AND (.FAB [FAB$L_STS] NEQU RMS$_CREATED)))
1672 2591 2  THEN
1673 2592 2  CCB [LUB$V_OLD_FILE] = 1;
1674 2593 2  !-
1675 2594 2  !+
1676 2595 2  ! Store away the Directory ID in case CLOSE needs to delete the file.
1677 2596 2  ! Also save the IFI.
1678 2597 2  !-
1679 2598 2  CH$MOVE (NAM$_DID, NAM_BLOCK [NAM$_DID], CCB [LUB$_DID]);
1680 2599 2  CCB [LUB$_IFI] = .FAB [FAB$_IFI];
1681 2600 2  !+
1682 2601 2  ! If we have an expanded name string or a resultant name string, point
1683 2602 2  ! the LUB to it instead of the user-supplied name, to improve error
1684 2603 2  ! messages.
1685 2604 2  !-
1686 2605 2  !-
1687 2606 3  IF (.NAM_BLOCK [NAM$_RSL] NEQA 0)
1688 2607 2  THEN
1689 2608 2  BEGIN
1690 2609 2  CCB [LUB$_RSN] = .NAM_BLOCK [NAM$_RSA];
1691 2610 2  CCB [LUB$_RSL] = .NAM_BLOCK [NAM$_RSL];
1692 2611 2  END
1693 2612 2  ELSE
1694 2613 2  !-
1695 2614 3  IF (.NAM_BLOCK [NAM$_ESL] NEQA 0)
1696 2615 2  THEN
1697 2616 2  BEGIN
1698 2617 2  CCB [LUB$_RSN] = .NAM_BLOCK [NAM$_ESA];
1699 2618 2  CCB [LUB$_RSL] = .NAM_BLOCK [NAM$_ESL];
1700 2619 2  END;
1701 2620 2  !-
1702 2621 2  !+
1703 2622 2  ! From here to the end of this routine, a call to BAS$$STOP_IO prints the
1704 2623 2  ! expanded or resultant name from RMS, hence any error detection should be
1705 2624 2  ! done after this point, if this is reasonable.
1706 2625 2  !-
1707 2626 2  !+
1708 2627 2  ! If OPEN or CREATE got an error, give an appropriate error message.
1709 2628 2  !-
1710 2629 2  !-
1711 2630 2  IF ( NOT .OPEN_STATUS) THEN BAS$$STOP_IO (BAS$_IOERR_OPE);
1712 2631 2  !-
1713 2632 2  !+
1714 2633 2  ! Since the CCB has been RMS OPENed, CLOSE it if we detect an error hereafter.
1715 2634 2  ! Note that this will actually do a DISCONNECT if LUB$V_M_STREAM is set.
1716 2635 2  !-
1717 2636 2  UNWIND_ACTION = UNWIND_CLOSE;
1718 2637 2  !-
1719 2638 2  IF ( NOT .CONNECT_STATUS) THEN BAS$$STOP_IO (BAS$_IOERR_CON);
1720 2639 2  !-
1721 2640 2  !+

```



```
1722 2641 2 ! If the device opened is a terminal, set the TERM_DEV bit in the LUB
1723 2642 2 and allocate a prompt buffer. Also, set the margin to infinite.
1724 2643 2 !- Note that CONNECT is not permitted on terminals (because indexed only).
1725 2644 2
1726 2645 2
1727 2646 3 IF ( NOT .CCB [LUB$V_M_STREAM])
1728 2647 3 THEN
1729 2648 3 BEGIN
1730 2649 3
1731 2650 4 IF ((.FAB [FAB$L_DEV] AND DEV$M_TRM) NEQ 0)
1732 2651 3 THEN
1733 2652 4 BEGIN
1734 2653 4
1735 2654 4 LOCAL
1736 2655 4 GET_VM_RESULT;
1737 2656 4
1738 2657 4 CCB [LUB$V_TERM_DEV] = 1;
1739 2658 4
1740 2659 5 IF ( NOT (GET_VM_RESULT = LIB$GET_VM (%REF (LUB$K_PBUF_SIZ), CCB [RAB$L_PBF])))
1741 2660 4 THEN
1742 2661 4 BAS$$STOP_IO (BAS$K_MAXMEMEXC);
1743 2662 4
1744 2663 4 CCB [RAB$B_PSZ] = 0;
1745 2664 4 CCB [RAB$V_PMT] = 1;
1746 2665 4
1747 2666 5 IF (.OPEN_ARG_BLK [OPN$W_RECORDSIZE] EQL 0)
1748 2667 4 THEN CCB [LUB$W_D_MARGIN] = .FAB [FAB$W_BLS];
1749 2668 4
1750 2669 4 CCB [LUB$W_R_MARGIN] = 0;
1751 2670 4 CCB [LUB$V_NOMARGIN] = 1;
1752 2671 3 END;
1753 2672 3
1754 2673 3 !+
1755 2674 3 if the device opened is a line printer, set the margin to the printer's
1756 2675 3 width (if the user didn't specify RECORDSIZE).
1757 2676 3
1758 2677 3 the way we check to see if its a line printer is to see if it IS a
1759 2678 3 carriage-control device, IS an output device, IS record-oriented,
1760 2679 3 IS NOT an input device, IS NOT a mailbox, and IS NOT a terminal.
1761 2680 3 !-
1762 2681 4 IF ( ( (.FAB [FAB$L_DEV] AND DEV$M_CCL) NEQ 0 ) AND
1763 2682 4 ( (.FAB [FAB$L_DEV] AND DEV$M_ODV) NEQ 0 ) AND
1764 2683 4 ( (.FAB [FAB$L_DEV] AND DEV$M_REC) NEQ 0 ) AND
1765 2684 4 ( (.FAB [FAB$L_DEV] AND DEV$M_IDV) EQL 0 ) AND
1766 2685 4 ( (.FAB [FAB$L_DEV] AND DEV$M_MBX) EQL 0 ) AND
1767 2686 4 ( (.FAB [FAB$L_DEV] AND DEV$M_TRM) EQL 0 ) )
1768 2687 3 THEN
1769 2688 4 BEGIN
1770 2689 4
1771 2690 5 IF (.OPEN_ARG_BLK [OPN$W_RECORDSIZE] EQL 0)
1772 2691 4 THEN CCB [LUB$W_D_MARGIN] = .FAB [FAB$W_BLS];
1773 2692 4
1774 2693 4 CCB [LUB$W_R_MARGIN] = 0;
1775 2694 4 CCB [LUB$V_NOMARGIN] = 1;
1776 2695 3 END;
1777 2696 3
1778 2697 2 END;
```


BASSOPEN
1-113

L 3
16-Sep-1984 00:52:31
14-Sep-1984 11:55:24

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASOPEN.B32;91

Page 36
(9)

: 1779
: 1780

2698 2
2699 2 !<BLF/PAGE>


```
1782 2700 2 1+
1783 2701 2 1+ Version 2 stores the MAP buffer size and the recordsize in the OPEN block.
1784 2702 2 1+ If this is a version 2 program, make the check here for a recordsize longer
1785 2703 2 1+ than the user's buffer.
1786 2704 2 1-
1787 2705 2 1-
1788 2706 2 IF (.OPEN_ARG_BLK [OPN$B_CNT] GTR K_V1_BLK_SIZE) AND
1789 2707 2 (.OPEN_ARG_BLK [OPN$A_MAP] NEQA 0)
1790 2708 2 THEN
1791 2709 2 IF (.OPEN_ARG_BLK [OPN$W_MAP_SIZE] LSSU .OPEN_ARG_BLK [OPN$W_RECORDSIZE])
1792 2710 2 THEN
1793 2711 2 BAS$$STOP_IO (BAS$K_RECOVERMAP);
1794 2712 2
1795 2713 2 1+
1796 2714 2 1+ If the file just opened was already in existence, perform
1797 2715 2 1+ consistency checks between the file's attributes and the
1798 2716 2 1+ open parameters.
1799 2717 2 1-
1800 2718 2
1801 2719 2 IF (.CCB [LUB$V_OLD_FILE])
1802 2720 2 THEN
1803 2721 2 BEGIN
1804 2722 2 1+
1805 2723 2 1+ Organization check: If the user did not specify an organization
1806 2724 2 1+ with this OPEN, use the attributes from the file. Otherwise,
1807 2725 2 1+ check that the user's specification agrees with the file.
1808 2726 2 1+ Store the organization in the LUB if the user did not specify one.
1809 2727 2 1-
1810 2728 2
1811 2729 2 CASE (.OPEN_ARG_BLK [OPN$B_ORG]) FROM OPN$K_ORG_TERM1 TO OPN$K_ORG_UNDEF OF
1812 2730 2 SET
1813 2731 2
1814 2732 2 [OPN$K_ORG_TERM1] :
1815 2733 2 BEGIN
1816 2734 2 CCB [LUB$B_ORGAN] = LUB$K_ORG_TERM1;
1817 2735 2
1818 2736 2 IF (.FAB [FAB$B_ORG] NEQ FAB$C_SEQ) THEN BAS$$STOP_IO (BAS$K_FILATTNOT);
1819 2737 2
1820 2738 2 END;
1821 2739 2
1822 2740 2 [OPN$K_ORG_VIRTU] :
1823 2741 2 BEGIN
1824 2742 2 CCB [LUB$B_ORGAN] = LUB$K_ORG_VIRTU;
1825 2743 2
1826 2744 2 IF (.FAB [FAB$B_ORG] NEQ FAB$C_SEQ) THEN BAS$$STOP_IO (BAS$K_FILATTNOT);
1827 2745 2
1828 2746 2 END;
1829 2747 2
1830 2748 2 [OPN$K_ORG_SEQUE] :
1831 2749 2 BEGIN
1832 2750 2 CCB [LUB$B_ORGAN] = LUB$K_ORG_SEQUE;
1833 2751 2
1834 2752 2 IF (.FAB [FAB$B_ORG] NEQ FAB$C_SEQ) THEN BAS$$STOP_IO (BAS$K_FILATTNOT);
1835 2753 2
1836 2754 2 END;
1837 2755 2
1838 2756 2 [OPN$K_ORG_RELAT] :
```



```
1839      BEGIN
1840      CCB [LUB$B_ORGAN] = LUB$K_ORG_RELAT;
1841
1842      IF (.FAB [FAB$B_ORG] NEQ FAB$C_REL) THEN BASS$$STOP_IO (BASS$K_FILATTNOT);
1843
1844      END;
1845
1846      [OPN$K_ORG_INDEX] :
1847      BEGIN
1848      CCB [LUB$B_ORGAN] = LUB$K_ORG_INDEX;
1849
1850      IF (.FAB [FAB$B_ORG] NEQ FAB$C_IDX) THEN BASS$$STOP_IO (BASS$K_FILATTNOT);
1851
1852      END;
1853
1854      [OPN$K_ORG_UNDEF] :
1855      !+
1856      !- If the user does not specify the organization, accept whatever
1857      !- is in the file.
1858
1859      BEGIN
1860
1861      SELECTONE (.FAB [FAB$B_ORG]) OF
1862      SET
1863
1864      [FAB$C_SEQ] :
1865      CCB [LUB$B_ORGAN] = LUB$K_ORG_SEQUE;
1866
1867      [FAB$C_REL] :
1868      CCB [LUB$B_ORGAN] = LUB$K_ORG_RELAT;
1869
1870      [FAB$C_IDX] :
1871      CCB [LUB$B_ORGAN] = LUB$K_ORG_INDEX;
1872
1873      [OTHERWISE] :
1874      BASS$$STOP_IO (BASS$K_FILATTNOT);
1875
1876      TES;
1877
1878      END;
1879      TES;
1880
1881      !+
1882      !- Verify that the user-declared bucket size agrees with the file.
1883      !- If the user specified zero, we accept the file attribute.
1884
1885      IF ((.OPEN_ARG_BLK [OPN$W_BUCKETSIZ] NEQ 0) !
1886      AND ((.OPEN_ARG_BLK [OPN$B_ORG] EQL OPN$K_ORG_RELAT) !
1887      OR (.OPEN_ARG_BLK [OPN$B_ORG] EQL OPN$K_ORG_INDEX))) !
1888      THEN
1889
1890      IF (.BKS NEQ .FAB [FAB$B_BKS]) THEN BASS$$STOP_IO (BASS$K_FILATTNOT);
1891
1892      !+
1893      !- Verify that the user-declared block size agrees with the file.
1894      !- If the user specified zero, we accept the file attribute.
1895
```



```

1896 2814 3
1897 2815
1898 2816 IF ((.OPEN_ARG_BLK [OPNSW_BLOCKSIZ] NEQ 0) AND (.FAB [FAB$B_ORG] EQL FAB$C_SEQ))
1899 2817 THEN
1900 2818 IF (.BLS NEQ .FAB [FAB$W_BLS]) THEN BASS$STOP_IO (BASS$K_FILATTNOT);
1901 2819
1902 2820 +
1903 2821 Verify that the user-declared record size agrees with the file.
1904 2822 If the user specified zero, we accept the file attribute.
1905 2823 -
1906 2824 CCB [LUB$W_RBUF_SIZE] = MAXU ( .FAB [FAB$W_MRS],
1907 2825 .XABFHC [XAB$W_LRL] );
1908 2826
1909 2827 +
1910 2828 If the file's purported record size is zero, use a reasonable size.
1911 2829 -
1912 2830
1913 2831 IF (.CCB [LUB$W_RBUF_SIZE] EQL 0) THEN CCB [LUB$W_RBUF_SIZE] = .RSZ;
1914 2832
1915 2833 +
1916 2834 if the record buffer is still zero-length at this point, try looking
1917 2835 at FAB$W_BLS.
1918 2836 -
1919 2837 IF (.CCB [LUB$W_RBUF_SIZE] EQL 0) THEN CCB [LUB$W_RBUF_SIZE] = .FAB [FAB$W_BLS];
1920 2838
1921 2839 +
1922 2840 If the user specifies a record size, make sure that the file has that record size
1923 2841 (for files with fixed-length records). For files with variable-length
1924 2842 records, the check is only necessary if the MRS is set and the file is opened
1925 2843 for write access. No checking is done here for variable length record files
1926 2844 since RMS will catch the error at write time.
1927 2845 -
1928 2846 IF (.OPEN_ARG_BLK [OPNSW_RECORDSIZ] NEQ 0)
1929 2847 THEN
1930 2848 IF (.FAB [FAB$B_RFM] EQL FAB$C_FIX)
1931 2849 THEN
1932 2850 IF (.RSZ NEQ .CCB [LUB$W_RBUF_SIZE]) THEN BASS$STOP_IO (BASS$K_BADRECVAL);
1933 2851
1934 2852 +
1935 2853 for V2 programs, check the MAP size too, since if no RECORDSIZE is specified
1936 2854 but a MAP size is, OPEN_ARG_BLK[OPNSW_RECORDSIZ] will still be zero. This
1937 2855 is different from V1, where the compiler would give us the MAP value in the
1938 2856 RECORDSIZE field.
1939 2857 -
1940 2858 IF ((.OPEN_ARG_BLK [OPNSB_CNT] GTR K V1_BLK_SIZE) AND
1941 2859 (.OPEN_ARG_BLK [OPNSA_MAP] NEQ 0))
1942 2860 THEN
1943 2861 IF (.RSZ LSS .CCB [LUB$W_RBUF_SIZE]) THEN BASS$STOP_IO (BASS$K_BADRECVAL);
1944 2862
1945 2863 +
1946 2864 The buffer size we actually use must be at least as large as the user
1947 2865 declared.
1948 2866
1949 2867 If the user did not declare a buffer size, then we want to
1950 2868 use the maximum record size from the file. If no MRS is set for the file and
1951 2869 the user did not declare a buffer size, then we will use the max of the
1952 2870 default recordsize and the longest record length. This guards against
```



```

: 1953      2871 3 | opening a file with only short records in it (and MRS=0) and being unable
: 1954      2872 3 | to write a large record. 1-113
: 1955      2873 3 |
: 1956      2874 4 |     IF ((.NO_MAP_REC_SPECIFIED) AND (.FAB [FAB$W_MRS] NEQ 0))
: 1957      2875 4 |     THEN
: 1958      2876 4 |     0
: 1959      2877 4 |     ELSE
: 1960      2878 4 |     CCB [LUB$W_RBUF_SIZE] = MAXU (.CCB [LUB$W_RBUF_SIZE], .RSZ);
: 1961      2879 4 |
: 1962      2880 4 |     !+ If the user is using a MAP, the record size must not be longer than the
: 1963      2881 4 |     space in the map.
: 1964      2882 4 |     -
: 1965      2883 4 |
: 1966      2884 4 |     IF ((.OPEN_ARG_BLK [OPN$A_MAP] NEQ 0) AND
: 1967      2885 4 |     (.CCB [LUB$W_RBUF_SIZE] LSSU .OPEN_ARG_BLK [OPN$W_RECORDSIZE]))
: 1968      2886 4 |     THEN
: 1969      2887 4 |     BAS$$STOP_IO (BAS$K_BADRECVAL);
: 1970      2888 4 |
: 1971      2889 4 |     !+
: 1972      2890 4 |     If the organization is virtual, the buffer size must be at least 512 bytes.
: 1973      2891 4 |     -
: 1974      2892 4 |
: 1975      2893 4 |     IF ((.OPEN_ARG_BLK [OPN$B_ORG] EQL OPN$K_ORG_VIRTU) AND
: 1976      2894 4 |     (.CCB [LUB$W_RBUF_SIZE] LSSU 512))
: 1977      2895 4 |     THEN
: 1978      2896 4 |     BAS$$STOP_IO (BAS$K_BADRECVAL);
: 1979      2897 4 |
: 1980      2898 4 |     !+
: 1981      2899 4 |     Verify that the user-declared record format agrees with the file.
: 1982      2900 4 |     If the user did not specify a record format, we accept the file
: 1983      2901 4 |     attribute.
: 1984      2902 4 |     -
: 1985      2903 4 |
: 1986      2904 4 |     CASE (.OPEN_ARG_BLK [OPN$B_RFM]) FROM OPN$K_RFM_DEFAU TO OPN$K_RFM_STREA OF
: 1987      2905 4 |     SET
: 1988      2906 4 |     [OPN$K_RFM_DEFAU] :
: 1989      2907 4 |
: 1990      2908 4 |     !+
: 1991      2909 4 |     Don't check for virtual and undefined.
: 1992      2910 4 |     -
: 1993      2911 4 |
: 1994      2912 4 |     IF ((.OPEN_ARG_BLK [OPN$B_ORG] NEQ OPN$K_ORG_VIRTU) AND (.OPEN_ARG_BLK [OPN$B_ORG] NEQ
: 1995      2913 4 |     OPN$K_ORG_UNDEF))
: 1996      2914 4 |     THEN
: 1997      2915 4 |     !+
: 1998      2916 4 |     If the device is a terminal, it is opened in PRN format.
: 1999      2917 4 |     -
: 2000      2918 4 |
: 2001      2919 4 |     IF (((.FAB [FAB$L_DEV] AND DEV$M_TRM) NEQ 0)
: 2002      2920 4 |     AND (.OPEN_ARG_BLK [OPN$B_ORG] EQL OPN$K_ORG_TERM)
: 2003      2921 4 |     AND (.OPEN_ARG_BLK [OPN$B_RFM] EQL OPN$K_RFM_DEFAU)
: 2004      2922 4 |     AND (.OPEN_ARG_BLK [OPN$B_RAT] EQL OPN$K_RAT_DEFAU))
: 2005      2923 4 |     THEN
: 2006      2924 4 |     BEGIN
: 2007      2925 4 |
: 2008      2926 4 |     IF (.FAB [FAB$B_RFM] NEQ FAB$C_VFC) THEN BAS$$STOP_IO (BAS$K_FILATTNOT);
: 2009      2927 4 |

```



```
2010 2928 4          IF (.FAB [FAB$B_FSZ] NEQ 2) THEN BAS$$STOP_IO (BAS$K_FILATTNOT);
2011 2929 4
2012 2930 4          END;
2013 2931 4
2014 2932 4      [OPN$K_RFM_FIXED] :
2015 2933 4
2016 2934 4          IF (.FAB [FAB$B_RFM] NEQ FAB$C_FIX) THEN BAS$$STOP_IO (BAS$K_FILATTNOT);
2017 2935 4
2018 2936 4      [OPN$K_RFM_VARIA] :
2019 2937 4
2020 2938 4          IF ((.FAB [FAB$B_RFM] NEQ FAB$C_VAR) AND (.FAB [FAB$B_RFM] NEQ FAB$C_VFC))
2021 2939 4      THEN
2022 2940 4          BAS$$STOP_IO (BAS$K_FILATTNOT);
2023 2941 4
2024 2942 4      [OPN$K_RFM_VFC] :
2025 2943 4      BEGIN
2026 2944 4
2027 2945 4          IF (.FAB [FAB$B_RFM] NEQ FAB$C_VFC) THEN BAS$$STOP_IO (BAS$K_FILATTNOT);
2028 2946 4
2029 2947 4          IF (.FAB [FAB$B_FSZ] NEQ .OPEN_ARG_BLK [OPN$B_FSZ]) THEN BAS$$STOP_IO (BAS$K_FILATTNOT);
2030 2948 4
2031 2949 4      END;
2032 2950 4
2033 2951 4      [OPN$K_RFM_STREA, OUTRANGE] :
2034 2952 4      BAS$$STOP_IO (BAS$K_FILATTNOT);
2035 2953 4      TES;
2036 2954 4
2037 2955 4      !+
2038 2956 4      !- Verify that the user-declared record attributes agree with the file.
2039 2957 4
2040 2958 4
2041 2959 4      CASE (.OPEN_ARG_BLK [OPN$B_RAT]) FROM OPN$K_RAT_DEFAU TO OPN$K_RAT_ANY OF
2042 2960 4      SET
2043 2961 4
2044 2962 4      [OPN$K_RAT_DEFAU] :
2045 2963 4      !+
2046 2964 4      !- If the device is a terminal, it is opened in PRN format.
2047 2965 4
2048 2966 4
2049 2967 4      IF (((.FAB [FAB$L_DEV] AND DEV$M_TRM) NEQ 0) !
2050 2968 4          AND (.OPEN_ARG_BLK [OPN$B_ORG] EQL OPN$K_ORG_TERM) !
2051 2969 4          AND (.OPEN_ARG_BLK [OPN$B_RFM] EQL OPN$K_RFM_DEFAU) !
2052 2970 4          AND (.OPEN_ARG_BLK [OPN$B_RAT] EQL OPN$K_RAT_DEFAU))
2053 2971 4      THEN
2054 2972 4      BEGIN
2055 2973 4
2056 2974 4          IF ( NOT .FAB [FAB$V_PRN]) THEN BAS$$STOP_IO (BAS$K_RECATTNOT)
2057 2975 4
2058 2976 4      END
2059 2977 4      ELSE
2060 2978 4      !+
2061 2979 4      !- If the organization is virtual, it should have RAT NONE or CR.
2062 2980 4
2063 2981 4
2064 2982 4          IF (.OPEN_ARG_BLK [OPN$B_ORG] EQL OPN$K_ORG_VIRTU)
2065 2983 4      THEN
2066 2984 4      BEGIN
```



```

: 2067      2985  4
: 2068      2986  4          IF (.FAB [FAB$V_FTN] OR .FAB [FAB$V_PRN]) THEN BASS$STOP_IO (BASS$K_RECATTNOT)
: 2069      2987  4
: 2070      2988  4          END
: 2071      2989  4          ELSE
: 2072      2990  4
: 2073      2991  4          IF ( NOT .FAB [FAB$V_CR]) THEN BASS$STOP_IO (BASS$K_RECATTNOT);
: 2074      2992  4
: 2075      2993  4 [OPN$K_RAT_FORTR] :
: 2076      2994  4
: 2077      2995  4          IF ( NOT .FAB [FAB$V_FTN]) THEN BASS$STOP_IO (BASS$K_RECATTNOT);
: 2078      2996  4
: 2079      2997  4 [OPN$K_RAT_CRLF] :
: 2080      2998  4
: 2081      2999  4          IF ( NOT .FAB [FAB$V_CR]) THEN BASS$STOP_IO (BASS$K_RECATTNOT);
: 2082      3000  4
: 2083      3001  4 [OPN$K_RAT_NONE] :
: 2084      3002  4
: 2085      3003  4          IF (.FAB [FAB$V_CR] OR .FAB [FAB$V_FTN] OR .FAB [FAB$V_PRN])
: 2086      3004  4          THEN
: 2087      3005  4              BASS$STOP_IO (BASS$K_RECATTNOT);
: 2088      3006  4
: 2089      3007  4 [OPN$K_RAT_PRINT] :
: 2090      3008  4
: 2091      3009  4          IF ( NOT .FAB [FAB$V_PRN]) THEN BASS$STOP_IO (BASS$K_RECATTNOT);
: 2092      3010  4
: 2093      3011  4 [OPN$K_RAT_ANY] :
: 2094      3012  4          BEGIN
: 2095      3013  4          0
: 2096      3014  4          END;
: 2097      3015  4
: 2098      3016  4 [OUTRANGE] :
: 2099      3017  4          BASS$STOP_IO (BASS$K_RECATTNOT);
: 2100      3018  4          TES;
: 2101      3019  4
: 2102      3020  4 !+
: 2103      3021  4 !- Mark the file as being in PRN format, if it is.
: 2104      3022  4
: 2105      3023  4
: 2106      3024  4          IF (.FAB [FAB$V_PRN]) THEN CCB [LUB$V_PRN] = 1;
: 2107      3025  4
: 2108      3026  4 !+
: 2109      3027  4 !- Check keys for indexed file organization.
: 2110      3028  4
: 2111      3029  4
: 2112      3030  4          IF ((.OPEN_ARG_BLK [OPN$B_ORG] EQL OPN$K_ORG_INDEX) AND (ACTUALCOUNT () GEQ 2))
: 2113      3031  4          THEN
: 2114      3032  4              BEGIN
: 2115      3033  4
: 2116      3034  4              LOCAL
: 2117      3035  4                  XABKEY : REF $XABKEY DECL,
: 2118      3036  4                  KEY PTR : REF BLOCK [0, BYTE] FIELD (OPN$KEY_BLOCK),
: 2119      3037  4                  KEYNO;
: 2120      3038  4
: 2121      3039  4 !+
: 2122      3040  4 !- The number of keys specified in the argument list must be less than or
: 2123      3041  4 !- equal to the number of keys in the file, as recorded in the summary XAB.
```



```
2124 3042 4 ! If the number of keys in the argument list is more than the number of
2125 3043 4 ! keys in the file, then give the user an error.
2126 3044 4 !-
2127 3045 5 IF (.KEY_INFO_BLK [KEYH$B_KEYNUM] GTRU .XABSUM [XAB$B_NOK])
2128 3046 4 THEN
2129 3047 4 BAS$$STOP_IO (BAS$K_FILATTNOT);
2130 3048 4
2131 3049 4 !+
2132 3050 4 ! Each key must match the argument list. The keys are stored in the XABKEY
2133 3051 4 ! blocks that were set up before the OPEN. Search through the XABKEY blocks
2134 3052 4 ! matching each with the argument list.
2135 3053 4 !-
2136 3054 4 XABKEY = .FAB [FAB$L_XAB];
2137 3055 4
2138 3056 4 WHILE (.XABKEY NEQA 0) DO
2139 3057 5 BEGIN
2140 3058 5
2141 3059 6 IF (.XABKEY [XAB$B_COD] EQL XAB$C_KEY)
2142 3060 5 THEN
2143 3061 6 BEGIN
2144 3062 6 KEYNO = .XABKEY [XAB$B_REF];
2145 3063 6 KEY_PTR = (.KEY_INFO_BLK [KEYH$B_LEN]*.KEYNO) + KEYH$K_LENGTH + .KEY_INFO_BLK;
2146 3064 6
2147 3065 6 !+
2148 3066 6 ! Check the size, position, CHANGES and DUPLICATES of
2149 3067 6 ! each key.
2150 3068 6 !-
2151 3069 8 IF ((.XABKEY [XAB$W_POS0] NEQ .KEY_PTR [KEY$W_OFFSET])
2152 3070 8 OR (.XABKEY [XAB$B_SIZE] NEQ .KEY_PTR [KEY$B_LEN])
2153 3071 8 OR (.XABKEY [XAB$V_CHG] NEQ .KEY_PTR [KEY$V_CHG])
2154 3072 7 OR (.XABKEY [XAB$V_DUP] NEQ .KEY_PTR [KEY$V_DUP]))
2155 3073 6 THEN
2156 3074 6 BAS$$STOP_IO (BAS$K_FILATTNOT);
2157 3075 6
2158 3076 6 !+
2159 3077 6 ! Check the data type of each key.
2160 3078 6 ! Basic has no unsigned data type, so we must allow signed
2161 3079 6 ! word & longword keys to open a file with unsigned word and
2162 3080 6 ! longword keys.
2163 3081 6 !-
2164 3082 6 CASE .KEY_PTR [KEY$B_DTYPE] FROM DSC$K_DTYPE_W TO DSC$K_DTYPE_P OF
2165 3083 6 SET
2166 3084 6
2167 3085 6 !+
2168 3086 6 ! user program has signed word key
2169 3087 6 !-
2170 3088 6 [DSC$K_DTYPE_W] :
2171 3089 6 IF .XABKEY [XAB$B_DTP] NEQ XAB$C_IN2 ! signed word
2172 3090 6 AND .XABKEY [XAB$B_DTP] NEQ XAB$C_BN2 ! unsigned word
2173 3091 6 THEN BAS$$STOP_IO (BAS$K_FILATTNOT);
2174 3092 6
2175 3093 6 !+
2176 3094 6 ! user program has signed longword key
2177 3095 6 !-
2178 3096 6 [DSC$K_DTYPE_L] :
2179 3097 6 IF .XABKEY [XAB$B_DTP] NEQ XAB$C_IN4 ! signed longword
2180 3098 6 AND .XABKEY [XAB$B_DTP] NEQ XAB$C_BN4 ! unsigned longword
```



```
2181 3099 6
2182 3100 6
2183 3101 6
2184 3102 6
2185 3103 6
2186 3104 6
2187 3105 6
2188 3106 6
2189 3107 6
2190 3108 6
2191 3109 6
2192 3110 6
2193 3111 6
2194 3112 6
2195 3113 6
2196 3114 6
2197 3115 6
2198 3116 6
2199 3117 6
2200 3118 6
2201 3119 6
2202 3120 6
2203 3121 6
2204 3122 6
2205 3123 6
2206 3124 6
2207 3125 6
2208 3126 6
2209 3127 6
2210 3128 6
2211 3129 6
2212 3130 7
2213 3131 7
2214 3132 7
2215 3133 8
2216 3134 8
2217 3135 9
2218 3136 9
2219 3137 9
2220 3138 9
2221 3139 9
2222 3140 10
2223 3141 9
2224 3142 9
2225 3143 9
2226 3144 9
2227 3145 9
2228 3146 10
2229 3147 9
2230 3148 9
2231 3149 9
2232 3150 9
2233 3151 9
2234 3152 10
2235 3153 9
2236 3154 9
2237 3155 9
```

```
THEN BASS$STOP_IO (BASS$K_FILATTNOT);

!+
! user program has text key
[DSC$K_DTYPE_T] :
  IF .XABKEY [XAB$B_DTP] NEQ XAB$C_STG ! string
  THEN BASS$STOP_IO (BASS$K_FILATTNOT);

!+
! user program has packed decimal key
[DSC$K_DTYPE_P] :
  IF .XABKEY [XAB$B_DTP] NEQ XAB$C_PAC ! packed decimal
  THEN BASS$STOP_IO (BASS$K_FILATTNOT);

!+
! user program has some other data type key
[INRANGE, OUTRANGE] :
  IF .XABKEY [XAB$B_DTP] NEQ XAB$C_STG ! string
  THEN BASS$STOP_IO (BASS$K_FILATTNOT);

TES;

!+
! For V2 programs, check all the key segments if there are
! multiples.
IF .OPEN_ARG_BLK [OPN$B_CNT] GTR K_V1_BLK_SIZE
THEN
  BEGIN
    IF .KEY_PTR [KEY$B_NUM_SEG] GTR 0
    THEN
      BEGIN
        INCR KEY_NUM FROM 1 TO .KEY_PTR [KEY$B_NUM_SEG] DO
          BEGIN
            CASE .KEY_NUM FROM 1 TO 7 OF
              SET
              [1]:
                IF (.XABKEY [XAB$W_POS1] NEQ .KEY_PTR [KEY$W_OFFSET1]) OR
                (.XABKEY [XAB$B_SIZ1] NEQ .KEY_PTR [KEY$B_LEN1])
                THEN
                  BASS$STOP_IO (BASS$K_FILATTNOT);

              [2]:
                IF (.XABKEY [XAB$W_POS2] NEQ .KEY_PTR [KEY$W_OFFSET2]) OR
                (.XABKEY [XAB$B_SIZ2] NEQ .KEY_PTR [KEY$B_LEN2])
                THEN
                  BASS$STOP_IO (BASS$K_FILATTNOT);

              [3]:
                IF (.XABKEY [XAB$W_POS3] NEQ .KEY_PTR [KEY$W_OFFSET3]) OR
                (.XABKEY [XAB$B_SIZ3] NEQ .KEY_PTR [KEY$B_LEN3])
                THEN
                  BASS$STOP_IO (BASS$K_FILATTNOT);
```



```
2238 3156 9
2239 3157 9
2240 3158 10
2241 3159 9
2242 3160 9
2243 3161 9
2244 3162 9
2245 3163 9
2246 3164 10
2247 3165 9
2248 3166 9
2249 3167 9
2250 3168 9
2251 3169 9
2252 3170 10
2253 3171 9
2254 3172 9
2255 3173 9
2256 3174 9
2257 3175 9
2258 3176 10
2259 3177 9
2260 3178 9
2261 3179 9
2262 3180 9
2263 3181 8
2264 3182 7
2265 3183 6
2266 3184 6
2267 3185 5
2268 3186 5
2269 3187 5
2270 3188 4
2271 3189 4
2272 3190 3
2273 3191 3
2274 3192 3
2275 3193 2
2276 3194 3
2277 3195 3
2278 3196 3
2279 3197 3
2280 3198 3
2281 3199 3
2282 3200 3
2283 3201 3
2284 3202 3
2285 3203 3
2286 3204 3
2287 3205 3
2288 3206 3
2289 3207 3
2290 3208 3
2291 3209 3
2292 3210 3
2293 3211 3
2294 3212 3

[4]:
IF (.XABKEY [XAB$W_POS4] NEQ .KEY_PTR [KEY$W_OFFSET4]) OR
(.XABKEY [XAB$B_SIZ4] NEQ .KEY_PTR [KEY$B_LEN4])
THEN
  BAS$$$STOP_IO (BAS$K_FILATTNOT);

[5]:
IF (.XABKEY [XAB$W_POS5] NEQ .KEY_PTR [KEY$W_OFFSET5]) OR
(.XABKEY [XAB$B_SIZ5] NEQ .KEY_PTR [KEY$B_LEN5])
THEN
  BAS$$$STOP_IO (BAS$K_FILATTNOT);

[6]:
IF (.XABKEY [XAB$W_POS6] NEQ .KEY_PTR [KEY$W_OFFSET6]) OR
(.XABKEY [XAB$B_SIZ6] NEQ .KEY_PTR [KEY$B_LEN6])
THEN
  BAS$$$STOP_IO (BAS$K_FILATTNOT);

[7]:
IF (.XABKEY [XAB$W_POS7] NEQ .KEY_PTR [KEY$W_OFFSET7]) OR
(.XABKEY [XAB$B_SIZ7] NEQ .KEY_PTR [KEY$B_LEN7])
THEN
  BAS$$$STOP_IO (BAS$K_FILATTNOT);

TES;
END;      ! end of processing each segment
END;      ! end of if segmented key
END;      ! end of if V2 program
END;      ! end of if key XAB
XABKEY = .XABKEY [XAB$L_NXT];
END;      ! end of processing this XAB
END;      ! end of old indexed file processing
END;      ! end of old file processing
ELSE
  BEGIN
    + This is a new file. If it is organized relative or index, the user
    - must have specified a record size.

    IF (.RSZ EQL 0) THEN BAS$$$STOP_IO (BAS$K_BADRECVAL) ELSE CCB [LUB$W_RBUF_SIZE] = .RSZ;

    + Make sure the LUB 'append' flag is off so FORTRAN BACKSPACE will work.
    - CCB [LUB$V_APPEND] = 0;

    + Set LUB$B_ORGAN based on the OPEN argument. This is legitimate since
    - we just created the file, so it must agree with the OPEN argument.

    CASE .OPEN_ARG_BLK [OPN$B_ORG] FROM OPN$K_ORG_TERM1 TO OPN$K_ORG_UNDEF OF
      SET
```



```
2295 3213 3
2296 3214 3
2297 3215 3 [OPNSK_ORG_TERM] :
2298 3216 3 CCB [LOB$B_ORG] = LUB$K_ORG_TERM;
2299 3217 3
2300 3218 3 [OPNSK_ORG_VIRTU] :
2301 3219 3 CCB [LOB$B_ORG] = LUB$K_ORG_VIRTU;
2302 3220 3
2303 3221 3 [OPNSK_ORG_SEQUE] :
2304 3222 3 CCB [LOB$B_ORG] = LUB$K_ORG_SEQUE;
2305 3223 3
2306 3224 3 [OPNSK_ORG_RELAT] :
2307 3225 3 CCB [LOB$B_ORG] = LUB$K_ORG_RELAT;
2308 3226 3
2309 3227 3 [OPNSK_ORG_INDEX] :
2310 3228 3 CCB [LOB$B_ORG] = LUB$K_ORG_INDEX;
2311 3229 3
2312 3230 3 [OPNSK_ORG_UNDEF] :
2313 3231 3 BASS$STOP_IO (BASS$K_FILATTNOT);
2314 3232 3 TES;
2315 3233 3
2316 3234 3 !+ Don't allow an open with ACCESS READ to create a file.
2317 3235 3 !-
2318 3236 3
2319 3237 3 IF (.OPEN_ARG_BLK [OPNS$B_ACCESS] EQL OPNS$K_ACC_READ) THEN BASS$STOP_IO (BASS$K_ILLILLACC);
2320 3238 3
2321 3239 3 !+ A virtual file's record size must not be less than 512 bytes.
2322 3240 3 !-
2323 3241 3
2324 3242 3
2325 3243 3 IF ((.OPEN_ARG_BLK [OPNS$B_ORG] EQL OPNS$K_ORG_VIRTU) AND !
2326 3244 3 (.RSZ [SS 512]))
2327 3245 3 THEN
2328 3246 3 BASS$STOP_IO (BASS$K_BADRECVAL);
2329 3247 3
2330 3248 3 END; ! End of new file processing
2331 3249 3
2332 3250 3
2333 3251 3 !+ Validate the record format. It must be one of those the run-time
2334 3252 3 library can process. In particular, we don't permit UNDEFINED unless
2335 3253 3 the organization is UNDEFINED.
2336 3254 3 !-
2337 3255 3
2338 3256 3 IF (.OPEN_ARG_BLK [OPNS$B_ORG] NEQ OPNS$K_ORG_UNDEF)
2339 3257 3 THEN
2340 3258 3
2341 3259 3 SELECTONE (.FAB [FAB$B_RFM]) OF
2342 3260 3 SET
2343 3261 3
2344 3262 3 [FAB$C_FIX, FAB$C_VAR, FAB$C_VFC] : ! This is ok.
2345 3263 3 ;
2346 3264 3
2347 3265 3 [OTHERWISE] :
2348 3266 3 BASS$STOP_IO (BASS$K_FILATTNOT);
2349 3267 3 TES;
2350 3268 3
2351 3269 2 !+
```



```
: 2352      3270 2 | Record the record attribute, record format block size and bucket
: 2353      3271 2 | size in the LUB, for the FSP$ function and for connect.
: 2354      3272 2 |
: 2355      3273 2 |   CCB [LUB$B-RAT] = .FAB [FAB$B-RAT];
: 2356      3274 2 |   CCB [LUB$B-RFM] = .FAB [FAB$B-RFM];
: 2357      3275 2 |   CCB [LUB$B-BKS] = .FAB [FAB$B-BKS];
: 2358      3276 2 |   CCB [LUB$W-BLS] = .FAB [FAB$W-BLS];
: 2359      3277 2 |   CCB [LUB$L-ALQ] = .FAB [FAB$L-ALQ];
: 2360      3278 2 |   CCB [LUB$L-REC_MAX] =
: 2361      3279 2 |   BEGIN
: 2362      3280 2 |
: 2363      3281 2 |   CASE .OPEN_ARG_BLK [OPN$B_ORG] FROM OPN$K_ORG_TERMI TO OPN$K_ORG_UNDEF OF
: 2364      3282 2 |       SET
: 2365      3283 2 |
: 2366      3284 2 |       [OPN$K_ORG_VIRTU] :
: 2367      3285 2 |         .FAB [FAB$L-ALQ];
: 2368      3286 2 |
: 2369      3287 2 |       [OPN$K_ORG_UNDEF] :
: 2370      3288 2 |         .FAB [FAB$L-MRN];
: 2371      3289 2 |
: 2372      3290 2 |       [INRANGE] :
: 2373      3291 2 |         0;
: 2374      3292 2 |       TES
: 2375      3293 2 |
: 2376      3294 2 |   END;
: 2377      3295 2 | +
: 2378      3296 2 | Remember the device characteristics, in case the user calls
: 2379      3297 2 | STATUS.
: 2380      3298 2 | -
: 2381      3299 2 |   L_STATUS = .FAB [FAB$L-DEV];
: 2382      3300 2 | +
: 2383      3301 2 | Free the key XABs, since they were allocated from virtual storage.
: 2384      3302 2 | -
: 2385      3303 2 |   BEGIN
: 2386      3304 2 |
: 2387      3305 2 |   LOCAL
: 2388      3306 2 |     XABKEY : REF $XABKEY_DECL,
: 2389      3307 2 |     FREE_VM_STATUS,
: 2390      3308 2 |     XAB_PTR;
: 2391      3309 2 |
: 2392      3310 2 |   XAB_PTR = FAB [FAB$L-XAB];
: 2393      3311 2 |
: 2394      3312 2 |   WHILE (..XAB_PTR NEQA 0) DO
: 2395      3313 4 |     BEGIN
: 2396      3314 4 |       XABKEY = ..XAB_PTR;
: 2397      3315 4 |
: 2398      3316 5 |       IF (.XABKEY [XAB$B-COD] EQL XAB$C-KEY)
: 2399      3317 4 |       THEN
: 2400      3318 5 |         BEGIN
: 2401      3319 5 | +
: 2402      3320 5 | We have found a key XAB. Unlink it from the XAB chain and free it.
: 2403      3321 5 | We have remembered XAB_PTR as its chain location.
: 2404      3322 5 | -
: 2405      3323 5 |       .XAB_PTR = .XABKEY [XAB$L-NXT];
: 2406      3324 5 |       FREE_VM_STATUS = LIB$FREE_VM (%REF (XAB$C-KEYLEN), XABKEY);
: 2407      3325 5 |
: 2408      3326 5 |       IF ( NOT .FREE_VM_STATUS) THEN BAS$$STOP_IO (BAS$K-PROLOSSOR);
```



```

2409 3327 5
2410 3328 5
2411 3329 4
2412 3330 4
2413 3331 4
2414 3332 3
2415 3333 3
2416 3334 2
2417 3335 2
2418 3336 2
2419 3337 2
2420 3338 2
2421 3339 2
2422 3340 2
2423 3341 2
2424 3342 2
2425 3343 2
2426 3344 2
2427 3345 2
2428 3346 2
2429 3347 2
2430 3348 2
2431 3349 2
2432 3350 2
2433 3351 2
2434 3352 2
2435 3353 2
2436 3354 2
2437 3355 2
2438 3356 2
2439 3357 2
2440 3358 2
2441 3359 2
2442 3360 2
2443 3361 2
2444 3362 2
2445 3363 2
2446 3364 2
2447 3365 2
2448 3366 2
2449 3367 2
2450 3368 2
2451 3369 2
2452 3370 2
2453 3371 2
2454 3372 2
2455 3373 2
2456 3374 2
2457 3375 2
2458 3376 2
2459 3377 2
2460 3378 2
2461 3379 2
2462 3380 2
2463 3381 2
2464 3382 2
2465 3383 3

      END
    ELSE
      XAB_PTR = XABKEY [XAB$L_NXT];
    END;
  END;
+ Allocate and clear a record buffer unless the user provided one.
-
  IF (.OPEN_ARG_BLK [OPN$A_MAP] EQLA 0)
  THEN
    BEGIN
      LOCAL
        GET_VM_RESULT;
+ If recordsize is still 0 then signal an error.
-
      IF (.CCB [LUB$W_RBUF_SIZE] EQL 0) THEN BASS$STOP_IO (BASS$K_BADRECVAL);
      GET_VM_RESULT = LIB$GET_VM (%REF (.CCB [LUB$W_RBUF_SIZE]), CCB [LUB$A_RBUF_ADR]);
      IF ( NOT .GET_VM_RESULT) THEN BASS$STOP_IO (BASS$K_MAXMEMEXC);
+ Make sure the buffer is null, in case the user fetches from it before
  the first GET.
-
      CH$FILL (0, .CCB [LUB$W_RBUF_SIZE], .CCB [LUB$A_RBUF_ADR]);
    END
  ELSE
    BEGIN
      CCB [LUB$A_RBUF_ADR] = .OPEN_ARG_BLK [OPN$A_MAP];
      CCB [LUB$V_USER_RBUF] = 1;
    END;
+ Allocate dynamic storage for the file name so that the name can be
  used later for error diagnostics. Point the LUB to the new location.
  Indicate that the space pointed to must be deallocated when the file
  is closed.
-
    BEGIN
      LOCAL
        GET_VM_RESULT,
        OLD_ADDRESS;
      OLD_ADDRESS = .CCB [LUB$A_RSN];
      GET_VM_RESULT = LIB$GET_VM (%REF (.CCB [LUB$B_RSL]), CCB [LUB$A_RSN]);
      IF ( NOT .GET_VM_RESULT) THEN BASS$STOP_IO (BASS$K_MAXMEMEXC);

```



```
2466 3384 3 CH$MOVE (.CCB [LUB$B_RSL], .OLD_ADDRESS, .CCB [LUB$A_RSN]);
2467 3385 3 CCB [LUB$V_VIRT_RSN] = 1;
2468 3386 3 END;
2469 3387 2
2470 3388 2 +
2471 3389 2 | Set those RAB fields that seldom change.
2472 3390 2 |
2473 3391 2 CCB [RAB$L_UBF] = .CCB [LUB$A_RBUF_ADR];
2474 3392 2 CCB [RAB$W_USZ] = .CCB [LUB$W_RBUF_SIZE];
2475 3393 2 CCB [LUB$A_UBF] = .CCB [LUB$A_RBUF_ADR];
2476 3394 2 +
2477 3395 2 | Clear LUB$A_FAB to indicate that the FAB is no longer present.
2478 3396 2 |
2479 3397 2 CCB [LUB$A_FAB] = 0;
2480 3398 2 CCB [RAB$L_FAB] = 0;
2481 3399 2 +
2482 3400 2 | Indicate that the file is now open for BASIC.
2483 3401 2 |
2484 3402 2 CCB [LUB$B_LANGUAGE] = LUB$K_LANG_BAS;
2485 3403 2 CCB [LUB$V_OPENED] = 1;
2486 3404 2 +
2487 3405 2 | Make sure that the BASIC exit handler will be called when the image
2488 3406 2 | exits to purge the file's I/O buffers and close it, if necessary.
2489 3407 2 | This can happen if the user's last PRINT statement ends with a
2490 3408 2 | comma or a semicolon.
2491 3409 2 |
2492 3410 2
2493 3411 2 IF ( NOT .BAS$L_XIT_LOCK) THEN BAS$$DECL_EXITH ();
2494 3412 2
2495 3413 2 +
2496 3414 2 | Pop back previous LUB or indicate that no I/O statement
2497 3415 2 | is currently active.
2498 3416 2 |
2499 3417 2 BAS$$CB_POP ();
2500 3418 2 RETURN;
2501 3419 2 END;
```

! end of BASSOPEN

.TITLE BASSOPEN
.IDENT \1-113\

.PSECT _BAS\$DATA,NOEXE, PIC,2

00000000 00000 L_STATUS:

.LONG 0

.EXTRN BAS\$L_XIT_LOCK
.EXTRN LIB\$STOP, BAS\$\$STOP
.EXTRN BAS\$\$STOP_IO, BAS\$\$CB_PUSH
.EXTRN BAS\$\$CB_POP, LIB\$GET_VM
.EXTRN LIB\$FREE_VM, BAS\$\$DECL_EXITH
.EXTRN OT\$\$TAKE_LUN, OT\$\$CLOSE_FILE
.EXTRN LIB\$MATCH_COND, BAS\$K_RECOVERMAP
.EXTRN BAS\$K_PROCOSSOR
.EXTRN BAS\$K_ILLIACC
.EXTRN BAS\$K_ILLIO_CHA
.EXTRN BAS\$K_IO_CHAALR


```
.EXTRN BASSK_FATSYSIO
.EXTRN BASSK_FILATTNOT
.EXTRN BASSK_RECATTNOT
.EXTRN BASSK_MAXMEMEXC
.EXTRN BASSK_BADRECVAL
.EXTRN BASSK_NOTIMP, BASSK_INVFILOPT
.EXTRN BASSK_IO_CHANOT
.EXTRN BASSK_REQRECSIZ
.EXTRN SYSSDISPLAY, SYSSPARSE
.EXTRN SYSSOPEN, SYSSCREATE
.EXTRN SYSSCONNECT

.PSECT _BASSCODE, NOWRT, SHR, PIC, 2

.OFFC 00000
.ENTRY BASSOPEN, Save R2, R3, R4, R5, R6, R7, R8, R9, R10, -, R11
1409
MOVAB -576(SP), SP
CLRQ UNWIND_CCB
1446
MOVAL 298$, 7FP)
CLRL UNWIND_ACTION
1500
ADDL3 #44, OPEN_ARG_BLK, R0
1505
MOVL (R0), CHANNEL
50 04 AC 60 D0 00017
50 04 AC 05 C1 0001B
2C AE 60 9E C7020
11 2C BE 06 E1 00024
2C BE 07 18 0002C
44 AE 44 BE 32 0002E
44 AE 44 BE D0 00035 1$:
52 AE 44 AE D0 0003A 2$:
00000077 8F 09 19 0003E
00000000G 00 52 D1 00040
7E 00G 0B 15 00047
00000000G 00 8F 9A 00049 3$:
01 FB 0004D
52 D5 00054 4$:
0B 12 00056
00000000G 00 8F 9A 00058
01 FB 0005C
3C AE D4 00063 5$:
50 04 AC 0C C1 00066
14 AE 60 3C 0006B
58 14 AE D0 0006F
34 04 BC 91 00075
13 1B 00079
50 04 AC 34 C1 0007B
60 B5 00080
0A 13 00082
50 04 AC 34 C1 00084
58 60 3C 00089
54 11 0008C
3C AE 01 D0 0008E 6$:
50 04 AC 01 C1 00092
05 00 60 8F 00097
0010 0016 0010 0009B 7$:
000C 0010 0016 0010 0009B 7$:

MOVAB -576(SP), SP
CLRQ UNWIND_CCB
MOVAL 298$, 7FP)
CLRL UNWIND_ACTION
ADDL3 #44, OPEN_ARG_BLK, R0
MOVL (R0), CHANNEL
ADDL3 #5, OPEN_ARG_BLK, R0
MOVAB (R0), 44(SP)
BBC #6, @44(SP), 2$
TSTB @44(SP)
1510
BGEQ 1$
CVTWL @CHANNEL, CHANNEL
1512
BRB 2$
MOVL @CHANNEL, CHANNEL
1514
MOVL CHANNEL, R2
1520
BLSS 3$
CMPL R2, #119
BLEQ 4$
MOVZBL #BASSK_ILLIO_CHA, -(SP)
CALLS #1, BASS$STOP
1526
TSTL R2
BNEQ 5$
MOVZBL #BASSK_ILLIO_CHA, -(SP)
CALLS #1, BASS$STOP
1531
CLRL NO_MAP_REC_SPECIFIED
1532
ADDL3 #12, OPEN_ARG_BLK, R0
MOVZWL (R0), 20(SP)
MOVL 20(SP), RSZ
1534
BNEQ 11$
CMPB @OPEN_ARG_BLK, #52
1542
BLEQU 6$
ADDL3 #52, OPEN_ARG_BLK, R0
1543
TSTW (R0)
BEQL 6$
ADDL3 #52, OPEN_ARG_BLK, R0
1545
MOVZWL (R0), RSZ
BRB 13$
MOVL #1, NO_MAP_REC_SPECIFIED
1548
ADDL3 #1, OPEN_ARG_BLK, R0
1549
CASEB (R0), #0, #5
.WORD 9$-7$, -
```


	000C	000C	000A3				
					10\$-7\$,-		
					9\$-7\$,-		
					8\$-7\$,-		
					8\$-7\$,-		
					8\$-7\$		
			58 D4 000A7 8\$:	CLRL	RSZ		
			37 11 000A9	BRB	13\$		
	58	84	8F 9A 000AB 9\$:	MOVZBL	#132, RSZ		
			31 11 000AF	BRB	13\$		
	58	0200	8F 3C 000B1 10\$:	MOVZWL	#512, RSZ		
			2A 11 000B6	BRB	13\$		1536
50	04	AC	01 C1 000B8 11\$:	ADDL3	#1, OPEN_ARG_BLK, R0		1563
		01	60 91 000BD	CMPB	(R0), #1		
			20 12 000C0	BNEQ	13\$		
	58		01 8A 000C2	BICB2	#1, RSZ		1566
50	04	AC	10 C1 000C5	ADDL3	#16, OPEN_ARG_BLK, R0		1576
			60 D5 000CA	TSTL	(R0)		
			14 12 000CC	BNEQ	13\$		
	50		58 D0 000CE	MOVL	RSZ, R0		
00000200		8F	50 D1 000D1	CMPL	R0, #512		
			05 18 000D8	BGEQ	12\$		
	50	0200	8F 3C 000DA	MOVZWL	#512, R0		
			50 D0 000DF 12\$:	MOVL	R0, RSZ		
50	04	AC	1A C1 000E2 13\$:	ADDL3	#26, OPEN_ARG_BLK, R0		1580
		5A	60 32 000E7	CVTBL	(R0), BKS		
			48 13 000EA	BEQL	20\$		1582
50	04	AC	03 C1 000EC	ADDL3	#3, OPEN_ARG_BLK, R0		1590
		02	60 91 000F1	CMPB	(R0), #2		
			05 12 000F4	BNEQ	14\$		
	50		02 D0 000F6	MOVL	#2, R0		
			02 11 000F9	BRB	15\$		
			50 D4 000FB 14\$:	CLRL	R0		
	50		58 C0 000FD 15\$:	ADDL2	RSZ, R0		1587
			53 D4 00100	CLRL	R3		1596
51	04	AC	01 C1 00102	ADDL3	#1, OPEN_ARG_BLK, R1		
		04	61 91 00107	CMPB	(R1), #4		
			07 12 0010A	BNEQ	16\$		
			53 D6 0010C	INCL	R3		
	51		07 D0 0010E	MOVL	#7, R1		
			03 11 00111	BRB	17\$		
	51		01 D0 00113 16\$:	MOVL	#1, R1		
	50		51 C0 00116 17\$:	ADDL2	R1, R0		1594
			5A C4 00119	MULL2	BKS, R0		1587
	05		53 E9 0011C	BLBC	R3, 18\$		1602
	51		0F D0 0011F	MOVL	#15, R1		
			02 11 00122	BRB	19\$		
			51 D4 00124 18\$:	CLRL	R1		
	50	01FF C140	9E 00126 19\$:	MOVAB	511(R1)(R0), R0		1587
5A			8F C7 0012C	DIVL3	#512, R0, BKS		1605
50	04	AC	0E C1 00134 20\$:	ADDL3	#14, OPEN_ARG_BLK, R0		1612
		51	60 3C 00139	MOVZWL	(R0), R1		
			03 14 0013C	BGTR	21\$		
	51		01 D0 0013E	MOVL	#1, R1		
50	04	AC	03 C1 00141 21\$:	ADDL3	#3, OPEN_ARG_BLK, R0		1615
	24	AE	60 98 00146	CVTBL	(R0), 36(SP)		
		02	AE 91 0014A	CMPB	36(SP), #2		
		24	06 12 0014E	BNEQ	22\$		

	50	04	A8	9E	00150	MOVAB	4(R8), R0		
			03	11	00154	BRB	23\$		
	50		58	D0	00156	22\$:	MOVL	RSZ, R0	
	50		51	C4	00159	23\$:	MULL2	R1, R0	1613
	50		03	C0	0015C		ADDL2	#3, R0	1612
	50		03	8A	0015F		BICB2	#3, R0	1618
	14		50	D1	00162		CMPL	R0, #20	1612
			03	18	00165		BGEQ	24\$	
	50		14	D0	00167		MOVL	#20, R0	
40	AE		50	D0	0016A	24\$:	MOVL	R0, BLS	
			50	D4	0016E		CLRL	R0	1625
		00000000G	00	16	00170		JSB	BASS\$CB_PUSH	
50	AE		5B	D0	00176		MOVL	CCB, UNWIND_CCB	1629
54	AE		01	D0	0017A		MOVL	#1, UNWIND_ACTION	1630
	26	FC	AB	E9	0017E		BLBC	-4(CCB), 28\$	1637
00000000G	00		00	FB	00182		CALLS	#0, OTS\$\$CLOSE_FILE	1641
	0A		50	E8	00189		BLBS	R0, 25\$	
	7E		01	CE	0018C		MNEGL	#1, -(SP)	
00000000G	00		01	FB	0018F		CALLS	#1, BASS\$STOP_IO	
		00000000G	00	16	00196	25\$:	JSB	BASS\$CB_POP	1648
			50	D4	0019C		CLRL	R0	1649
		00000000G	00	16	0019E		JSB	BASS\$CB_PUSH	
50	AE		5B	D0	001A4		MOVL	CCB, UNWIND_CCB	1653
	59	FC	AB	9E	001A8	26\$:	MOVAB	-4(CCB), R9	1664
	0A		69	E8	001AC		BLBS	(R9), 27\$	
05	FF		04	E0	001AF		BBS	#4, -1(CCB), 27\$	
		E8	AB	D5	001B4		TSTL	-24(CCB)	
			0B	13	001B7		BEQL	28\$	
	7E	00G	8F	9A	001B9	27\$:	MOVZBL	#BASSK_IO_CHAALR, -(SP)	1666
00000000G	00		01	FB	001BD		CALLS	#1, BASS\$STOP_IO	
50	04		08	C1	001C4	28\$:	ADDL3	#8, OPEN_ARG_BLK, R0	1673
	57		60	D0	001C9		MOVL	(R0), FILE_NAME_DESC	
	F8	04	A7	D0	001CC		MOVL	4(FILE_NAME_DESC), -8(CCB)	1674
	50		67	3C	001D1		MOVZWL	(FILE_NAME_DESC), R0	1675
00FF	8F		50	B1	001D4		CMPL	R0, #255	
			04	1B	001D9		BLEQU	29\$	
	50	FF	8F	9A	001DB		MOVZBL	#255, R0	
	F7		50	90	001DF	29\$:	MOVB	R0, -9(CCB)	
		44	AE	9F	001E3		PUSHAB	CHANNEL	1680
00000000G	00		01	FB	001E6		CALLS	#1, OTS\$\$TAKE_LUN	
	0B		50	E8	001ED		BLBS	R0, 30\$	
	7E	00G	8F	9A	001F0		MOVZBL	#BASSK_IO_CHAALR, -(SP)	
00000000G	00		01	FB	001F4		CALLS	#1, BASS\$STOP_IO	
	54		02	D0	001FB	30\$:	MOVL	#2, UNWIND_ACTION	1685
	AE	B0	AD	9E	001FF		MOVAB	FAB_BLOCK, -FAB	1690
0050	8F	00	00	2C	00203		MOVCS	#0, -(SP), #0, #80, (FAB)	1691
	6E		66		0020A				
		5003	8F	B0	0020B		MOVW	#20483, (FAB)	
	1F		02	90	00210		MOVB	#2, 31(FAB)	
50	04		14	C1	00214		ADDL3	#20, OPEN_ARG_BLK, R0	1692
	10		60	D0	00219		MOVL	(R0), 16(FAB)	
50	04		01	C1	0021D		ADDL3	#1, OPEN_ARG_BLK, R0	1694
	08		60	98	00222		CVTBL	(R0), 8(SP)	
		38	AE	D4	00226		CLRL	56(SP)	
	03	08	AE	91	00229		CMPL	8(SP), #3	
			05	12	0022D		BNEQ	31\$	
		38	AE	D6	0022F		INCL	56(SP)	

		04	08	06	11	00232	BRB	32\$		
				AE	91	00234	CMPB	8(SP), #4		
		50		14	12	00238	BNEQ	34\$		
	000000FF	8F		5A	D0	0023A	MOVL	BKS, R0	1696	
				50	D1	0023D	CMPL	R0, #255		
		50	FF	04	15	00244	BLEQ	33\$		
	3E	A6		8F	9A	00246	MOVZBL	#255, R0		
		0B	38	50	90	0024A	MOVB	R0, 62(FAB)	1698	
		04	08	AE	E8	0024E	BLBS	56(SP), 35\$	1699	
				AE	91	00252	CMPB	8(SP), #4		
	3C	A6	40	05	13	00256	BEQL	35\$		
50	04	AC		AE	B0	00258	MOVW	BLS, 60(FAB)	1701	
	14	A6		18	C1	0025D	ADDL3	#24, OPEN_ARG_BLK, R0	1703	
50	04	AC		60	B0	00262	MOVW	(R0), 20(FAB)		
		57		24	C1	00266	ADDL3	#36, OPEN_ARG_BLK, R0	1707	
				60	D0	0026B	MOVL	(R0), FILE_NAME_DESC		
	30	A6	04	17	13	0026E	BEQL	37\$	1709	
		50		A7	D0	00270	MOVL	4(FILE_NAME_DESC), 48(FAB)	1712	
	00FF	8F		67	3C	00275	MOVZWL	(FILE_NAME_DESC), R0	1713	
				50	B1	00278	CMPW	R0, #255		
		50	FF	04	1B	0027D	BLEQU	36\$		
	35	A6		8F	9A	0027F	MOVZBL	#255, R0		
51	04	AC		50	90	00283	MOVB	R0, 53(FAB)		
		6E		02	C1	00287	ADDL3	#2, OPEN_ARG_BLK, R1	1720	
		50		61	98	0028C	CVTBL	(R1), (SP)		
				01	D0	0028F	MOVL	#1, R0		
				6E	D5	00292	TSTL	(SP)	1723	
		01		0B	19	00294	BLSS	38\$		
				6E	91	00296	CMPB	(SP), #1		
				06	14	00299	BGTR	38\$		
	16	A6		50	D4	0029B	CLRL	R0		
				04	88	0029D	BISB2	#4, 22(FAB)	1724	
				6E	D5	002A1	TSTL	(SP)	1726	
		01		05	19	002A3	BLSS	39\$		
				6E	91	002A5	CMPB	(SP), #1		
		03		0A	15	002A8	BLEQ	40\$		
				6E	91	002AA	CMPB	(SP), #3		
		04		0B	19	002AD	BLSS	41\$		
				6E	91	002AF	CMPB	(SP), #4		
				06	14	002B2	BGTR	41\$		
	16	A6		50	D4	002B4	CLRL	R0		
				02	88	002B6	BISB2	#2, 22(FAB)	1727	
				6E	D5	002BA	TSTL	(SP)	1729	
		02		05	19	002BC	BLSS	42\$		
				6E	91	002BE	CMPB	(SP), #2		
		04		0A	15	002C1	BLEQ	43\$		
				6E	91	002C3	CMPB	(SP), #4		
		05		0B	19	002C6	BLSS	44\$		
				6E	91	002C8	CMPB	(SP), #5		
				06	14	002CB	BGTR	44\$		
				50	D4	002CD	CLRL	R0		
	16	A6		01	88	002CF	BISB2	#1, 22(FAB)	1730	
		04		6E	91	002D3	CMPB	(SP), #4	1732	
				06	12	002D6	BNEQ	45\$		
				50	D4	002D8	CLRL	R0		
	16	A6		10	88	002DA	BISB2	#16, 22(FAB)	1733	
				6E	D5	002DE	TSTL	(SP)	1735	

		01	05	19	002E0	BLSS	46\$			
		04	6E	91	002E2	CMPB	(SP), #1			
			05	15	002E5	BLEQ	47\$			
			6E	91	002E7	CMPB	(SP), #4			
			06	12	002EA	BNEQ	48\$			
			50	D4	002EC	CLRL	R0			
	16	A6	08	88	002EE	BISB2	#8, 22(FAB)			1736
		0B	50	E9	002F2	BLBC	R0, 49\$			1738
		7E	00G	8F	9A	MOVZBL	#BASSK_PROLOSSOR, -(SP)			1739
	00000000G	00	01	FB	002F9	CALLS	#1, BASS\$STOP_IO			
50		04	08	C1	00300	ADDL3	#8, OPEN_ARG_BLK, R0			1745
		57	60	D0	00305	MOVL	(R0), FILE_NAME_DESC			
	2C	A6	04	A7	D0	MOVL	4(FILE_NAME_DESC), 44(FAB)			1746
		57	67	3C	0030D	MOVZWL	(FILE_NAME_DESC), R7			1747
	00FF	8F	57	B1	00310	CMPW	R7, #255			
			04	1B	00315	BLEQU	50\$			
		57	FF	8F	9A	MOVZBL	#255, R7			
	34	A6	57	90	0031B	MOVB	R7, 52(FAB)			
09		2C	05	E0	0031F	BBS	#5, @44(SP), 51\$			1771
04		2C	04	E0	00324	BBS	#4, @44(SP), 51\$			
		07	02	88	00329	BISB2	#2, 7(FAB)			
04		2C	01	E1	0032D	BBC	#1, @44(SP), 52\$			1778
		06	20	88	00332	BISB2	#32, 6(FAB)			
50		04	04	C1	00336	ADDL3	#4, OPEN_ARG_BLK, R0			1785
			60	98	0033B	CVTBL	(R0), R2			
			52	91	0033E	CMPB	R2, #2			
			09	13	00341	BEQL	53\$			
		01	52	91	00343	CMPB	R2, #1			
			04	13	00346	BEQL	53\$			
	04	A6	20	88	00348	BISB2	#32, 4(FAB)			1788
		05	6E	91	0034C	CMPB	(SP), #5			1795
			04	13	0034F	BEQL	54\$			
	05	A6	04	88	00351	BISB2	#4, 5(FAB)			
		2C	02	E0	00355	BBS	#2, @44(SP), 55\$			1801
		04	8F	88	0035A	BISB2	#128, 4(FAB)			
			80	AE	D4	CLRL	52(SP)			1809
			34	AE	D5	TSTL	8(SP)			
			08	12	00365	BNEQ	56\$			
			34	AE	D6	INCL	52(SP)			
		04	40	8F	88	BISB2	#64, 4(FAB)			
04		2C	04	E1	0036F	BBC	#4, @44(SP), 57\$			1818
		04	04	88	00374	BISB2	#4, 4(FAB)			
04		2C	03	E1	00378	BBC	#3, @44(SP), 58\$			1825
		04	10	88	0037D	BISB2	#16, 4(FAB)			
50		04	07	C1	00381	ADDL3	#7, OPEN_ARG_BLK, R0			1827
		3F	60	90	00386	MOVB	(R0), 63(FAB)			
		36	58	B0	0038A	MOVW	RSZ, 54(FAB)			1828
			08	AE	8F	CASEB	8(SP), #0, #5			1834
0019	0025	0025	0025	001F	00393	.WORD	62\$-59\$,-			
		0025			0039B		62\$-59\$,-			
							62\$-59\$,-			
							60\$-59\$,-			
							61\$-59\$,-			
							62\$-59\$			
		7E	00G	8F	9A	MOVZBL	#BASSK_PROLOSSOR, -(SP)			1856
	00000000G	00	01	FB	003A3	CALLS	#1, BASS\$STOP_IO			
			0F	11	003AA	BRB	63\$			

	1D	A6		10	90	003AC	60\$:	MOVB	#16, 29(FAB)	1847
				09	11	003B0		BRB	63\$	
	1D	A6		20	90	003B2	61\$:	MOVB	#32, 29(FAB)	1850
				03	11	003B6		BRB	63\$	
		04	1D	A6	94	003B8	62\$:	CLRB	29(FAB)	1853
		A6	2C	BE	E9	003BB	63\$:	BLBC	244(SP), 64\$	1864
	50	04		08	88	003BF		BISB2	#8, 30(FAB)	
		AC		06	C1	003C3	64\$:	ADDL3	#6, OPEN_ARG_BLK, R0	1866
	05	AE		60	98	003C8		CVTBL	(R0), 16(SP)	
0031	0027	00	10	AE	8F	003CC		CASEB	16(SP), #0, #5	
		0021		0019		003D1	65\$:	.WORD	66\$-65\$,-	
		0019		002D		003D9			67\$-65\$,-	
									68\$-65\$,-	
									70\$-65\$,-	
									69\$-65\$,-	
									66\$-65\$	
		7E	00G	8F	9A	003DD		MOVZBL	#BASSK PROLOSSOR, -(SP)	1888
	00000000G	00		01	FB	003E1		CALLS	#1, BASS\$STOP_IO	
				18	11	003E8		BRB	70\$	
		01	08	AE	91	003EA	66\$:	CMPB	8(SP), #1	1871
				12	13	003EE		BEQL	70\$	
				06	11	003F0		BRB	68\$	
		1E		01	88	003F2	67\$:	BISB2	#1, 30(FAB)	1874
		A6		0A	11	003F6		BRB	70\$	
				02	88	003F8	68\$:	BISB2	#2, 30(FAB)	1877
		1E		04	11	003FC		BRB	70\$	
		A6		04	88	003FE	69\$:	BISB2	#4, 30(FAB)	1885
0037	04	00	24	AE	8F	00402	70\$:	CASEB	36(SP), #0, #4	1895
	0031	002B		0017		00407	71\$:	.WORD	73\$-71\$,-	
				000A		0040F			76\$-71\$,-	
									77\$-71\$,-	
									78\$-71\$,-	
									72\$-71\$	
		7E	00G	8F	9A	00411	72\$:	MOVZBL	#BASSK PROLOSSOR, -(SP)	1916
	00000000G	00		01	FB	00415		CALLS	#1, BASS\$STOP_IO	
				24	11	0041C		BRB	79\$	
		01	08	AE	91	0041E	73\$:	CMPB	8(SP), #1	1902
				05	12	00422		BNEQ	74\$	
		50		01	D0	00424		MOVL	#1, R0	
				03	11	00427		BRB	75\$	
		50		02	D0	00429	74\$:	MOVL	#2, R0	
		A6		50	90	0042C	75\$:	MOVB	R0, 31(FAB)	1900
				10	11	00430		BRB	79\$	1899
		1F		01	90	00432	76\$:	MOVB	#1, 31(FAB)	1907
		A6		0A	11	00436		BRB	79\$	
				02	90	00438	77\$:	MOVB	#2, 31(FAB)	1910
		1F		04	11	0043C		BRB	79\$	
		A6		03	90	0043E	78\$:	MOVB	#3, 31(FAB)	1913
	50	AC		1C	C1	00442	79\$:	ADDL3	#28, OPEN_ARG_BLK, R0	1919
		A6		60	90	00447		MOVB	(R0), 28(FAB)	
		50		01	D0	0044B		MOVL	#1, R0	1926
		01		52	91	0044E		CMPB	R2, #1	1929
				06	12	00451		BNEQ	80\$	
				50	D4	00453		CLRL	R0	
		17		04	88	00455		BISB2	#4, 23(FAB)	1930
		A6		52	91	00459	80\$:	CMPB	R2, #1	1932
		01		05	13	0045C		BEQL	81\$	

		03		52	91	0045E		CMPB	R2, #3		
				06	12	00461		BNEQ	82\$		
				50	D4	00463	81\$:	CLRL	R0		
	17	A6		02	88	00465		BISB2	#2, 23(FAB)	1933	
		04		52	91	00469	82\$:	CMPB	R2, #4	1935	
				05	13	0046C		BEQL	83\$		
		06		52	91	0046E		CMPB	R2, #6		
				06	12	00471		BNEQ	84\$		
				50	D4	00473	83\$:	CLRL	R0		
	17	A6		20	88	00475		BISB2	#32, 23(FAB)	1936	
				52	D5	00479	84\$:	TSTL	R2	1938	
		02		0B	15	0047B		BLEQ	85\$		
				52	91	0047D		CMPB	R2, #2		
				06	14	00480		BGTR	85\$		
				50	D4	00482		CLRL	R0		
	17	A6		01	88	00484		BISB2	#1, 23(FAB)	1939	
		01		52	91	00488	85\$:	CMPB	R2, #1	1941	
				06	12	0048B		BNEQ	86\$		
				50	D4	0048D		CLRL	R0		
	17	A6		08	88	0048F		BISB2	#8, 23(FAB)	1942	
				52	D5	00493	86\$:	TSTL	R2	1944	
				02	12	00495		BNEQ	87\$		
				50	D4	00497		CLRL	R0		
		0B		50	E9	00499	87\$:	BLBC	R0, 88\$	1949	
		7E	00G	8F	9A	0049C		MOVZBL	#BAS\$K_PROLOSSOR, -(SP)	1950	
	00000000G	00		01	FB	004A0		CALLS	#1, BAS\$\$STOP_IO		
			30	AE	D4	004A7	88\$:	CLRL	48(SP)	1956	
		04	08	AE	91	004AA		CMPB	8(SP), #4		
				07	12	004AE		BNEQ	89\$		
			30	AE	D6	004B0		INCL	48(SP)		
	17	A6		10	88	004B3		BISB2	#16, 23(FAB)		
	E8	AB		56	D0	004B7	89\$:	MOVL	FAB, -24(CCB)	1962	
	F8	AB	2C	A6	D0	004BB		MOVL	44(FAB), -8(CCB)	1966	
	F7	AB	34	A6	90	004C0		MOVB	52(FAB), -9(CCB)	1967	
		00		6E	8F	004C5		CASEB	(SP), #0, #5	1974	
0011	05			000C		004C9	90\$:	.WORD	91\$-90\$,-		
	000C	000C		0019		004D1			91\$-90\$,-		
		0025							91\$-90\$,-		
									92\$-90\$,-		
									94\$-90\$,-		
									96\$-90\$,-		
		69		04	8A	004D5	91\$:	BICB2	#4, (R9)	1979	
				03	11	004D8		BRB	93\$	1980	
		69		0C	88	004DA	92\$:	BISB2	#12, (R9)	1987	
		69		20	8A	004DD	93\$:	BICB2	#32, (R9)	1988	
				06	11	004E0		BRB	95\$	1989	
		69		04	8A	004E2	94\$:	BICB2	#4, (R9)	1994	
		69		20	88	004E5		BISB2	#32, (R9)	1995	
	01	A9		20	8A	004E8	95\$:	BICB2	#32, 1(R9)	1996	
				07	11	004EC		BRB	97\$	1974	
		69		24	8A	004EE	96\$:	BICB2	#36, (R9)	2002	
	01	A9		20	88	004F1		BISB2	#32, 1(R9)	2003	
		00	08	AE	8F	004F5	97\$:	CASEB	8(SP), #0, #5	2011	
0012	05			000C		004FA	98\$:	.WORD	99\$-98\$,-		
	000C	0012		0012		00502			100\$-98\$,-		
		0012							99\$-98\$,-		
									100\$-98\$,-		

			33	11	005AE	BRB	121\$	2068
			14	AE	D5 005B0	117\$: TSTL	20(SP)	2107
		50	06	13	005B3	BEQL	118\$	
			14	AE	D0 005B5	MOVL	20(SP), R0	2109
		34	1D	11	005B9	BRB	120\$	
			04	BC	91 005BB	118\$: CMPB	@OPEN_ARG_BLK, #52	2115
			13	1B	005BF	BLEQU	119\$	
50	04	AC	34	C1	005C1	ADDL3	#52, OPEN_ARG_BLK, R0	2116
			60	B5	005C6	TSTW	(R0)	
			0A	13	005C8	BEQL	119\$	
51	04	AC	34	C1	005CA	ADDL3	#52, OPEN_ARG_BLK, R1	2117
		50	61	3C	005CF	MOVZWL	(R1), R0	
			04	11	005D2	BRB	120\$	
		50	48	8F	9A 005D4	119\$: MOVZBL	#72, R0	2115
	D6	AB	50	B0	005D8	120\$: MOVW	R0, -42(CCB)	2105
			D4	AB	B4 005DC	CLRW	-44(CCB)	2121
	A1	AB	02	88	005DF	BISB2	#2, -95(CCB)	2122
			50	D4	005E3	121\$: CLRL	R0	2133
		02	6C	91	005E5	CMPB	(AP), #2	
			02	1F	005E8	BLSSU	122\$	
			50	D6	005EA	INCL	R0	
69	01	OF	50	F0	005EC	122\$: INSV	R0, #15, #1, (R9)	
	50	AC	1E	C1	005F1	ADDL3	#30, OPEN_ARG_BLK, R0	2138
			60	B5	005F6	TSTW	(R0)	
			06	13	005F8	BEQL	123\$	
		FF	04	88	005FA	BISB2	#4, -1(CCB)	2140
			04	11	005FE	BRB	124\$	
		FF	04	8A	00600	123\$: BICB2	#4, -1(CCB)	2142
		04	34	AE	E9 00604	124\$: BLBC	52(SP), 125\$	2149
		FE	10	88	00608	BISB2	#16, -2(CCB)	
		6B	4401	8F	B0 0060C	125\$: MOVW	#17409, (CCB)	2155
		3C	56	D0	00611	MOVL	FAB, 60(CCB)	2157
		AB	E0	AB	9E 00615	MOVAB	-32(R11), 48(CCB)	2163
		30	04	90	0061A	MOVB	#4, 52(CCB)	2164
50	04	AC	20	C1	0061E	ADDL3	#32, OPEN_ARG_BLK, R0	2165
			60	90	00623	MOVB	(R0), 54(CCB)	
50	04	AC	07	C1	00627	ADDL3	#7, OPEN_ARG_BLK, R0	2167
			60	95	0062C	TSTB	(R0)	
			09	13	0062E	BEQL	126\$	
50	04	AC	30	C1	00630	ADDL3	#48, OPEN_ARG_BLK, R0	
	2C	AB	60	D0	00635	MOVL	(R0), 44(CCB)	
			2C	AE	D4 00639	126\$: CLRL	44(SP)	2176
		34	04	BC	91 0063C	CMPB	@OPEN_ARG_BLK, #52	
			2A	1B	00640	BLEQU	128\$	
			2C	AE	D6 00642	INCL	44(SP)	
50	04	AC	36	C1	00645	ADDL3	#54, OPEN_ARG_BLK, R0	2178
		1F	60	E9	0064A	BLBC	(R0), 128\$	
		02	08	AE	91 0064D	CMPB	8(SP), #2	2186
			15	12	00651	BNEQ	127\$	
	0200	8F	14	AE	B1 00653	CMPW	20(SP), #512	2187
			0D	13	00659	BEQL	127\$	
		7E	00G	8F	9A 0065B	MOVZBL	#BAS\$K REQRECSIZ, -(SP)	2188
00000000G	00		01	FB	0065F	CALLS	#1, BAS\$\$STOP_10	
			04	11	00666	BRB	128\$	
	06	AB	04	88	00668	127\$: BISB2	#4, 6(CCB)	2190
		05	6E	91	0066C	128\$: CMPB	(SP), #5	2198
			04	12	0066F	BNEQ	129\$	

		05	AB		01	88	00671		BISB2	#1, 5(CCB)		
		04	AB	00010002	8F	CA	00675	129\$:	BICL2	#65538, 4(CCB)		2205
					28	AE	D4	0067D	CLRL	40(SP)		2216
			01		08	AE	91	00680	CMPB	8(SP), #1		
						03	12	00684	BNEQ	130\$		
					28	AE	D6	00686	INCL	40(SP)		
04	AB	01	04		28	AE	F0	00689	INSV	40(SP), #4, #1, 4(CCB)		
0060	8F	00	6E			00	2C	00690	MOVCS	#0, (SP), #0, #96, \$RMS_PTR		2222
				FF50	CD			00697				
				6002	8F	B0	0069A		MOVW	#24578, \$RMS_PTR		
				58	AE	9E	006A1		MOVAB	FILE_NAME, R0		2223
					50	D0	006A5		MOVL	R0, NAM_BLOCK+12		
				FF5C	CD	50	D0	006AA	MOVL	R0, NAM_BLOCK+4		
				FF54	CD	01	8E	006AF	MNEGB	#1, NAM_BLOCK+10		2224
				FF5A	CD	01	8E	006B4	MNEGB	#1, NAM_BLOCK+2		
				FF52	CD				MOVAB	NAM_BLOCK, 40(FAB)		2225
				28	A6	CD	9E	006B9	MOVCS	#0, (SP), #0, #44, \$RMS_PTR		2233
	2C	00	6E			00	2C	006BF				
				FF24	CD			006C4				
				2C1D	8F	B0	006C7		MOVW	#11293, \$RMS_PTR		
				FF24	CD	9E	006CE		MOVAB	XABSUM, \$RMS_PTR+4		
				FF18	CD	00	2C	006D5	MOVCS	#0, (SP), #0, #12, \$RMS_PTR		2234
						00		006DA				
				FF18	CD	8F	B0	006DD	MOVW	#3094, \$RMS_PTR		
				OC16	CD	D4	006E4		CLRL	XABSUM+4		2235
				FF1C	CD	A6	9E	006E8	MOVAB	36(FAB), 12(SP)		2236
				24	AE	CD	9E	006ED	MOVAB	XABFHC, @12(SP)		
				FF24	CD	91	006F3		CMPB	(AP), #2		2241
					6C	91	006F6		BGEQU	131\$		
					03	1E	006F8		BRW	161\$		
					0228	31	006FB	131\$:	MOVL	KEY_INFO_BLK, 32(SP)		2257
					08	AC	D0	006FB	MOVZBL	@KEY_INFO_BLK, KEYNO		
					08	BC	9A	00700	BRW	159\$		
					0214	31	00704	132\$:	ADDL3	#1, 32(SP), R1		
					01	C1	00707		MOVZBL	(R1), R0		
					61	9A	0070C		MULL2	KEYNO, R0		
					59	C4	0070F		ADDL2	KEY_INFO_BLK, R0		
					08	AC	C0	00712	MOVAB	4(R0), KEY_PTR		2263
					04	A0	9E	00716	PUSHAB	XABKEY		
					48	AE	9F	0071A	MOVZBL	#76, 32(SP)		
					4C	8F	9A	0071D	PUSHAB	32(SP)		
					20	AE	9F	00722	CALLS	#2, LIB\$GET_VM		
						02	FB	00725	BLBS	GET_VM_STATUS, 133\$		2265
						50	E8	0072C	MOVZBL	#BAS\$K-MAXMEMEXC, -(SP)		
					00G	8F	9A	0072F	CALLS	#1, BAS\$\$STOP_IO		
						01	FB	00733	MOVCS	#0, (SP), #0, #76, @XABKEY		2268
						00	2C	0073A				
						48	BE	00741				
						15	90	00743	MOVB	#21, @XABKEY		
						01	C1	00747	ADDL3	#1, XABKEY, R0		
						8F	90	0074C	MOVB	#76, (R0)		
						13	C1	00750	ADDL3	#19, XABKEY, R0		
						60	94	00755	CLRB	(R0)		
						17	C1	00757	ADDL3	#23, XABKEY, R0		2272
						59	90	0075C	MOVB	KEYNO, (R0)		
						1E	C1	0075F	ADDL3	#30, XABKEY, R0		2273
						A7	B0	00764	MOVW	2(KEY_PTR), (R0)		
						2E	C1	00768	ADDL3	#46, XABKEY, R0		2274
						67	90	0076D	MOVB	(KEY_PTR), (R0)		

PC	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

K 5
16-Sep-1984 00:52:31 VAX-11 BLISS-32 V4.0-742
14-Sep-1984 11:55:24 [BASRTL.SRC]BASOPEN.B32:91

Page 61
(10)[illegible]

53	48	AE		34	C1	008EF	ADDL3	#52, XABKEY, R3	2395
		51		63	9A	008F4	MOVZBL	(R3), R1	
		50		51	C0	008F7	ADDL2	R1, R0	2394
53	48	AE		35	C1	008FA	ADDL3	#53, XABKEY, R3	2395
		51		63	9A	008FF	MOVZBL	(R3), R1	
		50		51	C0	00902	ADDL2	R1, R0	
55	48	AE		16	C1	00905	ADDL3	#22, XABKEY, R5	2388
		65		50	90	0090A	MOVB	R0, (R5)	
50	48	AE		04	C1	0090D	ADDL3	#4, XABKEY, R0	2402
		60		BE	D0	00912	MOVL	@12(SP), (R0)	
	0C	BE	OC	AE	D0	00916	MOVL	XABKEY, @12(SP)	2403
		02		59	F4	0091B	SOBGEQ	KEYNO, 160\$	2255
				03	11	0091E	BRB	161\$	
				FDE4	31	00920	BRW	132\$	
03	FF	AB		02	E0	00923	BBS	#2, -1(CCB), 162\$	2414
				009B	31	00928	BRW	170\$	
		54		5B	D0	0092B	MOVL	CCB, OUR_CCB	2429
				50	D4	0092E	CLRL	R0	2430
53	04	AC		1E	C1	00930	ADDL3	#30, OPEN_ARG_BLK, R3	
		52		63	32	00935	CVTL	(R3), R2	
			00000000G	00	16	00938	JSB	BAS\$\$CB_PUSH	
		06		AB	E9	0093E	BLBC	-4(CCB), 163\$	2431
		53		AB	3C	00942	MOVZWL	-48(CCB), PARENT_IFI	
				02	11	00946	BRB	164\$	
				53	D4	00948	CLRL	PARENT_IFI	
	20	AE		AB	9A	0094A	MOVZBL	-60(CCB), PARENT_ORG	2432
	1C	AE		AB	3C	0094F	MOVZWL	-46(CCB), PARENT_MRS	2433
	18	AE		AB	9A	00954	MOVZBL	-10(CCB), PARENT_RAT	2434
		59		AB	9A	00959	MOVZBL	-39(CCB), PARENT_RFM	2435
		57		AB	9A	0095D	MOVZBL	-59(CCB), PARENT_BKS	2436
		55		AB	3C	00961	MOVZWL	-94(CCB), PARENT_BLS	2437
05	FF	AB		02	E1	00965	BBC	#2, -1(CCB), 165\$	2443
		52		01	D0	0096A	MOVL	#1, CONNECTED	2445
				06	11	0096D	BRB	166\$	
				52	D4	0096F	CLRL	CONNECTED	2448
	FF	AB		08	88	00971	BISB2	#8, -1(CCB)	2449
				00	16	00975	JSB	BAS\$\$CB_POP	2452
		5B		54	D0	0097B	MOVL	OUR_CCB, CCB	2453
		0B		52	E9	0097E	BLBC	CONNECTED, 167\$	2455
		7E		8F	9A	00981	MOVZBL	#BAS\$K_INVFILOPT, -(SP)	
00000000G		00		01	FB	00985	CALLS	#1, BAS\$\$STOP_IO	
				53	D5	0098C	TSTL	PARENT_IFI	2457
				0B	12	0098E	BNEQ	168\$	
		7E		8F	9A	00990	MOVZBL	#BAS\$K_IO_CHANOT, -(SP)	
00000000G		00		01	FB	00994	CALLS	#1, BAS\$\$STOP_IO	
		03		AE	D1	0099B	CMPL	PARENT_ORG, #3	2459
				0B	13	0099F	BEQL	169\$	
		7E		8F	9A	009A1	MOVZBL	#BAS\$K_FILATTNOT, -(SP)	
00000000G		00		01	FB	009A5	CALLS	#1, BAS\$\$STOP_IO	
		02		53	B0	009AC	MOVW	PARENT_IFI, 2(FAB)	2461
		36		AE	B0	009B0	MOVW	PARENT_MRS, 54(FAB)	2462
		1E		AE	90	009B5	MOVB	PARENT_RAT, 30(FAB)	2463
		04		59	90	009BA	MOVB	PARENT_RFM, @4(SP)	2464
		3E		57	90	009BE	MOVB	PARENT_BKS, 62(FAB)	2465
		3C		55	B0	009C2	MOVW	PARENT_BLS, 60(FAB)	2466
50	04	AC		28	C1	009C6	ADDL3	#40, OPEN_ARG_BLK, R0	2478
				60	D5	009CB	TSTL	(R0)	

				3E	13	009CD	BEQL	173\$		
	A1	AB		04	88	009CF	BISB2	#4, -95(CCB)	2481	
	20	AE	C6	AB	32	009D3	CVTWL	-58(CCB), 32(SP)	2482	
			20	AE	9F	009D8	PUSHAB	32(SP)		
			0840	8F	BB	009DB	PUSHR	#*M<R6,R11>		
52	04	AC		28	C1	009DF	ADDL3	#40, OPEN_ARG_BLK, R2		
		92		03	FB	009E4	CALLS	#3, @R2		
		59		50	D0	009E7	MOVL	R0, OPEN_STATUS		
		57		01	D0	009EA	MOVL	#1, CONNECT_STATUS	2483	
		1B		59	E8	009ED	BLBS	OPEN_STATUS, 172\$	2485	
		17	08	A6	E9	009F0	BLBC	8(FAB), 172\$	2492	
		0A	08	AB	E8	009F4	BLBS	8(CCB), 171\$	2499	
		59	08	A6	D0	009F8	MOVL	8(FAB), OPEN_STATUS	2505	
		57	08	AB	D0	009FC	MOVL	8(CCB), CONNECT_STATUS	2506	
				09	11	00A00	BRB	172\$	2499	
	00000000G	00		59	DD	00A02	PUSHL	OPEN_STATUS	2512	
				01	FB	00A04	CALLS	#1, [IB\$STOP		
OB	FF	AB		77	11	00A0B	BRB	178\$	2492	
				02	E1	00A0D	BBC	#2, -1(CCB), 174\$	2527	
	00000000G	00		56	DD	00A12	PUSHL	FAB	2529	
				01	FB	00A14	CALLS	#1, SYSS\$DISPLAY		
				55	11	00A1B	BRB	177\$		
	00000000G	00		56	DD	00A1D	PUSHL	FAB	2539	
				01	FB	00A1F	CALLS	#1, SYSS\$PARSE		
				50	D0	00A26	MOVL	R0, OPEN_STATUS		
28	40	A6		59	E9	00A29	BLBC	OPEN_STATUS, 175\$	2541	
		24		02	E1	00A2C	BBC	#2, 84(FAB), 175\$	2545	
			34	AE	E9	00A31	BLBC	52(SP), 175\$	2546	
			24	AE	D5	00A35	TSTL	36(SP)	2547	
			10	1F	12	00A38	BNEQ	175\$		
				AE	D5	00A3A	TSTL	16(SP)	2548	
				1A	12	00A3D	BNEQ	175\$		
	1E	A6		02	8A	00A3F	BICB2	#2, 30(FAB)	2554	
	1E	A6		04	88	00A43	BISB2	#4, 30(FAB)	2555	
	04	BE		03	90	00A47	MOVB	#3, @4(SP)	2559	
	3F	A6		02	90	00A4B	MOVB	#2, 63(FAB)	2560	
	2C	AB	DA	AB	9E	00A4F	MOVAB	-38(R11), 44(CCB)	2564	
	FE	AB	40	8F	88	00A54	BISB2	#64, -2(CCB)	2568	
OB	FC	AB		03	E1	00A59	BBC	#3, -4(CCB), 176\$	2573	
				56	DD	00A5E	PUSHL	FAB	2575	
	00000000G	00		01	FB	00A60	CALLS	#1, SYSS\$OPEN		
				09	11	00A67	BRB	177\$		
				56	DD	00A69	PUSHL	FAB	2577	
	00000000G	00		01	FB	00A6B	CALLS	#1, SYSS\$CREATE		
				50	D0	00A72	MOVL	R0, OPEN_STATUS		
		0C		59	E9	00A75	BLBC	OPEN_STATUS, 178\$	2580	
				5B	DD	00A78	PUSHL	CCB		
	00000000G	00		01	FB	00A7A	CALLS	#1, SYSS\$CONNECT		
		57		50	D0	00A81	MOVL	R0, CONNECT_STATUS		
	20	AE	FE	AB	9E	00A84	MOVAB	-2(CCB), 32(SP)	2590	
OF	20	BE		0A	E0	00A89	BBS	#10, @32(SP), 179\$		
OE	07	A6		01	E1	00A8E	BBC	#1, 7(FAB), 180\$		
	00010619	8F	08	A6	D1	00A93	CMPL	8(FAB), #67097		
				04	13	00A9B	BEQL	180\$		
	FC	AB		08	88	00A9D	BISB2	#8, -4(CCB)	2592	
FO	AB	CD		06	28	00AA1	MOV3	#6, NAM_BLOCK+42, -16(CCB)	2598	
	D0	AB	02	A6	B0	00AA8	MOVW	2(FAB), -48(CCB)	2599	

		50	FF53	CD	9A	00AAD	MOVZBL	NAM_BLOCK+3, R0	2606
				08	13	00AB2	BEQL	181\$	
	F8	AB	FF54	CD	D0	00AB4	MOVL	NAM_BLOCK+4, -8(CCB)	2609
		50	FF5B	OD	11	00ABA	BRB	182\$	2610
				CD	9A	00ABC	MOVZBL	NAM_BLOCK+11, R0	2614
				0A	13	00AC1	BEQL	183\$	
	F8	AB	FF5C	CD	D0	00AC3	MOVL	NAM_BLOCK+12, -8(CCB)	2617
	F7	AB		50	90	00AC9	MOVB	R0, -9(CCB)	2618
		0A		59	E8	00ACD	BLBS	OPEN_STATUS, 184\$	2630
		7E		02	CE	00AD0	MNEGL	#2, -(SP)	
	00000000G	00		01	FB	00AD3	CALLS	#1, BASS\$STOP_IO	
	54	AE		03	D0	00ADA	MOVL	#3, UNWIND_ACTION	2636
		0A		57	E8	00ADE	BLBS	CONNECT_STATUS, 185\$	2638
		7E		03	CE	00AE1	MNEGL	#3, -(SP)	
	00000000G	00		01	FB	00AE4	CALLS	#1, BASS\$STOP_IO	
6D	20	BE		0A	E0	00AEB	BBS	#10, @32(SP), -190\$	2646
		52	40	A6	D0	00AF0	MOVL	64(FAB), R2	2650
3D		52		02	E1	00AF4	BBC	#2, R2, 188\$	
	20	BE		20	88	00AF8	BISB2	#32, @32(SP)	2657
			30	AB	9F	00AFC	PUSHAB	48(CCB)	2659
	20	AE	50	8F	9A	00AFF	MOVZBL	#80, 32(SP)	
			20	AE	9F	00B04	PUSHAB	32(SP)	
	00000000G	00		02	FB	00B07	CALLS	#2, LIB\$GET_VM	
		0B		50	E8	00B0E	BLBS	GET_VM_RESULT, 186\$	
		7E	00G	8F	9A	00B11	MOVZBL	#BASSK_MAXMEMEXC, -(SP)	2661
	00000000G	00		01	FB	00B15	CALLS	#1, BASS\$STOP_IO	
			34	AB	94	00B1C	CLRB	52(CCB)	2663
	07	AB	40	8F	88	00B1F	BISB2	#64, 7(CCB)	2664
			14	AE	D5	00B24	TSTL	20(SP)	2666
				05	12	00B27	BNEQ	187\$	
	D6	AB	3C	A6	B0	00B29	MOVW	60(FAB), -42(CCB)	2667
			D4	AB	B4	00B2E	CLRW	-44(CCB)	2669
	A1	AB		02	88	00B31	BISB2	#2, -95(CCB)	2670
24		52		01	E1	00B35	BBC	#1, R2, 190\$	2681
20		52		1B	E1	00B39	BBC	#27, R2, 190\$	2682
		1D		52	E9	00B3D	BLBC	R2, 190\$	2683
19		52		1A	E0	00B40	BBS	#26, R2, 190\$	2684
15		52		14	E0	00B44	BBS	#20, R2, 190\$	2685
11		52		02	E0	00B48	BBS	#2, R2, 190\$	2686
			14	AE	D5	00B4C	TSTL	20(SP)	2690
				05	12	00B4F	BNEQ	189\$	
	D6	AB	3C	A6	B0	00B51	MOVW	60(FAB), -42(CCB)	2691
			D4	AB	B4	00B56	CLRW	-44(CCB)	2693
	A1	AB		02	88	00B59	BISB2	#2, -95(CCB)	2694
	1F		2C	AE	E9	00B5D	BLBC	44(SP), 191\$	2706
50	04	AC		10	C1	00B61	ADDL3	#16, OPEN_ARG_BLK, R0	2707
				60	D5	00B66	TSTL	(R0)	
				16	13	00B68	BEQL	191\$	
50	04	AC		34	C1	00B6A	ADDL3	#52, OPEN_ARG_BLK, R0	2709
	14	AE		60	B1	00B6F	CMPL	(R0), 20(SP)	
		7E	00G	0B	1E	00B73	BGEQU	191\$	
	00000000G	00		8F	9A	00B75	MOVZBL	#BASSK_RECOVERMAP, -(SP)	2711
		00		01	FB	00B79	CALLS	#1, BASS\$STOP_IO	
03	FC	AB		03	E0	00B80	BBS	#3, -4(CCB), T92\$	2719
			0395	31	00B85	BRW	269\$		
		50	1D	A6	9E	00B88	MOVAB	29(FAB), R0	2736
05		00	08	AE	8F	00B8C	CASEB	8(SP), #0, #5	2729

0021	0018	0012 003B	000C 002D	00B91 00B99	193\$:	.WORD	194\$-193\$,- 195\$-193\$,- 196\$-193\$,- 198\$-193\$,- 199\$-193\$,- 201\$-193\$,-	
		C4 AB	04 90	00B9D	194\$:	MOVB	#4, -60(CCB)	2734
			0A 11	00BA1		BRB	197\$	2736
		C4 AB	05 90	00BA3	195\$:	MOVB	#5, -60(CCB)	2742
			04 11	00BA7		BRB	197\$	2744
		C4 AB	01 90	00BA9	196\$:	MOVB	#1, -60(CCB)	2750
		52	60 9A	00BAD	197\$:	MOVZBL	(R0), R2	2752
			16 11	00BB0		BRB	200\$	
		C4 AB	02 90	00BB2	198\$:	MOVB	#2, -60(CCB)	2758
		52	60 9A	00BB6		MOVZBL	(R0), R2	2760
		10	52 91	00BB9		CMPB	R2, #16	
			0A 11	00BBC		BRB	200\$	
		C4 AB	03 90	00BBE	199\$:	MOVB	#3, -60(CCB)	2766
		52	60 9A	00BC2		MOVZBL	(R0), R2	2768
		20	52 91	00BC5		CMPB	R2, #32	
			2E 13	00BC8	200\$:	BEQL	205\$	
		52	21 11	00BCA		BRB	204\$	
			60 9A	00BCC	201\$:	MOVZBL	(R0), R2	2779
		C4 AB	06 12	00BCF		BNEQ	202\$	2782
			01 90	00BD1		MOVB	#1, -60(CCB)	2783
		10	21 11	00BD5		BRB	205\$	
			52 91	00BD7	202\$:	CMPB	R2, #16	2785
			06 12	00BDA		BNEQ	203\$	
		C4 AB	02 90	00BDC		MOVB	#2, -60(CCB)	2786
		20	16 11	00BE0		BRB	205\$	
			52 91	00BE2	203\$:	CMPB	R2, #32	2788
			06 12	00BE5		BNEQ	204\$	
		C4 AB	03 90	00BE7		MOVB	#3, -60(CCB)	2789
		7E	0B 11	00BEB		BRB	205\$	
		00000000G	00G 8F	9A 00BED	204\$:	MOVZBL	#BASSK_FILATTNOT, -(SP)	2792
		50 04	01 FB	00BF1		CALLS	#1, BASS\$STOP_IO	
			1A C1	00BF8	205\$:	ADDL3	#26, OPEN_ARG_BLK, R0	2803
			60 B5	00BFD		TSTW	(R0)	
			1B 13	00BFF		BEQL	207\$	
		04	38 AE	E8 00C01		BLBS	56(SP), 206\$	2804
		13	30 AE	E9 00C05		BLBC	48(SP), 207\$	2805
SA	3E	A6	00	ED 00C09	206\$:	CMPZV	#0, #8, 62(FAB), BKS	2808
			0B 13	00C0F		BEQL	207\$	
		7E	00G 8F	9A 00C11		MOVZBL	#BASSK_FILATTNOT, -(SP)	
		00000000G	01 FB	00C15		CALLS	#1, BASS\$STOP_IO	
		50 04	0E C1	00C1C	207\$:	ADDL3	#14, OPEN_ARG_BLK, R0	2815
			60 B5	00C21		TSTW	(R0)	
			18 13	00C23		BEQL	208\$	
			52 D5	00C25		TSTL	R2	
			14 12	00C27		BNEQ	208\$	
		10	00	ED 00C29		CMPZV	#0, #16, 60(FAB), BLS	2818
			0B 13	00C30		BEQL	208\$	
		7E	00G 8F	9A 00C32		MOVZBL	#BASSK_FILATTNOT, -(SP)	
		00000000G	01 FB	00C36		CALLS	#1, BASS\$STOP_IO	
		52	D2 AB	9E 00C3D	208\$:	MOVAB	-46(CCB), R2	2824
		50	36 A6	3C 00C41		MOVZWL	54(FAB), R0	2825
		50	FF2E	CD B1	00C45	CMPW	XABFHC+10, R0	

				10	55	12	00D0B		BNEQ	227\$		
					AE	D5	00D0D		TSTL	16(SP)		2922
		03		04	50	12	00D10		BNEQ	227\$		
					BE	91	00D12		CMPB	24(SP), #3		2926
		7E			OB	13	00D16		BEQL	220\$		
	00000000G	00		00G	8F	9A	00D18		MOVZBL	#BAS\$K_FILATTNOT, -(SP)		
		02			01	FB	00D1C	220\$:	CALLS	#1, BAS\$\$STOP_IO		2928
				3F	A6	91	00D23		CMPB	63(FAB), #2		
		01			2C	11	00D27	221\$:	BRB	225\$		2934
				04	BE	91	00D29		CMPB	24(SP), #1		
		02			26	11	00D2D	222\$:	BRB	225\$		2938
				04	BE	91	00D2F		CMPB	24(SP), #2		
		03			2D	13	00D33		BEQL	227\$		
				04	BE	91	00D35		CMPB	24(SP), #3		
		03			1A	11	00D39	223\$:	BRB	225\$		2945
				04	BE	91	00D3B		CMPB	24(SP), #3		
		7E			OB	13	00D3F		BEQL	224\$		
	00000000G	00		00G	8F	9A	00D41		MOVZBL	#BAS\$K_FILATTNOT, -(SP)		
50		AC			01	FB	00D45	224\$:	CALLS	#1, BAS\$\$STOP_IO		2947
	04	60		3F	07	C1	00D4C		ADDL3	#7, OPEN_ARG_BLK, R0		
					A6	91	00D51	225\$:	CMPB	63(FAB), -(R0)		
		7E			OB	13	00D55	226\$:	BEQL	227\$		
	00000000G	00		00G	8F	9A	00D57		MOVZBL	#BAS\$K_FILATTNOT, -(SP)		
					01	FB	00D5B	227\$:	CALLS	#1, BAS\$\$STOP_IO		2959
003D	05	00		10	AE	8F	00D62	228\$:	CASEB	16(SP), #0, #5		
	0030	0037			000E		00D67		.WORD	229\$-228\$,-		
		0054		0044			00D6F			233\$-228\$,-		
										232\$-228\$,-		
										234\$-228\$,-		
										235\$-228\$,-		
										237\$-228\$		
										236\$		3017
					3B	11	00D73	229\$:	BRB	#2, 64(FAB), 230\$		2967
	0E	40	A6		02	E1	00D75		BBC	52(SP), 230\$		2968
			0A		AE	E9	00D7A		BLBC	36(SP)		2969
				34	AE	D5	00D7E		TSTL	230\$		
				24	AE	D5	00D81		BNEQ	16(SP)		2970
					05	12	00D81		TSTL	235\$		
				10	AE	D5	00D83		BEQL	40(SP), 232\$		2982
			OB		23	13	00D86	230\$:	BLBC	30(FAB), 236\$		2986
		20		28	AE	E9	00D88		BLBS	#2, 30(FAB), 239\$		
		A6		1E	A6	E8	00D8C	231\$:	BBC	236\$		
	2F	1E	A6		02	E1	00D90		BRB	#1, 30(FAB), 237\$		2991
					19	11	00D95	232\$:	BBS	30(FAB), 237\$		2995
	1F	1E	A6		01	E0	00D97		BRB	236\$		
					12	11	00D9C	233\$:	BLBS	#1, 30(FAB), 236\$		3003
			19	1E	A6	E8	00D9E		BRB	#2, 30(FAB), 238\$		3009
					0C	11	00DA2	234\$:	MOVZBL	#BAS\$K_RECATTNOT, -(SP)		
	07	1E	A6		01	E0	00DA4		CALLS	#1, BAS\$\$STOP_IO		3024
					E1	11	00DA9	235\$:	BBC	#1, -95(CCB)		3030
	10	1E	A6		02	E0	00DAB	236\$:	BRB	48(SP), 241\$		
		7E		00G	8F	9A	00DB0		BISB2	281\$		
		00			01	FB	00DB4	237\$:	BRW	(AP), #2		
	04	1E	A6		02	E1	00DBB	238\$:	CMPB	240\$		
		A1	AB		01	88	00DC0	239\$:	BLSSU			
			03		AE	E8	00DC4	240\$:				
				30	01CD	31	00DC8	241\$:				
			02		6C	91	00DCB					
					F8	1F	00DCE					

		62	04	29	12	0100C	BNEQ	290\$		
			4C	A0	D0	0100E	MOVL	4(R0), (XAB_PTR)	3323	
	44	AE	4C	AE	9F	01012	PUSHAB	XABKEY	3324	
			44	8F	9A	01015	MOVZBL	#76, 68(SP)		
				AE	9F	0101A	PUSHAB	68(SP)		
	00000000G	00		02	FB	0101D	CALLS	#2, LIB\$FREE VM		
		53		50	D0	01024	MOVL	R0, FREE VM STATUS		
		D2		53	E8	01027	BLBS	FREE VM STATUS, 289\$	3326	
		7E	00G	8F	9A	0102A	MOVZBL	#BAS\$K_PROLOSSOR, -(SP)		
	00000000G	00		01	FB	0102E	CALLS	#1, BAS\$\$\$STOP_IO		
				C5	11	01035	BRB	289\$	3316	
		52	04	A0	9E	01037	MOVAB	4(R0), XAB_PTR	3330	
				BF	11	0103B	BRB	289\$	3312	
		57	EC	AB	9E	0103D	MOVAB	-20(CCB), R7	3352	
50	04	AC		10	C1	01041	ADDL3	#16, OPEN_ARG_BLK, R0	3339	
				60	D5	01046	TSTL	(R0)		
				39	12	01048	BNEQ	294\$		
			D2	AB	B5	0104A	TSTW	-46(CCB)	3350	
				0B	12	0104D	BNEQ	292\$		
		7E	00G	8F	9A	0104F	MOVZBL	#BAS\$K_BADRECVAL, -(SP)		
	00000000G	00		01	FB	01053	CALLS	#1, BAS\$\$\$STOP_IO	3352	
				57	DD	0105A	PUSHL	R7		
		44	D2	AB	3C	0105C	MOVZWL	-46(CCB), 68(SP)		
			44	AE	9F	01061	PUSHAB	68(SP)		
	00000000G	00		02	FB	01064	CALLS	#2, LIB\$GET VM		
		0B		50	E8	0106B	BLBS	GET VM_RESULT, 293\$	3354	
		7E	00G	8F	9A	0106E	MOVZBL	#BAS\$K_MAXMEMEXC, -(SP)		
	00000000G	00		01	FB	01072	CALLS	#1, BAS\$\$\$STOP_IO		
D2	AB	00		00	2C	01079	MOVCS	#0, (SP), #0, -46(CCB), @0(R7)	3360	
			00	B7		0107F				
				0E	11	01081	BRB	295\$	3339	
		50	04	10	C1	01083	ADDL3	#16, OPEN_ARG_BLK, R0	3364	
				60	D0	01088	MOVL	(R0), (R7)		
		20	BE	8000	8F	A8	BISW2	#32768, @32(SP)	3365	
		52		F8	AB	D0	MOVL	-8(CCB), OLD_ADDRESS	3380	
				F8	AB	9F	PUSHAB	-8(CCB)	3381	
				F7	AB	9A	MOVZBL	-9(CCB), 68(SP)		
		44	AE	44	AE	9F	PUSHAB	68(SP)		
	00000000G	00		02	FB	010A0	CALLS	#2, LIB\$GET VM		
		0B		50	E8	010A7	BLBS	GET VM_RESULT, 296\$	3383	
		7E	00G	8F	9A	010AA	MOVZBL	#BAS\$K_MAXMEMEXC, -(SP)		
	00000000G	00		01	FB	010AE	CALLS	#1, BAS\$\$\$STOP_IO		
		50	F7	AB	9A	010B5	MOVZBL	-9(CCB), R0	3385	
F8	BB	62		50	28	010B9	MOVCS	R0, (OLD_ADDRESS), @-8(CCB)		
		BE		01	88	010BE	BISB2	#1, @32(SP)	3386	
	20	AB		67	D0	010C2	MOVL	(R7), 36(CCB)	3391	
	24	AB	D2	AB	B0	010C6	MOVW	-46(CCB), 32(CCB)	3392	
	20	AB		67	D0	010CB	MOVL	(R7), -100(CCB)	3393	
	9C	AB		AB	D4	010CF	CLRL	-24(CCB)	3397	
			E8	AB	D4	010D2	CLRL	60(CCB)	3398	
			3C	01	90	010D5	MOVB	#1, -40(CCB)	3402	
	D8	AB		01	88	010D9	BISB2	#1, -4(CCB)	3403	
	FC	AB		00	E8	010DD	BLBS	BAS\$SL_XIT_LOCK, 297\$	3411	
		07	00000000G	00	FB	010E4	CALLS	#0, BAS\$DECL_EXITH		
	00000000G	00		00	16	010EB	JSB	BAS\$CB_POP	3417	
			00000000G	00	04	010F1	RET		3419	
				0000	010F2	298\$:	.WORD	Save nothing	1446	

BASSOPEN
1-113

1 6
16-Sep-1984 00:52:31
14-Sep-1984 11:55:24

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASOPEN.B32;91

Page 72
(10)

	50	08	AC	D0	010F4	MOVL	8(AP), R0	
	50	04	A0	D0	010F8	MOVL	4(R0), R0	
		FE10	C0	9F	010FC	PUSHAB	UNWIND_CCB	
		FE14	C0	9F	01100	PUSHAB	UNWIND_ACTION	
			02	DD	01104	PUSHL	#2	
			5E	DD	01106	PUSHL	SP	
	7E	04	AC	7D	01108	MOVQ	4(AP), -(SP)	
0000V	CF		03	FB	0110C	CALLS	#3, OPEN_HANDLER	
			04	01	1111	RET		

; Routine Size: 4370 bytes, Routine Base: _BAS\$CODE + 0000

; 2502 3420 1


```
: 2504 3421 1 ROUTINE OPEN_HANDLER (           ! Handle an UNWIND from OPEN
: 2505 3422 1     SIG,                         ! Signal vector
: 2506 3423 1     MECH,                       ! Mechanism vector
: 2507 3424 1     ENBL,                       ! Enable vector
: 2508 3425 1     ) =
: 2509 3426 1
: 2510 3427 1 ++
: 2511 3428 1 FUNCTIONAL DESCRIPTION:
: 2512 3429 1
: 2513 3430 1     If we are unwinding, do the indicated OPEN cleanup, either nothing, POP
: 2514 3431 1     the CCB, mark the CCB for deallocation and POP it, or RMS CLOSE the CCB
: 2515 3432 1     and POP it.
: 2516 3433 1
: 2517 3434 1 FORMAL PARAMETERS:
: 2518 3435 1
: 2519 3436 1     SIG.rl.a      A counted vector of parameters to LIB$SIGNAL/STOP
: 2520 3437 1     MECH.rl.a    A counted vector of info from CHF
: 2521 3438 1     ENBL.ra.a   A counted vector of ENABLE argument addresses.
: 2522 3439 1
: 2523 3440 1 IMPLICIT INPUTS:
: 2524 3441 1
: 2525 3442 1     NONE
: 2526 3443 1
: 2527 3444 1 IMPLICIT OUTPUTS:
: 2528 3445 1
: 2529 3446 1     NONE
: 2530 3447 1
: 2531 3448 1 COMPLETION CODES:
: 2532 3449 1
: 2533 3450 1     Always SS$_RESIGNAL, which is ignored when unwinding.
: 2534 3451 1
: 2535 3452 1 SIDE EFFECTS:
: 2536 3453 1
: 2537 3454 1     May RMS CLOSE or DISCONNECT the file, and may deallocate the CCB.
: 2538 3455 1
: 2539 3456 1 --
: 2540 3457 1
: 2541 3458 2 BEGIN
: 2542 3459 2
: 2543 3460 2 MAP
: 2544 3461 2     SIG : REF VECTOR,
: 2545 3462 2     MECH : REF VECTOR,
: 2546 3463 2     ENBL : REF VECTOR;
: 2547 3464 2
: 2548 3465 2 LOCAL
: 2549 3466 2     MY_UNWIND_ACT : VOLATILE,
: 2550 3467 2     MY_UNWIND_CCB : VOLATILE;
: 2551 3468 2
: 2552 3469 2 GLOBAL REGISTER
: 2553 3470 2     CCB = K_CCB_REG : REF BLOCK [, BYTE];
: 2554 3471 2
: 2555 3472 2 !+
: 2556 3473 2 ! Define names for the two items in the ENABLE vector.
: 2557 3474 2 !-
: 2558 3475 2
: 2559 3476 2 BIND
: 2560 3477 2     UNWIND_ACTION = .ENBL [1],
```



```

2561      3478      2      UNWIND_CCB = .ENBL [2];
2562      3479      2
2563      3480      2      +
2564      3481      2      We are our own handler, in case the CLOSE or DISCONNECT fails.
2565      3482      2      -
2566      3483      2
2567      3484      2      ENABLE
2568      3485      2      OPEN_HANDLER (MY_UNWIND_ACT, MY_UNWIND_CCB);
2569      3486      2
2570      3487      2      +
2571      3488      2      Don't do anything yet.
2572      3489      2      -
2573      3490      2      MY_UNWIND_ACT = UNWIND_NOP;
2574      3491      2      +
2575      3492      2      Just resignal if this is an UNWIND or if the error is not severe. Otherwise
2576      3493      2      we clean up the I/O data base prior to signalling the SEVERE error.
2577      3494      2      -
2578      3495      2
2579      3496      2      IF (LIB$MATCH_COND (SIG [1], %REF (SS$_UNWIND)) OR (.BLOCK [SIG [1], STS$_SEVERITY] NEQ STS$_SEVERE))
2580      3497      2      THEN
2581      3498      2      RETURN (SS$_RESIGNAL);
2582      3499      2
2583      3500      2      +
2584      3501      2      Depending on the action selected, do things.
2585      3502      2      -
2586      3503      2      CCB = .UNWIND_CCB;
2587      3504      2
2588      3505      2      CASE .UNWIND_ACTION FROM UNWIND_MIN TO UNWIND_MAX OF
2589      3506      2      SET
2590      3507      2
2591      3508      2      [UNWIND_NOP] :      ! Do nothing.
2592      3509      2      BEGIN
2593      3510      2      0
2594      3511      2      END;
2595      3512      2
2596      3513      2      [UNWIND_POP] :      ! POP the specified CCB
2597      3514      2      BAS$$CB_POP ();
2598      3515      2
2599      3516      2      [UNWIND_DEALLOC] :      ! Mark the specified CCB for deallocation, then POP it
2600      3517      2      BEGIN
2601      3518      2      CCB [LIB$_DEALLOC] = 1;
2602      3519      2      BAS$$CB_POP ();
2603      3520      2      END;
2604      3521      2      +
2605      3522      2      RMS CLOSE or DISCONNECT the specified CCB (which marks it for deallocation), then POP it
2606      3523      2      (which will usually deallocate it).
2607      3524      2      -
2608      3525      2
2609      3526      2      [UNWIND_CLOSE] :
2610      3527      2      BEGIN
2611      3528      2      +
2612      3529      2      If the CLOSE fails, deallocate and POP.
2613      3530      2      -
2614      3531      2      MY_UNWIND_CCB = .CCB;
2615      3532      2      MY_UNWIND_ACT = UNWIND_DEALLOC;
2616      3533      2
2617      3534      2      IF ( NOT OTS$$CLOSE_FILE () ) THEN BAS$$STOP_IO (BAS$_IOERR_REC);

```


BASSOPEN
1-113

L 6
16-Sep-1984 00:52:31
14-Sep-1984 11:55:24

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASOPEN.B32;91

Page 75
(11)

```
: 2618      3535  3
: 2619      3536  3
: 2620      3537  2
: 2621      3538  2
: 2622      3539  2
: 2623      3540  2
: 2624      3541  1

      BASS$CB_POP ();
      END;
      TES;
      RETURN (SS$_RESIGNAL);
      END;
```

! end of OPEN_HANDLER

```
0804 00000 OPEN_HANDLER:
      52      0C      AC      D0      00002      .WORD      Save R2,R11
      6D      0065      CF      DE      00006      MOVL      ENBL, R2
      7E      0920      AE      D4      0000D      CLRQ      MY_UNWIND_CCB
      5B      04      AC      D0      00017      MOVAL      6$, (FP)
      04      04      AB      9F      0001B      CLRL      MY_UNWIND_ACT
      00000000G 00      02      FB      0001E      MOVZWL     #2336, -(SP)
      43      50      E8      00025      PUSHL      SP
      03      00      ED      00028      MOVL      SIG, R11
      5B      08      B2      D0      00030      PUSHAB     4(R11)
      00      04      B2      CF      00034      CALLS      #2, LIB$MATCH_COND
      0010      03      00      00      00032      BLBS      R0, 5$
      000A      002C      0032      00039      CMPZV     #0, #3, 4(R11), #4
      5B      08      B2      D0      00030      BNEQ      5$
      00      04      B2      CF      00034      MOVL      @8(R2), CCB
      0010      03      00      00      00032      CASEL     @4(R2), #0, #3
      000A      002C      0032      00039      .WORD      5$-1$, -
      5B      08      B2      D0      00030      4$-1$, -
      00      04      B2      CF      00034      2$-1$, -
      0010      03      00      00      00032      3$-1$
      000A      002C      0032      00039      5$
      FF      AB      10      88      00043      BRB      #16, -1(CCB)
      04      AE      1C      11      00047      BISB2     4$
      08      AE      5B      D0      00049      BRB      4$
      00000000G 00      02      D0      0004D      MOVL      CCB, MY_UNWIND_CCB
      00      0A      50      E8      00058      MOVL      #2, MY_UNWIND_ACT
      7E      01      CE      0005B      CALLS      #0, OT$$$CLOSE_FILE
      00000000G 00      01      FB      0005E      BLBS      R0, 4$
      50      00000000G 00      16      00065      MNEGL     #1, -(SP)
      0918      8F      3C      0006B      CALLS      #1, BASS$STOP_IO
      04      00070      RET      JSB      BASS$CB_POP
      50      08      AC      D0      00073      MOVZWL     #2328, R0
      04      04      A0      D0      00077      .WORD      Save nothing
      FC      A0      9F      0007B      MOVL      8(AP), R0
      7E      04      AC      D0      00077      MOVL      4(R0), R0
      FF72      CF      03      FB      00089      PUSHAB     MY_UNWIND_CCB
      04      04      A0      9F      0007E      PUSHAB     MY_UNWIND_ACT
      03      02      DD      00081      PUSHL      #2
      04      5E      DD      00083      PUSHL      SP
      7E      04      AC      7D      00085      MOVQ      4(AP), -(SP)
      CF      03      FB      00089      CALLS      #3, OPEN_HANDLER
      04      0008E      RET
```

; Routine Size: 143 bytes, Routine Base: _BASS\$CODE + 1112

BAS\$OPEN
1-113

M 6
16-Sep-1984 00:52:31
14-Sep-1984 11:55:24

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASOPEN.B32;91

Page 76
(11)

: 2625

3542 1


```
2627 3543 1 GLOBAL ROUTINE BAS$STATUS ! Status of last file opened
2628 3544 1 =
2629 3545 1
2630 3546 1 ++
2631 3547 1 FUNCTIONAL DESCRIPTION:
2632 3548 1
2633 3549 1 Get the status of the last file opened. The necessary bits
2634 3550 1 were saved by OPEN in L_STATUS.
2635 3551 1
2636 3552 1 FORMAL PARAMETERS:
2637 3553 1
2638 3554 1 NONE
2639 3555 1
2640 3556 1 IMPLICIT INPUTS:
2641 3557 1
2642 3558 1 L_STATUS A copy of FAB$L_DEV from the last OPEN, or 0
2643 3559 1
2644 3560 1 IMPLICIT OUTPUTS:
2645 3561 1
2646 3562 1 NONE
2647 3563 1
2648 3564 1 ROUTINE VALUE:
2649 3565 1 COMPLETION CODES:
2650 3566 1
2651 3567 1 Bits 0 through 4 reflect device characteristics, see below.
2652 3568 1
2653 3569 1 SIDE EFFECTS:
2654 3570 1
2655 3571 1 NONE
2656 3572 1
2657 3573 1 --
2658 3574 1
2659 3575 2 BEGIN
2660 3576 2 +
2661 3577 2 The following field describes the status bits returned.
2662 3578 2 -
2663 3579 2
2664 3580 2 FIELD
2665 3581 2 STATUS_BITS =
2666 3582 2 SET
2667 3583 2 STATUS_REC = [0, 0, 1, 0], ! Record-oriented device
2668 3584 2 STATUS_CCL = [0, 1, 1, 0], ! Carriage control device
2669 3585 2 STATUS_TRM = [0, 2, 1, 0], ! Device is a terminal
2670 3586 2 STATUS_DIR = [0, 3, 1, 0], ! Directory device (disk)
2671 3587 2 STATUS_SDI = [0, 4, 1, 0], ! Single-directory device
2672 3588 2 STATUS_SQD = [0, 5, 1, 0] ! Sequential, block-oriented device (magtape)
2673 3589 2 TES;
2674 3590 2
2675 3591 2 LOCAL
2676 3592 2 STATUS : BLOCK [2, BYTE] FIELD (STATUS_BITS);
2677 3593 2
2678 3594 2 +
2679 3595 2 Clear all of the bits in STATUS, then set the appropriate ones.
2680 3596 2 -
2681 3597 2 STATUS = 0;
2682 3598 2
2683 3599 2 IF ((L_STATUS AND DEV$M_REC) NEQ 0) THEN STATUS [STATUS_REC] = 1;
```



```

: 2684      3600 2
: 2685      3601 2
: 2686      3602 2
: 2687      3603 2
: 2688      3604 2
: 2689      3605 2
: 2690      3606 2
: 2691      3607 2
: 2692      3608 2
: 2693      3609 2
: 2694      3610 2
: 2695      3611 2
: 2696      3612 2
: 2697      3613 2
: 2698      3614 2
: 2699      3615 1

```

```

IF ((.L_STATUS AND DEV$M_CCL) NEQ 0) THEN STATUS [STATUS_CCL] = 1;
IF ((.L_STATUS AND DEV$M_TRM) NEQ 0) THEN STATUS [STATUS_TRM] = 1;
IF ((.L_STATUS AND DEV$M_DIR) NEQ 0) THEN STATUS [STATUS_DIR] = 1;
IF ((.L_STATUS AND DEV$M_SDI) NEQ 0) THEN STATUS [STATUS_SDI] = 1;
IF ((.L_STATUS AND DEV$M_SQD) NEQ 0) THEN STATUS [STATUS_SQD] = 1;

!+
Return the bits as our value.
!-
RETURN (.STATUS);
END;

```

! of routine BAS\$STATUS

		0000 00000	.ENTRY BAS\$STATUS, Save nothing	: 3543
		51 B4 00002	CLRW STATUS	: 3597
	50 00000000'	EF D0 00004	MOVL L STATUS, R0	: 3599
	03	50 E9 0000B	BLBC R0, 1\$	
	51	01 88 0000E	BISB2 #1, STATUS	
03	50	01 E1 00011 1\$:	BBC #1, R0, 2\$	: 3601
	51	02 88 00015	BISB2 #2, STATUS	
03	50	02 E1 00018 2\$:	BBC #2, R0, 3\$	: 3603
	51	04 88 0001C	BISB2 #4, STATUS	
03	50	03 E1 0001F 3\$:	BBC #3, R0, 4\$	: 3605
	51	08 88 00023	BISB2 #8, STATUS	
03	50	04 E1 00026 4\$:	BBC #4, R0, 5\$	: 3607
	51	10 88 0002A	BISB2 #16, STATUS	
03	50	05 E1 0002D 5\$:	BBC #5, R0, 6\$	: 3609
	51	20 88 00031	BISB2 #32, STATUS	
	50	51 3C 00034 6\$:	MOVZWL STATUS, R0	: 3614
		04 00037	RET	: 3615

: Routine Size: 56 bytes, Routine Base: _BAS\$CODE + 11A1

: 2700 3616 1


```
: 2702      3617 1 GLOBAL ROUTINE BAS$$STATU_INIT : NOVALUE =      ! Initialize status
: 2703      3618 1
: 2704      3619 1 ++
: 2705      3620 1 FUNCTIONAL DESCRIPTION:
: 2706      3621 1
: 2707      3622 1      Initialize the STATUS variable. This is needed by the RUN command in case this
: 2708      3623 1      is not the first RUN command in this image.
: 2709      3624 1
: 2710      3625 1 FORMAL PARAMETERS:
: 2711      3626 1
: 2712      3627 1      NONE
: 2713      3628 1
: 2714      3629 1 IMPLICIT INPUTS:
: 2715      3630 1
: 2716      3631 1      NONE
: 2717      3632 1
: 2718      3633 1 IMPLICIT OUTPUTS:
: 2719      3634 1
: 2720      3635 1      L_STATUS, always zet to zero.
: 2721      3636 1
: 2722      3637 1 ROUTINE VALUE:
: 2723      3638 1 COMPLETION CODES:
: 2724      3639 1
: 2725      3640 1      NONE
: 2726      3641 1
: 2727      3642 1 SIDE EFFECTS:
: 2728      3643 1
: 2729      3644 1      NONE
: 2730      3645 1
: 2731      3646 1 --
: 2732      3647 1
: 2733      3648 2 BEGIN
: 2734      3649 2 L_STATUS = 0;
: 2735      3650 1 END;
```

! of routine BAS\$\$STATU_INIT

```
00000000' EF 0000 00000
                D4 00002
                04 00008
```

```
.ENTRY BAS$$STATU_INIT, Save nothing
CLRL L_STATUS
RET
```

```
: 3617
: 3649
: 3650
```

: Routine Size: 9 bytes, Routine Base: _BAS\$CODE + 11D9

```
: 2736      3651 1
: 2737      3652 1 END
: 2738      3653 1
: 2739      3654 0 ELUDOM
```

! end of module BAS\$OPEN

PSECT SUMMARY

BAS\$OPEN
1-113

D 7
16-Sep-1984 00:52:31
14-Sep-1984 11:55:24

VAX-11 Bliss-32 V4.0-742
[BASRTL.SRC]BASOPEN.B32;91

Page 80
(13)

Name	Bytes	Attributes
:_BAS\$DATA	4	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
:_BAS\$CODE	4578	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
:_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	188	1	581	00:01.2

COMMAND QUALIFIERS

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

: Size: 4578 code + 4 data bytes
: Run Time: 01:55.7
: Elapsed Time: 04:15.6
: Lines/CPU Min: 1894
: Lexemes/CPU-Min: 18938
: Memory Used: 1272 pages
: Compilation Complete

0029 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

BASOPEN
LIS

BASPOWJ
LIS

BASPOS
LIS

BASPOWJ
LIS

BASOPENDE
LIS

BASPOWGG
LIS

BASPOWH
LIS

BASPOWRJ
LIS

BASPOWII
LIS

BASPURIOB
LIS

BASPOWDD
LIS

BASOPENZE
LIS

BASPOWR
LIS

BASPOWJ
LIS

BASPOWR
LIS

BASPOWH
LIS

BASPOWR
LIS