

```

BBBBBBBBBBBBBB      AAAAAAAAAA      CCCCCCCCCCCCCC      KKK      KKK      UUU      UUU      PPPPPPPPPPPP
BBBBBBBBBBBBBBBB      AAAAAAAAAA      CCCCCCCCCCCCCC      KKK      KKK      UUU      UUU      PPPPPPPPPPPP
BBBBBBBBBBBBBBBB      AAAAAAAAAA      CCCCCCCCCCCCCC      KKK      KKK      UUU      UUU      PPPPPPPPPPPP
BBB      BBB      AAA      AAA      CCC      KKK      KKK      UUU      UUU      PPP      PPP
BBB      BBB      AAA      AAA      CCC      KKK      KKK      UUU      UUU      PPP      PPP
BBB      BBB      AAA      AAA      CCC      KKK      KKK      UUU      UUU      PPP      PPP
BBB      BBB      AAA      AAA      CCC      KKK      KKK      UUU      UUU      PPP      PPP
BBB      BBB      AAA      AAA      CCC      KKK      KKK      UUU      UUU      PPP      PPP
BBB      BBB      AAA      AAA      CCC      KKK      KKK      UUU      UUU      PPP      PPP
BBB      BBB      AAA      AAA      CCC      KKK      KKK      UUU      UUU      PPP      PPP
BBBBBBBBBBBBBBBB      AAA      AAA      CCC      KKKKKKKKKK      UUU      UUU      PPPPPPPPPPPP
BBBBBBBBBBBBBBBB      AAA      AAA      CCC      KKKKKKKKKK      UUU      UUU      PPPPPPPPPPPP
BBBBBBBBBBBBBBBB      AAA      AAA      CCC      KKKKKKKKKK      UUU      UUU      PPPPPPPPPPPP
BBB      BBB      AAAAAAAAAAAAAAAAAA      CCC      KKK      KKK      UUU      UUU      PPP
BBB      BBB      AAAAAAAAAAAAAAAAAA      CCC      KKK      KKK      UUU      UUU      PPP
BBB      BBB      AAAAAAAAAAAAAAAAAA      CCC      KKK      KKK      UUU      UUU      PPP
BBB      BBB      AAA      AAA      CCC      KKK      KKK      UUU      UUU      PPP
BBB      BBB      AAA      AAA      CCC      KKK      KKK      UUU      UUU      PPP
BBB      BBB      AAA      AAA      CCC      KKK      KKK      UUU      UUU      PPP
BBB      BBB      AAA      AAA      CCC      KKK      KKK      UUU      UUU      PPP
BBBBBBBBBBBBBBBB      AAA      AAA      CCCCCCCCCCCCCC      KKK      KKK      UUUUUUUUUUUUUUUU      PPP
BBBBBBBBBBBBBBBB      AAA      AAA      CCCCCCCCCCCCCC      KKK      KKK      UUUUUUUUUUUUUUUU      PPP
BBBBBBBBBBBBBBBB      AAA      AAA      CCCCCCCCCCCCCC      KKK      KKK      UUUUUUUUUUUUUUUU      PPP

```

```

FFFFFFFFF  AAAAAA  SSSSSSSS  TTTTTTTTT  SSSSSSSS  CCCCCCCC  AAAAAA  NN  NN
FFFFFFFFF  AAAAAA  SSSSSSSS  TTTTTTTTT  SSSSSSSS  CCCCCCCC  AAAAAA  NN  NN
FF         AA      AA  SS         TT         SS         CC         AA      AA  NN  NN
FF         AA      AA  SS         TT         SS         CC         AA      AA  NN  NN
FF         AA      AA  SS         TT         SS         CC         AA      AA  NNNN  NN
FF         AA      AA  SS         TT         SS         CC         AA      AA  NNNN  NN
FFFFFFFFF  AA      AA  SSSSSS  TT         SS         CC         AA      AA  NN  NN
FFFFFFFFF  AA      AA  SSSSSS  TT         SS         CC         AA      AA  NN  NN
FF         AAAAAAAAAA  SS         TT         SS         CC         AAAAAAAAAA  NN  NNNN
FF         AAAAAAAAAA  SS         TT         SS         CC         AAAAAAAAAA  NN  NNNN
FF         AA      AA  SS         TT         SS         CC         AA      AA  NN  NN
FF         AA      AA  SS         TT         SS         CC         AA      AA  NN  NN
FF         AA      AA  SSSSSSSS  TT         SSSSSSSS  CCCCCCCC  AA      AA  NN  NN
FF         AA      AA  SSSSSSSS  TT         SSSSSSSS  CCCCCCCC  AA      AA  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 MODULE FASTSCAN (XTITLE 'Fast file scan'
2 0002 0 IDENT = 'V04-000'
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1
7 0007 1 *****
8 0008 1 *
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 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:
33 0033 1 Backup/Restore
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1 This module contains the fast file scan routines.
37 0037 1
38 0038 1 ENVIRONMENT:
39 0039 1 VAX/VMS user mode.
40 0040 1 --
41 0041 1
42 0042 1 AUTHOR: M. Jack, CREATION DATE: 19-Nov-1980
43 0043 1
44 0044 1 MODIFIED BY:
45 0045 1
46 0046 1 V03-015 LMP0272 L. Mark Pilant, 6-Jul-1984 8:41
47 0047 1 Modify BACKUP to always use a full length FIB.
48 0048 1
49 0049 1 V03-014 LY0499 Larry Yetto 25-JUN-1984 12:36
50 0050 1 Modify the concealed root directory code to properly deal
51 0051 1 with [000000]
52 0052 1
53 0053 1 V03-013 ACG0381 Andrew C. Goldstein, 9-Dec-1983 19:43
54 0054 1 Fix byte count of initial MFD file name string
55 0055 1 (broken by ACG0366)
56 0056 1
57 0057 1 V03-012 ACG0366 Andrew C. Goldstein, 11-Oct-1983 18:55

```

```

58 0058 1 Fix rooted directory handling to track $PARSE changes
59 0059 1
60 0060 1 V03-011 ACG0325 Andrew C. Goldstein, 4-Apr-1983 15:51
61 0061 1 Fix header area length validation
62 0062 1
63 0063 1 V03-010 MLJ0105 Martin L. Jack, 14-Feb-1983 17:07
64 0064 1 Correct error in execute-only 'no such file' path.
65 0065 1
66 0066 1 V03-009 ACG0313 Andrew C. Goldstein, 12-Feb-1983 16:12
67 0067 1 Add routine subtitles
68 0068 1
69 0069 1 V03-008 MLJ0104 Martin L. Jack, 24-Jan-1983 18:51
70 0070 1 Correct access control bits used for index file under
71 0071 1 /IGNORE=INTERLOCK. Allow access to execute-only directories
72 0072 1 via ACP lookup. Change bootblock handling to allow copying RSX
73 0073 1 system disks. Tighten V03-006 to save scanned but non-selected
74 0074 1 directories only if the selection file specification is "*. *;*"
75 0075 1 and the directory terminator is "*".
76 0076 1
77 0077 1 V03-007 MLJ0101 Martin L. Jack, 13-Nov-1982 19:46
78 0078 1 Correct ODS-1 latest-version selection.
79 0079 1
80 0080 1 V03-006 MLJ48949 Martin L. Jack, 6-Sep-1982 17:17
81 0081 1 Save scanned but non-selected directories only if the selection
82 0082 1 file specification is "*. *;*". This saves space in cases where
83 0083 1 an incremental restore is meaningless anyway.
84 0084 1
85 0085 1 V03-005 MLJ0096 Martin L. Jack, 24-Aug-1982 15:50
86 0086 1 Correct /IMAGE file number conflict bug.
87 0087 1
88 0088 1 V03-004 MLJ48500 Martin L. Jack, 18-Aug-1982 10:23
89 0089 1 Correct bad test that caused scanned but non-selected
90 0090 1 directories to be copied in a disk to disk operation. Also
91 0091 1 add a test to avoid extra accesses to directories.
92 0092 1
93 0093 1 V03-003 MLJ0087 Martin L. Jack, 08-Apr-1982 16:45
94 0094 1 Do not truncate the index file for ODS-1 version 2 because it
95 0095 1 cannot be re-extended to multi-header dynamically. Fix setting
96 0096 1 of DIR_STATUS.
97 0097 1
98 0098 1 V03-002 MLJ0083 Martin L. Jack, 22-Mar-1982 13:25
99 0099 1 Inhibit saving of scanned but not selected directories under
100 0100 1 /INTERCHANGE, to save space on distribution media.
101 0101 1
102 0102 1 V03-001 MLJ0082 Martin L. Jack, 16-Mar-1982 18:42
103 0103 1 Initialize DIR STATUS before index file scan to avoid incorrect
104 0104 1 results from SELECT_INPUT_FILE.
105 0105 1
106 0106 1 V02-011 MLJ0078 Martin L. Jack, 10-Feb-1982 15:30
107 0107 1 Correct error in exclusion of aliased files introduced by
108 0108 1 V02-007.
109 0109 1
110 0110 1 V02-010 MLJ0077 Martin L. Jack, 8-Feb-1982 15:11
111 0111 1 Implement negative version numbers.
112 0112 1
113 0113 1 V02-009 MLJ0075 Martin L. Jack, 28-Jan-1982 20:16
114 0114 1 Implement DIR_VERLIM and VERLIMIT attributes to support version

```

```
115 0115 1 | Limit handling. Add FIB$V_NORECORD to file accesses to support
116 0116 1 | file expiration handling.
117 0117 1 |
118 0118 1 | V02-008 MLJ0063 Martin L. Jack, 22-Dec-1981 4:20
119 0119 1 | Support rooted directories.
120 0120 1 |
121 0121 1 | V02-007 MLJ0062 Martin L. Jack, 4-Dec-1981 14:33
122 0122 1 | Implement /INCREMENTAL.
123 0123 1 |
124 0124 1 | V02-006 MLJ0054 Martin L. Jack, 15-Oct-1981 19:26
125 0125 1 | Support segmented directory records. Combine routines
126 0126 1 | ODS1_VOLUME_ATTRIBUTES and ODS2_VOLUME_ATTRIBUTES. Remove
127 0127 1 | COM_IMP_NOBACK, as this must be computed on every file.
128 0128 1 | Implement /VOLUME. Implement /IGNORE=INTERLOCK. Integrate
129 0129 1 | GET_VM and FREE_VM jacket routines.
130 0130 1 |
131 0131 1 | V02-005 MLJ0037 Martin L. Jack, 29-Aug-1981 16:04
132 0132 1 | Correct file selection with /IMAGE.
133 0133 1 |
134 0134 1 | V02-004 MLJ0036 Martin L. Jack, 28-Aug-1981 17:55
135 0135 1 | Reimplement file scan.
136 0136 1 |
137 0137 1 | V02-003 MLJ0025 Martin L. Jack, 8-May-1981 11:25
138 0138 1 | Implement latest-version selection. Make /RECORD restartable.
139 0139 1 |
140 0140 1 | V02-002 MLJ0018 Martin L. Jack, 7-Apr-1981 21:08
141 0141 1 | Restore expanded string length in input NAM block.
142 0142 1 |
143 0143 1 | V02-001 MLJ0010 Martin L. Jack, 25-Mar-1981 16:33
144 0144 1 | Reorganize global storage. Reorganize so that index file
145 0145 1 | processing occurs on one volume at a time to minimize global
146 0146 1 | storage requirement. Incorporate "slow" file scan into fast
147 0147 1 | file scan so that BACKUP, not RMS, maintains directory context.
148 0148 1 | Add attributes check on directories so that spurious
149 0149 1 | "illegal format" errors are not produced.
150 0150 1 |
151 0151 1 | **
```

```

153 0152 1 REQUIRE 'SRCS:COMMON';
154 1258 1 LIBRARY 'SYSS$LIBRARY:LIB';
155 1259 1
156 1260 1
157 1261 1 FORWARD ROUTINE
158 1262 1 FAST_FILE_SCAN: NOVALUE,      ! Scan with index file
159 1263 1 SLOW_FILE_SCAN: NOVALUE,     ! Scan without index file
160 1264 1 READ_HOME_BLOCK: NOVALUE,  ! Read and check home block
161 1265 1 VERIFY_HEADER,               ! Check a file header
162 1266 1 PROCESS_FILE: NOVALUE,       ! Process one file
163 1267 1 DIR_SCAN: NOVALUE,         ! Driver for directory scan
164 1268 1 INIT_DIR_SCAN: NOVALUE,    ! Initialize directory scan
165 1269 1 RESET_DIR_SPEC: NOVALUE,   ! Reset selection filespec
166 1270 1 FIND_NEXT,                 ! Scan for next matching file
167 1271 1 FREE_DIR_DATA: NOVALUE;  ! Free directory scan context
168 1272 1
169 1273 1
170 1274 1 EXTERNAL ROUTINE
171 1275 1 LIB$EXTRACT_CONCEALED: ADDRESSING_MODE(GENERAL),
172 1276 1 ! Parse concealed device/root directory
173 1277 1 CHECKSUM,                    ! Compute file header checksum
174 1278 1 CHECKSUM2,                  ! Compute home block checksum
175 1279 1 GEN_FID_RECORD: NOVALUE,  ! Write FID record
176 1280 1 INIT_NAME_BLOCK: NOVALUE, ! Initialize extended name block fields
177 1281 1 VOLUME_ATTRIBUTES:         ! Write volume attribute record
178 1282 1 NOVALUE,
179 1283 1 SAVE_ONE_FILE,              ! Routine to copy one file
180 1284 1 FILE_ERROR: NOVALUE,      ! Signal file-related error
181 1285 1 INIT_ATTR: NOVALUE,     ! Initialize attributes area
182 1286 1 LEFT_ONE,                ! Find leftmost one bit in a word
183 1287 1 MAKE_STRING,           ! Convert ODS-1 filename to ASCII
184 1288 1 INIT_SEL_INFO: NOVALUE, ! Initialize selection information
185 1289 1 MATCH_DIRECTORY,       ! Match directory specification
186 1290 1 MATCH_FILENAME,       ! Match file name, type, and version
187 1291 1 SELECT_INPUT_FILE,     ! Select input file based on qualifiers
188 1292 1 TERMINATE_SCAN,        ! Test termination of directory scan
189 1293 1 FREE_VM: NOVALUE,       ! Free virtual memory
190 1294 1 GET_VM,                ! Get virtual memory
191 1295 1 GET_ZERO_VM,          ! Get and zero virtual memory
192 1296 1 LIB$SIGNAL: ADDRESSING_MODE(GENERAL);
193 1297 1 ! Signal a condition
194 1298 1
195 1299 1
196 1300 1 EXTERNAL LITERAL
197 1301 1 BACKUP$MAXVOLS,
198 1302 1 BACKUP$PROCINDEX,
199 1303 1 BACKUP$OPENDIR,
200 1304 1 BACKUP$OPENIN,
201 1305 1 BACKUP$READDIR,
202 1306 1 BACKUP$BADDIR,
203 1307 1 BACKUP$NOSUCHRVN,
204 1308 1 LIB$_INVFILSPE;
205 1309 1
206 1310 1
207 1311 1 G$DEFINE(); ! Define global common area
208 1312 1
209 1313 1

```

```

: 210      1314 1 LITERAL
: 211      1315 1
: 212      1316 1      MAX_VOLUMES= 255,      ! Largest volume set
: 213      1317 1      DIR_BUF_COUNT= 16,      ! Size of directory buffer
: 214      1318 1      INDEX_BUF_COUNT=64;      ! Size of index file buffer
: 215      1319 1
: 216      1320 1 BIND
: 217      1321 1      MFD=          UPLIT BYTE (%ASCIC '000000');
: 218      1322 1
: 219      1323 1
: 220      1324 1 BUILTIN
: 221      1325 1      TESTBITCC,
: 222      1326 1      TESTBITSC,
: 223      1327 1      CALLG,
: 224      1328 1      ROT;

```

```

: 226 1329 1 %SBTTL 'FAST FILE SCAN - fast file scan main routine'
: 227 1330 1 GLOBAL ROUTINE FAST_FILE_SCAN: NOVALUE=
: 228 1331 1
: 229 1332 1 !++
: 230 1333 1
: 231 1334 1 FUNCTIONAL DESCRIPTION:
: 232 1335 1 This routine is the driver for the fast file scan.
: 233 1336 1
: 234 1337 1 INPUT PARAMETERS:
: 235 1338 1 NONE
: 236 1339 1
: 237 1340 1 IMPLICIT INPUTS:
: 238 1341 1 INPUT_FAB - Pointer to current input FAB.
: 239 1342 1 INPUT_NAM - Pointer to current input NAM block.
: 240 1343 1
: 241 1344 1 OUTPUT PARAMETERS:
: 242 1345 1 NONE
: 243 1346 1
: 244 1347 1 IMPLICIT OUTPUTS:
: 245 1348 1 NONE
: 246 1349 1
: 247 1350 1 ROUTINE VALUE:
: 248 1351 1 NONE
: 249 1352 1
: 250 1353 1 SIDE EFFECTS:
: 251 1354 1 NONE
: 252 1355 1
: 253 1356 1 !--
: 254 1357 1
: 255 1358 2 BEGIN
: 256 1359 2 LOCAL
: 257 1360 2 RVN, ! Relative volume number
: 258 1361 2 FIB: BBLOCK[FIB$C_LENGTH], ! FIB
: 259 1362 2 FIB_DESC: VECTOR[2], ! Descriptor for FIB
: 260 1363 2 STATUS, ! Status variable
: 261 1364 2 IOSB: VECTOR[4,WORD]; ! I/O status block
: 262 1365 2
: 263 1366 2
: 264 1367 2 ! Initialize FIB descriptor.
: 265 1368 2
: 266 1369 2 FIB_DESC[0] = FIB$C_LENGTH;
: 267 1370 2 FIB_DESC[1] = FIB;
: 268 1371 2
: 269 1372 2
: 270 1373 2 ! Allocate the general buffer.
: 271 1374 2
: 272 1375 2 COM_FLAGS[COM_DSBL_CHKPT] = TRUE;
: 273 1376 2 FAST_BUFFER_SIZE = 512 * INDEX_BUF_COUNT;
: 274 1377 2 FAST_BUFFER = GET_VM(512 * INDEX_BUF_COUNT);
: 275 1378 2
: 276 1379 2
: 277 1380 2 ! Loop over all volumes in the volume set.
: 278 1381 2
: 279 1382 2 RVN = 1;
: 280 1383 2 DO
: 281 1384 3 BEGIN
: 282 1385 3 LOCAL

```



```

: 283 1386 3      ATR_DESC:      BBLOCK[12],      : Attribute list
: 284 1387 3      STATBLK:      BBLOCK[8],      : Statistics block
: 285 1388 3      ACCTL,      : FIBSM_WRITE or 0
: 286 1389 3      EOF,      : Index file EOF position
: 287 1390 3      CLUSTER,      : Cluster factor
: 288 1391 3      BITMAP_OFFSET,      : Index file bitmap VBN offset
: 289 1392 3      VBN;      : Current index file VBN
: 290 1393 3
: 291 1394 3
: 292 1395 3      : Access the index file on RVN 1.
: 293 1396 3
: 294 1397 3      CH$FILL (0, FIB$C LENGTH, FIB);
: 295 1398 3      ACCTL = FIBSM_WRITE OR FIBSM_NORECORD;
: 296 1399 3      FIB[FIB$L_ACCTL] = .ACCTL;
: 297 1400 3      FIB[FIB$W_FID_NUM] = FID$C_INDEXF;
: 298 1401 3      FIB[FIB$W_FID_SEQ] = FID$C_INDEXF;
: 299 1402 3      FIB[FIB$W_FID_RVN] = .RVN;
: 300 1403 3      ATR_DESC[0,0,16,0] = 8;
: 301 1404 3      ATR_DESC[2,0,16,0] = ATR$C_STATBLK;
: 302 1405 3      ATR_DESC[4,0,32,0] = STATBLK;
: 303 1406 3      ATR_DESC[8,0,32,0] = 0;
: 304 1407 3      STATUS = $QIOQ(
: 305 P 1408 3      FUNC=IOS_ACCESS OR IOSM_ACCESS,
: 306 P 1409 3      CHAN=.INPUT_CHAN,
: 307 P 1410 3      IOSB=IOSB,
: 308 P 1411 3      P1=FIB_DESC,
: 309 1412 3      P5=ATR_DESC);
: 310 1413 3      IF .STATUS THEN STATUS = .IOSB[0];
: 311 1414 3      IF .STATUS EQL $$$_WRITLCK
: 312 1415 3      THEN
: 313 1416 4          BEGIN
: 314 1417 4              CH$FILL (0, FIB$C LENGTH, FIB);
: 315 1418 4              ACCTL = FIBSM_NORECORD;
: 316 1419 4              FIB[FIB$L_ACCTL] = FIBSM_NORECORD;
: 317 1420 4              FIB[FIB$W_FID_NUM] = FID$C_INDEXF;
: 318 1421 4              FIB[FIB$W_FID_SEQ] = FID$C_INDEXF;
: 319 1422 4              FIB[FIB$W_FID_RVN] = .RVN;
: 320 P 1423 4              STATUS = $QIOQ(
: 321 P 1424 4                  FUNC=IOS_ACCESS OR IOSM_ACCESS,
: 322 P 1425 4                  CHAN=.INPUT_CHAN,
: 323 P 1426 4                  IOSB=IOSB,
: 324 P 1427 4                  P1=FIB_DESC,
: 325 1428 4                  P5=ATR_DESC);
: 326 1429 4              IF .STATUS THEN STATUS = .IOSB[0];
: 327 1430 4              END;
: 328 1431 3      IF NOT .STATUS
: 329 1432 3      THEN
: 330 1433 3          SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
: 331 1434 3
: 332 1435 3
: 333 1436 3      : Read the home block. If RVN 1, establish the size of the volume set
: 334 1437 3      : and the structure level.
: 335 1438 3
: 336 1439 3      READ_HOMEBLOCK(.RVN, .FAST_BUFFER);
: 337 1440 3
: 338 1441 3
: 339 1442 3      : Allocate the per-volume data. Each variable between FAST_VOL_BEG and

```

```
340 1443 3      ! FAST_VOL_END is a pointer to a vector that varies with the number of
341 1444 3      ! volumes in the volume set.
342 1445 3      !
343 1446 3      IF .RVN EQL 1
344 1447 3      THEN
345 1448 3      BEGIN
346 1449 3      INCR A FROM FAST_VOL_BEG TO FAST_VOL_END-%UPVAL BY %UPVAL DO
347 1450 3      BEGIN
348 1451 3      .A = GET_ZERO_VM(.COM_I_SETCOUNT*%UPVAL);
349 1452 3      END;
350 1453 3      END;
351 1454 3      !
352 1455 3      ! Initialize information from the home block.
353 1456 3      !
354 1457 3      IF .FAST_STRUCLEV EQL 2
355 1458 3      THEN
356 1459 3      BEGIN
357 1460 3      CLUSTER = .FAST_BUFFER[HM2$W_CLUSTER];
358 1461 3      FAST_IMAP_SIZE[.RVN-1] = .FAST_BUFFER[HM2$W_IBMAPSIZE];
359 1462 3      BITMAP_OFFSET = .CLUSTER*4 + 1;
360 1463 3      FAST_HDR_OFFSET[.RVN-1] = .CLUSTER*4 + .FAST_BUFFER[HM2$W_IBMAPSIZE];
361 1464 3      END
362 1465 3      ELSE
363 1466 3      BEGIN
364 1467 3      CLUSTER = 1;
365 1468 3      FAST_IMAP_SIZE[.RVN-1] = .FAST_BUFFER[HM1$W_IBMAPSIZE];
366 1469 3      BITMAP_OFFSET = 3;
367 1470 3      FAST_HDR_OFFSET[.RVN-1] = 2 + .FAST_BUFFER[HM1$W_IBMAPSIZE];
368 1471 3      END;
369 1472 3      !
370 1473 3      ! Allocate memory for and read in index file bitmap.
371 1474 3      !
372 1475 3      FAST_IMAP[.RVN-1] = GET_VM(.FAST_IMAP_SIZE[.RVN-1] * 512);
373 1476 3      STATUS = $QIOW(
374 1477 3      FUNC=IOS_READVBLK,
375 1478 P 1478 3      CHAN=.INPUT_CHAN,
376 1479 P 1479 3      IOSB=IOSB,
377 1480 P 1480 3      P1=.FAST_IMAP[.RVN-1],
378 1481 P 1481 3      P2=.FAST_IMAP_SIZE[.RVN-1] * 512,
379 1482 P 1482 3      P3=.BITMAP_OFFSET);
380 1483 P 1483 3      IF .STATUS THEN STATUS = .IOSB[0];
381 1484 3      IF NOT .STATUS
382 1485 3      THEN
383 1486 3      SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
384 1487 3      !
385 1488 3      ! Read the index file header and get the EOF stored therein.
386 1489 3      !
387 1490 3      !
388 1491 3      STATUS = $QIOW(
389 1492 3      FUNC=IOS_READVBLK,
390 1493 P 1493 3      CHAN=.INPUT_CHAN,
391 1494 P 1494 3      IOSB=IOSB,
392 1495 P 1495 3      P1=.FAST_BUFFER + 512,
393 1496 P 1496 3      P2=512,
394 1497 P 1497 3      P3=.FAST_HDR_OFFSET[.RVN-1] + 1);
395 1498 P 1498 3
396 1499 3
```

```

: 397 1500 3 IF .STATUS THEN STATUS = .IOSB[0];
: 398 1501 3 IF NOT .STATUS
: 399 1502 3 THEN
: 400 1503 3     SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
: 401 1504 3
: 402 1505 3
: 403 1506 3 EOF = 0;
: 404 1507 3 IF .FAST_STRUCLEV EQL 2
: 405 1508 3     THEN EOF = ROT(.BBLOCK[BBLOCK[FAST_BUFFER + 512, FH2$W_RECATTR], FAT$L_EFBLK], 16) - 1
: 406 1509 3     ELSE IF .FAST_BUFFER[HM2$B_STRUCVER] EQL 2
: 407 1510 3         THEN EOF = ROT(.STATBLK[SBK$L_FILESIZE], 16);
: 408 1511 3
: 409 1512 3
: 410 1513 3 ! Now scan the volume's index file bitmap backwards, looking for the
: 411 1514 3 ! highest set bit. The maximum of the EOF mark and the highest set
: 412 1515 3 ! bit is taken to be the true index file EOF.
: 413 1516 3
: 414 1517 3 DECR J FROM .FAST_IMAP_SIZE[.RVN-1] * 128 - 1 TO 0 DO
: 415 1518 4     BEGIN
: 416 1519 4     IF .VECTOR[FAST_IMAP[.RVN-1], .J] NEQ 0
: 417 1520 4     THEN
: 418 1521 5         BEGIN
: 419 1522 5         EOF = MAXU(
: 420 1523 5             .J*32 +
: 421 1524 5             LEFT_ONE(.VECTOR[FAST_IMAP[.RVN-1], .J]) +
: 422 1525 5             .FAST_HDR_OFFSET[.RVN-1],
: 423 1526 5             .EOF);
: 424 1527 5         EXITLOOP;
: 425 1528 4         END;
: 426 1529 4     END;
: 427 1530 3
: 428 1531 3
: 429 1532 3 IF .QUAL[QUAL_IMAG]
: 430 1533 3 THEN
: 431 1534 4     BEGIN
: 432 1535 4
: 433 1536 4     ! Read the boot block.
: 434 1537 4     !
: 435 1538 4     STATUS = $QIOW(
: 436 1539 4     FUNC=IOS$ READVBLK,
: 437 1540 4     CHAN=.INPUT_CHAN,
: 438 1541 4     IOSB=IOSB,
: 439 1542 4     P1=.FAST_BUFFER + 512,
: 440 1543 4     P2=512,
: 441 1544 4     P3=1);
: 442 1545 4     IF .STATUS THEN STATUS = .IOSB[0];
: 443 1546 4     IF NOT .STATUS
: 444 1547 4     THEN
: 445 1548 4         SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
: 446 1549 4
: 447 1550 4
: 448 1551 4     ! If the volume is bootable, save the LBN from the boot block.
: 449 1552 4     !
: 450 1553 4     FAST_BOOT_LBN[.RVN-1] = -1;
: 451 1554 4     IF CHR$RCHR(.FAST_BUFFER + 512 + 5) EQL 0
: 452 1555 4     THEN
: 453 1556 4         FAST_BOOT_LBN[.RVN-1] = ROT(.VECTOR[FAST_BUFFER + 512, 1], 16);

```

```

: 454      1557      3      END;
: 455      1558
: 456      1559
: 457      1560      : Deaccess the index file.
: 458      1561      :
: 459      P 1562      $QIOW(
: 460      P 1563      FUNC=IOS DEACCESS,
: 461      1564      CHAN=.INPUT_CHAN);
: 462      1565
: 463      1566
: 464      1567      IF .QUAL[QUAL_IMAG]
: 465      1568      THEN
: 466      1569      BEGIN
: 467      1570
: 468      1571      : Access the bitmap file.
: 469      1572      :
: 470      1573      CH$FILL (0, FIB$C_LENGTH, FIB);
: 471      1574      FIB[FIB$L_ACCTL] = .ACCTL;
: 472      1575      FIB[FIB$W_FID_NUM] = FID$C_BITMAP;
: 473      1576      FIB[FIB$W_FID_SEQ] = FID$C_BITMAP;
: 474      1577      FIB[FIB$W_FID_RVN] = .RVN;
: 475      P 1578      STATUS = $QIOW(
: 476      P 1579      FUNC=IOS ACCESS OR IOSM_ACCESS,
: 477      P 1580      CHAN=.INPUT_CHAN,
: 478      P 1581      IOSB=IOSB,
: 479      1582      P1=FIB_DESC);
: 480      1583      IF .STATUS THEN STATUS = .IOSB[0];
: 481      1584      IF NOT .STATUS
: 482      1585      THEN
: 483      1586      SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
: 484      1587
: 485      1588
: 486      1589      : Read the storage control block.
: 487      1590      :
: 488      P 1591      STATUS = $QIOW(
: 489      P 1592      FUNC=IOS READVBLK,
: 490      P 1593      CHAN=.INPUT_CHAN,
: 491      P 1594      IOSB=IOSB,
: 492      P 1595      P1=.FAST_BUFFER + 1024,
: 493      P 1596      P2=512,
: 494      1597      P3=1);
: 495      1598      IF .STATUS THEN STATUS = .IOSB[0];
: 496      1599      IF NOT .STATUS
: 497      1600      THEN
: 498      1601      SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
: 499      1602
: 500      1603
: 501      1604      : Deaccess the bitmap file.
: 502      1605      :
: 503      P 1606      $QIOW(
: 504      P 1607      FUNC=IOS DEACCESS,
: 505      1608      CHAN=.INPUT_CHAN);
: 506      1609
: 507      1610
: 508      1611      : Generate the volume attribute record.
: 509      1612
: 510      1613      IF NOT .QUAL[QUAL_VOLU] OR .QUAL[QUAL_VOLU_VALUE] EQL .RVN

```

```

511 1614 4 THEN
512 1615 4     VOLUME_ATTRIBUTES(
513 1616 4     .FAST_BUFFER,           ! home block
514 1617 4     .FAST_BUFFER+512,       ! boot block
515 1618 4     .FAST_BUFFER+1024,     ! storage control block
516 1619 4     .EOF = .FAST_HDR_OFFSET[.RVN-1]);
517 1620 4 END;
518 1621 4
519 1622 4
520 1623 4 ! Access the index file.
521 1624 4 !
522 1625 4 CH$FILL (0, FIB$C_LENGTH, FIB);
523 1626 4 FIB[FIB$L_ACCTL] = .ACCTL;
524 1627 4 FIB[FIB$W_FID_NUM] = FID$C_INDEXF;
525 1628 4 FIB[FIB$W_FID_SEQ] = FID$C_INDEXF;
526 1629 4 FIB[FIB$W_FID_RVN] = .RVN;
527 1630 4 STATUS = $QIOW(
P 1631 4     FUNC=IOS$ ACCESS OR IOSM_ACCESS,
P 1632 4     CHAN=.INPUT_CHAN,
P 1633 4     IOSB=IOSB,
531 1634 4     P1=FIB_DESC);
532 1635 4 IF .STATUS THEN STATUS = .IOSB[0];
533 1636 4 IF NOT .STATUS
534 1637 4 THEN
535 1638 4     SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
536 1639 4
537 1640 4
538 1641 4 ! Loop for all headers in the index file. Read multiple blocks into a
539 1642 4 ! buffer and process them one at a time.
540 1643 4 !
541 1644 4 DIR_STATUS = 0; ! Used by SELECT_INPUT_FILE
542 1645 4 VBN = .FAST_HDR_OFFSET[.RVN-1] + 1;
543 1646 4 UNTIL .VBN GTRU .EOF DO
544 1647 4 BEGIN
545 1648 4 LOCAL
546 1649 4     HEADER: REF BBLOCK, ! Pointer to header
547 1650 4     READ_COUNT; ! Blocks to read on current iteration
548 1651 4
549 1652 4
550 1653 4 ! Establish the count of blocks to read and execute the read.
551 1654 4 !
552 1655 4 READ_COUNT = MINU(INDEX_BUF_COUNT, .EOF + 1 - .VBN);
553 1656 4 STATUS = $QIOW(
P 1657 4     FUNC=IOS$ READVBLK,
P 1658 4     CHAN=.INPUT_CHAN,
P 1659 4     IOSB=IOSB,
557 1660 4     P1=.FAST_BUFFER,
558 1661 4     P2=512 * .READ_COUNT,
559 1662 4     P3=.VBN);
560 1663 4 IF .STATUS THEN STATUS = .IOSB[0];
561 1664 4
562 1665 4
563 1666 4 ! If an error occurred, read each block separately, reporting any
564 1667 4 ! errors.
565 1668 4 !
566 1669 4 IF NOT .STATUS
567 1670 4 THEN

```

```

: 568      1671  5      BEGIN
: 569      1672  5      INCR XVBN FROM 0 TO .READ_COUNT-1 DO
: 570      1673  6      BEGIN
: 571      1674  6      LOCAL
: 572      1675  6      HEADER:      REF BBLOCK;      ! Pointer to header
: 573      1676  6
: 574      1677  6
: 575      1678  6      HEADER = .FAST_BUFFER + .XVBN * 512;
: 576      1679  6      STATUS = $QIOWT
: 577      1680  6      FUNC=IOS$ READVBLK,
: 578      1681  6      CHAN=.INPUT_CHAN,
: 579      1682  6      IOSB=IOSB,
: 580      1683  6      P1=.HEADER,
: 581      1684  6      P2=512,
: 582      1685  6      P3=.VBN + .XVBN);
: 583      1686  6      IF .STATUS THEN STATUS = .IOSB[0];
: 584      1687  6      IF NOT .STATUS
: 585      1688  6      THEN
: 586      1689  6      CH$FILL(0, 512, .HEADER);
: 587      1690  5      END;
: 588      1691  4      END;
: 589      1692  4
: 590      1693  4
: 591      1694  4      ! For each header, verify that it is a valid file header.
: 592      1695  4      ! If it is, process it.
: 593      1696  4
: 594      1697  4      !
: 595      1698  4      HEADER = .FAST_BUFFER;
: 596      1699  4      INCR XVBN FROM 0 TO .READ_COUNT-1 DO
: 597      1700  5      BEGIN
: 598      1701  5      LOCAL
: 599      1702  5      MAP_AREA:      REF BBLOCK,      ! Pointer to map area
: 600      1703  5      FILE_NUMBER,      ! Current file number
: 601      1704  5      FILE_ID:      BBLOCK[FID$C_LENGTH];      ! Current file ID
: 602      1705  5
: 603      1706  5      ! Get a clean file ID.
: 604      1707  5      !
: 605      1708  5      FILE_NUMBER = .VBN + .XVBN - .FAST_HDR_OFFSET[.RVN-1];
: 606      1709  5      FILE_ID[FID$W_NUM] = .FILE_NUMBER<0,16>;
: 607      1710  5      FILE_ID[FID$B_NMX] = .FILE_NUMBER<16,8>;
: 608      1711  5      IF .FAST_STRUC$LEV EQL 2
: 609      1712  5      THEN FILE_ID[FID$W_SEQ] = .HEADER[FH2$W_FID_SEQ]
: 610      1713  5      ELSE FILE_ID[FID$W_SEQ] = .HEADER[FH1$W_FID_SEQ];
: 611      1714  5      FILE_ID[FID$B_RVN] = .RVN;
: 612      1715  5
: 613      1716  5
: 614      1717  5      ! Generate the FID record for an image mode save.
: 615      1718  5      !
: 616      1719  5      IF
: 617      1720  5      .QUAL[QUAL_IMAG] AND
: 618      1721  5      .XVBN EQL 0 AND
: 619      1722  6      (NOT .QUAL[QUAL_VOLU] OR .QUAL[QUAL_VOLU_VALUE] EQL .RVN)
: 620      1723  5      THEN
: 621      1724  5      GEN_FID_RECORD(.HEADER, .READ_COUNT, .FILE_NUMBER, .RVN);
: 622      1725  5
: 623      1726  5
: 624      1727  5      ! Validate the header. In ODS-2, a valid header is to be taken as

```

```

: 625      1728  5      ! valid even if the index file bitmap shows it as available. In
: 626      1729  5      ! ODS-1, a header corresponding to a clear index file bitmap bit
: 627      1730  5      ! is not to be examined.
: 628      1731  5
: 629      1732  5      STATUS = VERIFY_HEADER(.HEADER, FILE_ID);
: 630      1733  5      IF
: 631      1734  5          .FAST_STRUCLEV EQL 1 AND
: 632      1735  5          NOT .BITVECTOR[FAST_IMAP[RVN-1], .FILE_NUMBER-1]
: 633      1736  5      THEN
: 634      1737  5          STATUS = FALSE;
: 635      1738  5
: 636      1739  5
: 637      1740  5      ! Clear the index file bitmap bit until we determine that the
: 638      1741  5      ! header is not an extension header.
: 639      1742  5
: 640      1743  5      BITVECTOR[FAST_IMAP[RVN-1], .FILE_NUMBER-1] = FALSE;
: 641      1744  5
: 642      1745  5
: 643      1746  5      IF .STATUS
: 644      1747  5      THEN
: 645      1748  6          BEGIN
: 646      1749  6              ! Header is valid.
: 647      1750  6              ! Other processing executed only if not an extension header.
: 648      1751  6
: 649      1752  6              MAP_AREA = .HEADER + .HEADER[FH1$B_MPOFFSET]*2;
: 650      1753  6              IF
: 651      1754  6                  BEGIN
: 652      1755  7                  IF .FAST_STRUCLEV EQL 2
: 653      1756  7                      THEN .HEADER[FH2$W_SEG_NUM] EQL 0
: 654      1757  7                      ELSE .MAP_AREA[FH1$B_EX_SEGNUM] EQL 0
: 655      1758  7                  END
: 656      1759  7              THEN
: 657      1760  6                  BEGIN
: 658      1761  7                      LOCAL
: 659      1762  7                          FCH:      REF BBLOCK;      ! Pointer to characteristics
: 660      1763  7
: 661      1764  7
: 662      1765  7
: 663      1766  7              ! Make sure the index file bitmap indicates that the header
: 664      1767  7              ! is valid.
: 665      1768  7
: 666      1769  7              BITVECTOR[FAST_IMAP[RVN-1], .FILE_NUMBER-1] = TRUE;
: 667      1770  7
: 668      1771  7
: 669      1772  7              ! Get file characteristics pointer.
: 670      1773  7
: 671      1774  7              IF .FAST_STRUCLEV EQL 2
: 672      1775  7                  THEN FCH = HEADER[FH2$L_FILECHAR]
: 673      1776  7                  ELSE FCH = HEADER[FH1$W_FILECHAR];
: 674      1777  7
: 675      1778  7
: 676      1779  7              ! Evaluate file selection criteria for header. However,
: 677      1780  7              ! always reject files marked for delete. If this is /IMAGE
: 678      1781  7              ! mode, select all other valid files.
: 679      1782  7
: 680      1783  7              IF
: 681      1784  8                  BEGIN

```

```

: 682      1785      8          IF .FCH[FCHSV_MARKDEL]
: 683      1786      8          THEN
: 684      1787      8              TRUE
: 685      1788      8          ELSE
: 686      1789      9              BEGIN
: 687      1790      9                  INIT_ATTR(.HEADER);
: 688      1791      9                  IF .QUAL[QUAL_IMAG]
: 689      1792      9                      THEN FALSE
: 690      1793      9                      ELSE NOT SELECT_INPUT_FILE(%B'001')
: 691      1794      9              END
: 692      1795      8          END
: 693      1796      7          THEN
: 694      1797      7              BITVECTOR[.FAST_IMAP[.RVN-1], .FILE_NUMBER-1] = FALSE;
: 695      1798      6          END;
: 696      1799      5          END;
: 697      1800      5
: 698      1801      5
: 699      1802      5          HEADER = .HEADER + 512;
: 700      1803      4          END;
: 701      1804      4
: 702      1805      4
: 703      1806      4          VBN = .VBN + INDEX_BUF_COUNT;
: 704      1807      3          END;
: 705      1808      3
: 706      1809      3
: 707      1810      3          ! Deaccess the index file.
: 708      1811      3          !
: 709      1812      3          $QIOW(
: 710      1813      3              FUNC=IOS$ DEACCESS,
: 711      1814      3              CHAN=.INPUT_CHAN);
: 712      1815      3
: 713      1816      3
: 714      1817      3          RVN = .RVN + 1;
: 715      1818      3          END
: 716      1819      2          WHILE .RVN LEQU .COM_I_SETCOUNT;
: 717      1820      2
: 718      1821      2
: 719      1822      2          ! Free the general buffer.
: 720      1823      2          !
: 721      1824      2          FREE_VM(.FAST_BUFFER_SIZE, .FAST_BUFFER);
: 722      1825      2          FAST_BUFFER = FAST_BUFFER_SIZE = 0;
: 723      1826      2          COM_FLAGS[COM_DSBL_CHKPT] = FALSE;
: 724      1827      2
: 725      1828      2
: 726      1829      2          ! Scan all directories on all volumes.
: 727      1830      2          !
: 728      1831      2          INCR RVN FROM 1 TO .COM_I_SETCOUNT DO DIR_SCAN(.RVN);
: 729      1832      2
: 730      1833      2
: 731      1834      2          ! Allocate a buffer to read file headers.
: 732      1835      2          !
: 733      1836      2          FAST_BUFFER_SIZE = 512;
: 734      1837      2          FAST_BUFFER = GET_VM(512);
: 735      1838      2
: 736      1839      2
: 737      1840      2          ! If this is /IMAGE mode, scan for and process lost files.
: 738      1841      2          !

```



```

739 1842 2 IF .QUAL[QUAL_IMAG]
740 1843 2 THEN
741 1844 3 BEGIN
742 1845 3 INCR RVN FROM 1 TO .COM_I_SETCOUNT DO
743 1846 4 BEGIN
744 1847 4
745 1848 4 ! Scan bitmap for files not yet processed and process these.
746 1849 4 !
747 1850 4 INCR FILE_NUMBER FROM 1 TO .FAST_IMAP_SIZE[RVN-1]*4096 DO
748 1851 5 BEGIN
749 1852 5 IF .BITVECTOR[FAST_IMAP[RVN-1], .FILE_NUMBER-1]
750 1853 5 THEN
751 1854 6 BEGIN
752 1855 6 LOCAL
753 1856 6 NAME_LENGTH, ! Length of filename
754 1857 6 NAME_ADDRESS, ! Address of filename
755 1858 6 RSA_DESC: VECTOR[2], ! Descriptor for RSA
756 1859 6 FILENAME: VECTOR[20, BYTE]; ! Filename buffer
757 1860 6
758 1861 6
759 1862 6 ! Access index file on current RVN.
760 1863 6 !
761 1864 6 CH$FILL (0, FIB$C_LENGTH, FIB);
762 1865 6 FIB[FIB$C_ACCTL] = FIB$M_NORECORD;
763 1866 6 FIB[FIB$W_FID_NUM] = FIB$C_INDEXF;
764 1867 6 FIB[FIB$W_FID_SEQ] = FIB$C_INDEXF;
765 1868 6 FIB[FIB$W_FID_RVN] = .RVN;
766 1869 6 STATUS = $QIOW(
767 P 1870 6 FUNC=IOS_ACCESS OR IOSM_ACCESS,
768 P 1871 6 CHAN=.INPUT_CHAN,
769 P 1872 6 IOSB=IOSB,
770 1873 6 P1=FIB_DESC);
771 1874 6 IF .STATUS THEN STATUS = .IOSB[0];
772 1875 6 IF NOT .STATUS
773 1876 6 THEN
774 1877 6 SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
775 1878 6
776 1879 6
777 1880 6 ! Read file header.
778 1881 6 !
779 P 1882 6 STATUS = $QIOW(
780 P 1883 6 FUNC=IOS_READVBLK,
781 P 1884 6 CHAN=.INPUT_CHAN,
782 P 1885 6 IOSB=IOSB,
783 P 1886 6 P1=.FAST_BUFFER,
784 P 1887 6 P2=512,
785 1888 6 P3=.FAST_HDR_OFFSET[RVN-1] + .FILE_NUMBER);
786 1889 6 IF .STATUS THEN STATUS = .IOSB[0];
787 1890 6 IF NOT .STATUS
788 1891 6 THEN
789 1892 6 SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
790 1893 6
791 1894 6
792 1895 6 ! Deaccess index file on current RVN.
793 1896 6 !
794 P 1897 6 $QIOW(
795 P 1898 6 FUNC=IOS_DEACCESS,

```

```

: 796
: 797
: 798
: 799
: 800
: 801
: 802
: 803
: 804
: 805
: 806
: 807
: 808
: 809
: 810
: 811
: 812
: 813
: 814
: 815
: 816
: 817
: 818
: 819
: 820
: 821
: 822
: 823
: 824
: 825
: 826
: 827
: 828
: 829
: 830
: 831
: 832
: 833
: 834
: 835
: 836
: 837
: 838
: 839
: 840
: 841
: 842
: 843
: 844
: 845
: 846
: 847
: 848
: 849
: 850
: 851
: 852

```

```

1899 6
1900 6
1901 6
1902 6
1903 6
1904 6
1905 6
1906 6
1907 6
1908 6
1909 6
1910 6
1911 6
1912 6
1913 6
1914 6
1915 6
1916 6
1917 7
1918 7
1919 7
1920 7
1921 7
1922 7
1923 7
1924 7
1925 7
1926 7
1927 7
1928 7
1929 6
1930 7
1931 7
1932 7
1933 7
1934 7
1935 7
1936 7
1937 6
1938 6
1939 6
1940 6
1941 6
1942 6
1943 6
1944 6
1945 6
1946 6
1947 6
1948 6
1949 6
1950 6
1951 6
1952 6
1953 5
1954 4
1955 3

```

```

        CHAN=.INPUT_CHAN);

: Set up file ID.
INPUT_NAM[NAM$W_FID_NUM] = .FILE_NUMBER;
INPUT_NAM[NAM$B_FID_NUM] = .FILE_NUMBER<16,B>;
INPUT_NAM[NAM$B_FID_RVN] = .RVN;
INPUT_NAM[NAM$W_DID_NUM] = 0;
INPUT_NAM[NAM$W_DID_SEQ] = 0;
INPUT_NAM[NAM$W_DID_RVN] = 0;

: Generate filename string from file header ident area.
: Also get file sequence number.
IF .FAST_STRUCLEV EQL 2
THEN
    BEGIN
        LOCAL
            P;

        INPUT_NAM[NAM$W_FID_SEQ] = .FAST_BUFFER[FH2$W_FID_SEQ];
        NAME_ADDRESS =
            BBLOCK[.FAST_BUFFER + .FAST_BUFFER[FH2$B_IDOFFSET]*2,
                F12$F_FILENAME];
        NAME_LENGTH = F12$S_FILENAME;
        P = CH$FIND_CH(F12$S_FILENAME, .NAME_ADDRESS, %C' ');
        IF .P NEQ 0 THEN NAME_LENGTH = .P - .NAME_ADDRESS;
    END
ELSE
    BEGIN
        INPUT_NAM[NAM$W_FID_SEQ] = .FAST_BUFFER[FH1$W_FID_SEQ];
        NAME_ADDRESS = FILENAME;
        NAME_LENGTH = MAKE_STRING(
            BBLOCK[.FAST_BUFFER + .FAST_BUFFER[FH1$B_IDOFFSET]*2,
                F11$W_FILENAME] - $BYTEOFFSET(NMBSW_NAME),
            FILENAME) AND 65535;
    END;
    RSA_DESC[0] = NAM$C_MAXRSS;
    RSA_DESC[1] = .INPUT_NAM[NAM$L_RSA];
    $FAO(
        $DESCRIPTOR('!AS[!AD'),
        RSA_DESC,
        RSA_DESC,
        INPUT_QUAL[QUAL_DEV_DESC],
        .NAME_LENGTH, .NAME_ADDRESS);
    INPUT_NAM[NAM$B_RSL] = .RSA_DESC[0];
    INIT_NAMEBLOCK(.INPUT_NAM);

: Call the routine to process the file.
: SAVE_ONE_FILE();
END;
END;
END;

```

```

: 853      1956      2      END;
: 854      1957
: 855      1958
: 856      1959      ! Free the file header buffer.
: 857      1960
: 858      1961      FREE_VM(.FAST_BUFFER_SIZE, .FAST_BUFFER);
: 859      1962      FAST_BUFFER = FAST_BUFFER_SIZE = 0;
: 860      1963
: 861      1964
: 862      1965      ! Free memory for index file bitmaps.
: 863      1966
: 864      1967      INCR RVN FROM 1 TO .COM_I_SETCOUNT DO
: 865      1968          BEGIN
: 866      1969              FREE_VM(.FAST_IMAP_SIZE[.RVN-1] * 512, .FAST_IMAP[.RVN-1]);
: 867      1970              FAST_IMAP[.RVN-1] = 0;
: 868      1971          END;
: 869      1972
: 870      1973
: 871      1974      ! Free the per-volume data.
: 872      1975
: 873      1976      INCRA A FROM FAST_VOL_BEG TO FAST_VOL_END-%UPVAL BY %UPVAL DO
: 874      1977          BEGIN
: 875      1978              FREE_VM(.COM_I_SETCOUNT*%UPVAL, ..A);
: 876      1979              A = 0;
: 877      1980          END;
: 878      1981      1      END;

```

.TITLE FASTSCAN Fast file scan
.IDENT \V04-000\

.PSECT COMMON,NOEXE, OVR,2

```

00000 GLOBAL_BASE:
      .BLKB 0
00000 FREE_LIST:
      .BLKB 8
00008 INPUT_WAIT:
      .BLKB 8
00010 REREAD_WAIT:
      .BLKB 8
00018 OUTPUT_WAIT:
      .BLKB 8
00020 JPI_UIC:
      .BLKB 4
00024 JPI_USERNAME:
      .BLKB 12
00030 JPI_DATE:
      .BLKB 8
00038 JPI_NODE_DESC:
      .BLKB 8
00040 JPI_CURPRIV:
      .BLKB 8
00048 SYI_VERSION:
      .BLKB 4
0004C SYI_SID:
      .BLKB 4
00050 RWSV_HOLD_LIST:
      .BLKB 8

```

00058 RWSV_CRC16:
 .BLKB 64
00098 RWSV_AUTODIN:
 .BLKB 64
000D8 RWSV_FILESET_ID:
 .BLKB 8
000E0 RWSV_VOLUME_ID:
 .BLKB 12
000EC RWSV_VOL_NUMBER:
 .BLKB 2
000EE RWSV_SEG_NUMBER:
 .BLKB 2
000F0 RWSV_FILE_NUMBER:
 .BLKB 4
000F4 RWSV_SAVE_QUAL:
 .BLKB 4
000F8 RWSV_SAVE_FAB:
 .BLKB 4
000FC RWSV_CHAN:
 .BLKB 4
00100 RWSV_XOR_BCB:
 .BLKB 4
00104 RWSV_IN_SEQ:
 .BLKB 4
00108 RWSV_IN_SEQ 0:
 .BLKB 4
0010C RWSV_IN_XOR_SEQ:
 .BLKB 4
00110 RWSV_IN_XOR_RFA:
 .BLKB 6
00116 RWSV_LOOKAHEAD:
 .BLKB 1
00117 RWSV_XOR_SIZE:
 .BLKB 1
00118 RWSV_IN_GROUP_SIZE:
 .BLKB 4
0011C RWSV_IN_ERRORS:
 .BLKB 2
0011E RWSV_IN_XORUSE:
 .BLKB 2
00120 RWSV_IN_ORGERR:
 .BLKB 8
00128 RWSV_IN_VBN:
 .BLKB 4
0012C RWSV_IN_VBN 0:
 .BLKB 4
00130 RWSV_ALLOC:
 .BLKB 4
00134 RWSV_EOF:
 .BLKB 4
00138 RWSV_OUT_SEQ:
 .BLKB 4
0013C RWSV_OUT_VBN:
 .BLKB 4
00140 RWSV_OUT_BLOCK_COUNT:
 .BLKB 4
00144 RWSV_OUT_ERRORS:

00146 RWSV_SEQ_ERRORS: .BLKB 2
00148 RWSV_OUT_GROUP_COUNT: .BLKB 2
00149 RWSV_PADDING: .BLKB 1
0014C QUAL: .BLKB 3
001BC COM_SSNAME: .BLKB 112
001C4 COM_VALID_TYPES: .BLKB 8
001C6 COM_FLAGS: .BLKB 2
001C8 COM_PADDING: .BLKB 2
001C9 COM_BUFF_COUNT: .BLKB 1
001CA COM_I_SETCOUNT: .BLKB 1
001CB COM_O_SETCOUNT: .BLKB 1
001CC COM_I_STRUCNAME: .BLKB 1
001DB COM_O_STRUCNAME: .BLKB 12
001E4 COM_O_BSRDATE: .BLKB 12
001EC ALT_SSNAME: .BLKB 8
0020C INPUT_FUNC: .BLKB 32
0020D INPUT_RTYPE: .BLKB 1
0020E OUTPUT_FUNC: .BLKB 1
0020F FAST_STRUCLEV: .BLKB 1
00210 INPUT_BEG: .BLKB 1
00210 INPUT_CHAN: .BLKB 0
00214 INPUT_FLAGS: .BLKB 4
00216 INPUT_PADDING: .BLKB 2
00218 INPUT_FAB: .BLKB 2
0021C INPUT_NAM: .BLKB 4
00220 INPUT_BCB: .BLKB 4
00224 INPUT_QUAL: .BLKB 4
00228 INPUT_BAD: .BLKB 4
0022C INPUT_BLOCK: .BLKB 4

00230	INPUT_MAXBLOCK:	.BLKB	4
00234	INPUT_MEDIA_ID:	.BLKB	4
00238	INPUT_NAMEDESC:	.BLKB	4
00240	INPUT_STATBLK:	.BLKB	8
00248	INPUT_HDR_BEG:	.BLKB	8
00248	INPUT_CRDATE:	.BLKB	0
00250	INPUT_REVDATE:	.BLKB	8
00258	INPUT_EXPDATE:	.BLKB	8
00260	INPUT_BAKDATE:	.BLKB	8
00268	INPUT_FILEOWNER:	.BLKB	8
0026C	INPUT_FILECHAR:	.BLKB	4
00270	INPUT_RECATTR:	.BLKB	4
00290	INPUT_HDR_END:	.BLKB	32
00290	INPUT_END:	.BLKB	0
00290	INPUT_PROC_LIST:	.BLKB	0
00294	INPUT_PLACEMENT:	.BLKB	4
0029C	INPUT_VBN_LIST:	.BLKB	8
002A4	INPUT_PLACE_LEN:	.BLKB	8
002A6	INPUT_PADDING_2:	.BLKB	2
002A8	OUTPUT_BEG:	.BLKB	2
002A8	OUTPUT_CHAN:	.BLKB	0
002AC	OUTPUT_FLAGS:	.BLKB	4
002AE	OUTPUT_PADDING:	.BLKB	2
002B0	OUTPUT_FAB:	.BLKB	2
002B4	OUTPUT_NAM:	.BLKB	4
002B8	OUTPUT_BCB:	.BLKB	4
002BC	OUTPUT_QUAL:	.BLKB	4
002C0	OUTPUT_BAD:	.BLKB	4

002C4 OUTPUT_BLOCK:
 .BLKB 4
002C8 OUTPUT_MAXBLOCK:
 .BLKB 4
002CC OUTPUT_DEVGEO:
 .BLKB 8
002D4 OUTPUT_ATTBUF:
 .BLKB 144
00364 OUTPUT_END:
 .BLKB 0
00364 LIST_TOTFILES:
 .BLKB 4
00368 LIST_TOTSIZE:
 .BLKB 4
0036C VERIFY_FAB:
 .BLKB 4
00370 VERIFY_USE_COUNT:
 .BLKB 4
00374 VERIFY_QUAL:
 .BLKB 4
00378 COMPARE_BCB:
 .BLKB 4
0037C FAST_BUFFER:
 .BLKB 4
00380 FAST_BUFFER_SIZE:
 .BLKB 4
00384 FAST_RVN:
 .BLKB 1
00385 FAST_PADDING:
 .BLKB 1
00386 DIR_VERLIMIT:
 .BLKB 2
00388 FAST_VOL_BEG:
 .BLKB 0
00388 FAST_IMAP_SIZE:
 .BLKB 4
0038C FAST_IMAP:
 .BLKB 4
00390 FAST_HDR_OFFSET:
 .BLKB 4
00394 FAST_BOOT_LBN:
 .BLKB 4
00398 FAST_VOL_END:
 .BLKB 0
00398 JOUR_BUFFER:
 .BLKB 4
0039C JOUR_DIR:
 .BLKB 4
003AC JOUR_HIBLK:
 .BLKB 4
003A4 JOUR_EFBK:
 .BLKB 4
003A8 JOUR_INBK:
 .BLKB 4
003AC JOUR_FFBYTE:
 .BLKB 2
003AE JOUR_INBYTE:

003B0	JOUR_STRUCT_LEV:	.BLKB	2
		.BLKB	2
003B2	JOUR_COUNT:	.BLKB	1
003B3	JOUR_REVERSE:	.BLKB	1
003B4	JOUR_EXSZ:	.BLKB	2
003B6	JOUR_PADDING:	.BLKB	2
003B8	CHKPT_HIGH_SP:	.BLKB	4
003BC	CHKPT_LOW_SP:	.BLKB	4
003C0	CHKPT_STACK:	.BLKB	4
003C4	CHKPT_VARS:	.BLKB	4
003C8	CHKPT_STATUS:	.BLKB	4
003CC	DIR_BEG:	.BLKB	0
003CC	DIR_CHAN:	.BLKB	4
003D0	DIR_NAM:	.BLKB	4
003D4	DIR_DEV_DESC:	.BLKB	4
003D8	DIR_SEL_DIR:	.BLKB	8
003E0	DIR_SEL_NTV:	.BLKB	8
003E8	DIR_STRUCLEV:	.BLKB	1
003E9	DIR_LEVELS:	.BLKB	1
003EA	DIR_FLAGS:	.BLKB	1
003EB	DIR_STATUS:	.BLKB	1
003EC	DIR_STRING:	.BLKB	320
0052C	DIR_STACK:	.BLKB	612
00790	DIR_SP:	.BLKB	4
00794	DIR_SEL_LATEST:	.BLKB	4
00798	DIR_END:	.BLKB	0
00798	DIR_SCANLIMIT:	.BLKB	36
007BC	INPUT_MTL:	.BLKB	4
007C0	OUTPUT_MTL:	.BLKB	4
007C4	CURRENT_MTL:	.BLKB	4
007C8	CURRENT_VCB:	.BLKB	4


```

007CC CURRENT_WCB:
          .BLKB 4
007D0 ACL_FIB_DESCR:
          .BLKB 8
007D8 ACL_FIB: .BLKB 64
00818 ACL_LENGTH:
          .BLKB 4
0081C ACL_BUFFER:
          .BLKB 4
00820 CRY_P_IN_CONTEXT:
          .BLKB 4
00824 CRY_P_OU_CONTEXT:
          .BLKB 4
00828 CRY_P_DA_CONTEXT:
          .BLKB 4
0082C CRY_P_DATA_ENCIV:
          .BLKB 8
00834 CRY_P_DATA_CODE:
          .BLKB 4
00838 CRY_P_DATA_KEY:
          .BLKB 8
00840 CRY_P_DATA_IV:
          .BLKB 8
00848 CRY_P_DATA_CKSM:
          .BLKB 4

```

.PSECT CODE, NOWRT, 2

```

44 30 30 30 30 30 30 06 00000 P.AAA: .ASCII <6>\000000\
    41 21 5D 5B 53 41 21 00007 P.AAC: .ASCII \!ASC]!AD\
          0000F .BLKB 1
          00000008 00010 P.AAB: .LONG 8
          00000000 00014 .ADDRESS P.AAC

```

MFD=

```

          P.AAA
.EXTRN LIB$EXTRACT CONCEALED
.EXTRN CHECKSUM, CHECKSUM2
.EXTRN GEN_FID_RECORD, INIT_NAMEBLOCK
.EXTRN VOLUME_ATTRIBUTES
.EXTRN SAVE_ONE_FILE, FILE_ERROR
.EXTRN INIT_ATTR, LEFT_ONE
.EXTRN MAKE_STRING, INIT_SEL_INFO
.EXTRN MATCH_DIRECTORY
.EXTRN MATCH_FILENAME, SELECT_INPUT_FILE
.EXTRN TERMINATE_SCAN, FREE_VM
.EXTRN GET_VM, GET_ZERO_VM
.EXTRN LIB$SIGNAL, BACKUPS_MAXVOLS
.EXTRN BACKUPS_PROCINDEX
.EXTRN BACKUPS_OPENDIR
.EXTRN BACKUPS_OPENIN, BACKUPS_READDIR
.EXTRN BACKUPS_BADDIR, BACKUPS_NOSUCHRVM
.EXTRN LIB$INVFILSPE, SYSSQIOQ
.EXTRN SYSSFAO

```

```

          OFFC 00000 .ENTRY FAST_FILE_SCAN, Save R2,R3,R4,R5,R6,R7,R8,- : 1330
          SE 90 AE 9E 00002 MOVAB R9,RT0,R1T
          -112(SP), SP

```

		28	AE	40	8F	9A	00006	MOVZBL	#64, FIB_DESC	1369
		2C	AE	30	AE	9E	0000B	MOVAB	FIB, FIB_DESC+4	1370
		00000000'	EF	40	8F	88	00010	BISB2	#64, COM_FLAGS	1375
		00000000'	EF	8000	8F	3C	00018	MOVZWL	#32768, FAST_BUFFER_SIZE	1376
		00000000G	7E	8000	8F	3C	00021	MOVZWL	#32768, -(SP)	1377
		00000000'	00		01	FB	00026	CALLS	#1, GET_VM	
			EF		50	DO	0002D	MOVL	R0, FAST_BUFFER	
0040	8F		58		01	DO	00034	MOVL	#1, RVN	1382
			6E		00	2C	00037	MOVCS	#0, (SP), #0, #64, FIB	1397
				30	AE		0003E			
			56	00200100	8F	DO	00040	MOVL	#2097408, ACCTL	1398
		30	AE		56	DO	00047	MOVL	ACCTL, FIB	1399
		34	AE	00010001	8F	DO	0004B	MOVL	#65537, FIB+4	1400
		38	AE		58	BO	00053	MOVW	RVN, FIB+8	1402
		14	AE	00090008	8F	DO	00057	MOVL	#589832, ATR_DESC	1403
		18	AE		AE	9E	0005F	MOVAB	STATBLK, ATR_DESC+4	1405
				1C	AE	D4	00064	CLRL	ATR_DESC+8	1406
					7E	D4	00067	CLRL	-(SP)	1412
				18	AE	9F	00069	PUSHAB	ATR_DESC	
					7E	7C	0006C	CLRQ	-(SP)	
				3C	AE	9F	00070	CLRL	-(SP)	
				40	AE	9F	00075	PUSHAB	FIB_DESC	
			7E	72	8F	9A	00078	CLRL	-(SP)	
				00000000'	EF	DD	0007C	PUSHAB	IOSB	
					7E	D4	00082	MOVZBL	#114, -(SP)	
		00000000G	00		0C	FB	00084	PUSHL	INPUT_CHAN	
			5A		50	DO	0008B	CLRL	-(SP)	
			04		5A	E9	0008E	CALLS	#12, SYSSQIOW	
			5A	20	AE	3C	00091	MOVL	R0, STATUS	1413
		0000025C	8F		5A	D1	00095	BLBC	STATUS, 2\$	
					52	12	0009C	MOVZWL	IOSB, STATUS	1414
0040	8F		6E		00	2C	0009E	CMPL	STATUS, #604	
				30	AE		000A5	BNEQ	3\$	
			56	00200000	8F	DO	000A7	MOVCS	#0, (SP), #0, #64, FIB	1417
		30	AE	00200000	8F	DO	000AE			1418
		34	AE	00010001	8F	DO	000B6	MOVL	#2097152, ACCTL	1419
		38	AE		58	BO	000BE	MOVL	#2097152, FIB	1420
				18	7E	D4	000C2	MOVL	#65537, FIB+4	1422
					AE	9F	000C4	MOVW	RVN, FIB+8	1428
					7E	7C	000C7	CLRL	-(SP)	
					7E	D4	000C9	PUSHAB	ATR_DESC	
				3C	AE	9F	000CB	CLRQ	-(SP)	
				40	7E	7C	000CE	CLRL	-(SP)	
			7E	72	AE	9F	000D0	PUSHAB	FIB_DESC	
				00000000'	8F	9A	000D3	CLRQ	-(SP)	
					EF	DD	000D7	PUSHAB	IOSB	
					7E	D4	000DD	MOVZBL	#114, -(SP)	
		00000000G	00		0C	FB	000DF	PUSHL	INPUT_CHAN	
			5A		50	DO	000E6	CLRL	-(SP)	
			07		5A	E9	000E9	CALLS	#12, SYSSQIOW	
			5A	20	AE	3C	000EC	MOVL	R0, STATUS	1429
			1B		5A	E8	000F0	BLBC	STATUS, 4\$	
				0500	8F	BB	000F3	MOVZWL	IOSB, STATUS	1431
		7E 00000000'	EF		10	C1	000F7	BLBS	STATUS, 5\$	
					02	DD	000FF	PUSHR	#*M<R8,R10>	1433
								ADDL3	#16, INPUT_QUAL, -(SP)	
								PUSHL	#2	

		00000000G	8F	DD	00101	PUSHL	#BACKUP\$ PROCINDEX		
		00	05	FB	00107	CALLS	#5, LIB\$SIGNAL		
		00000000'	EF	DD	0010E	5\$: PUSHL	FAST_BUFFER	1439	
			58	DD	00114	PUSHL	RVN		
	0000V	CF	02	FB	00116	CALLS	#2, READ_HOMEBLOCK		
		01	58	D1	0011B	CMPL	RVN, #1	1446	
			2A	12	0011E	BNEQ	8\$		
		52 00000000'	EF	9E	00120	MOVAB	FAST_VOL_BEG, R2	1449	
		53 00000000'	EF	9E	00127	MOVAB	FAST_VOL_END-4, R3		
			15	11	0012E	BRB	7\$		
		50 00000000'	EF	9A	00130	6\$: MOVZBL	COM_I SETCOUNT, R0	1451	
7E		50	02	78	00137	ASHL	#2, -R0, -(SP)		
	00000000G	00	01	FB	0013B	CALLS	#1, GET_ZERO_VM		
		82	50	D0	00142	MOVL	R0, (A)↑		
		53	52	D1	00145	7\$: CML	A, R3	1449	
			E6	1B	00148	BLEQU	6\$		
		54 00000000'	EF	D0	0014A	8\$: MOVL	FAST_BUFFER, R4	1461	
		53 00000000'FF48	DE	00151	MOVAL	@FAST_IMAP_SIZE[RVN], R3	1462		
		51 00000000'FF48	DE	00159	MOVAL	@FAST_HDR_OFFSET[RVN], R1	1464		
		02 00000000'	EF	91	00161	CMPL	FAST_STRUCLEV, #2	1458	
			19	12	00168	BNEQ	9\$		
		50	0E	A4	3C	0016A	MOVZWL	14(R4), CLUSTER	1461
		55	20	A4	3C	0016E	MOVZWL	32(R4), R5	1462
	FC	A3	55	D0	00172	MOVL	R5, -4(R3)		
52		50	02	78	00176	ASHL	#2, CLUSTER, BITMAP_OFFSET	1463	
			52	D6	0017A	:NCL	BITMAP_OFFSET		
	FC	A1	6540	DE	0017C	MOVAL	(R5)[CLUSTER], -4(R1)	1464	
			12	11	00181	BRB	10\$	1458	
		50	01	D0	00183	9\$: MOVL	#1, CLUSTER	1468	
	FC	A3	64	3C	00186	MOVZWL	(R4), -4(R3)	1469	
		52	03	D0	0018A	MOVL	#3, BITMAP_OFFSET	1470	
	FC	A1	64	3C	0018D	MOVZWL	(R4), -4(RT)	1471	
	FC	A1	02	C0	00191	ADDL2	#2, -4(R1)		
		54 00000000'FF48	DE	00195	10\$: MOVAL	@FAST_IMAP[RVN], R4	1477		
7E	FC	A3	09	78	0019D	ASHL	#9, -4(R3), -(SP)		
	00000000G	00	01	FB	001A2	CALLS	#1, GET_VM		
		FC	50	D0	001A9	MOVL	R0, -4(R4)		
			7E	7C	001AD	CLRQ	-(SP)	1484	
			7E	D4	001AF	CLRL	-(SP)		
			52	DD	001B1	PUSHL	BITMAP_OFFSET		
		50 00000000'FF48	DE	001B3	MOVAL	@FAST_IMAP_SIZE[RVN], R0			
7E	FC	A0	09	78	001BB	ASHL	#9, -4(R0), -(SP)		
		50 00000000'FF48	DE	001C0	MOVAL	@FAST_IMAP[RVN], R0			
		FC	A0	DD	001C8	PUSHL	-4(R0)		
			7E	7C	001CB	CLRQ	-(SP)		
		40	AE	9F	001CD	PUSHAB	IOSB		
			31	DD	001D0	PUSHL	#49		
		00000000'	EF	DD	001D2	PUSHL	INPUT_CHAN		
			7E	D4	001D8	CLRL	-(SP)		
	00000000G	00	0C	FB	001DA	CALLS	#12, SYSSQIOW		
		5A	50	D0	001E1	MOVL	R0, STATUS		
		07	5A	E9	001E4	BLBC	STATUS, 11\$	1485	
		5A	20	AE	3C	001E7	MOVZWL	IOSB, STATUS	
		1B	5A	E8	001EB	BLBS	STATUS, 12\$	1486	
		0500	8F	BB	001EE	11\$: PUSHR	#*M<R8,R10>	1488	
7E	00000000'	EF	10	C1	001F2	ADDL3	#16, INPUT_QUAL, -(SP)		
			02	DD	001FA	PUSHL	#2		

00000000G	00	00000000G	8F DD 001FC	PUSHL	#BACKJPS PROCINDEX	
			05 FB 00202	CALLS	#5, LIB\$SIGNAL	
			7E 7C 00209	CLRQ	-(SP)	1499
			7E D4 0020B	CLRL	-(SP)	
7E	FC	50 00000000'FF48	DE 0020D	MOVAL	@FAST_HDR_OFFSET[RVN], R0	
			01 C1 00215	ADDL3	#1, -4(R0), -(SP)	
7E		0200	8F 3C 0021A	MOVZWL	#512, -(SP)	
7E	00000000'	EF 00000200	8F C1 0021F	ADDL3	#512, FAST_BUFFER, -(SP)	
			7E 7C 0022B	CLRQ	-(SP)	
		40	AE 9F 0022D	PUSHAB	IOSB	
			31 DD 00230	PUSHL	#49	
		00000000'	EF DD 00232	PUSHL	INPUT_CHAN	
			7E D4 00238	CLRL	-(SP)	
00000000G	00		0C FB 0023A	CALLS	#12, SYSSQIOW	
			5A 50 D0 00241	MOVL	R0, STATUS	1500
			07 5A E9 00244	BLBC	STATUS, 13\$	
		20	AE 3C 00247	MOVZWL	IOSB, STATUS	1501
			1B 5A E8 0024B	BLBS	STATUS, 14\$	
7E	00000000'	EF 0500	8F BB 0024E	PUSHR	#^M<R8,R1C>	1503
			10 C1 00252	ADDL3	#16, INPUT_QUAL, -(SP)	
			02 DD 0025A	PUSHL	#2	
		00000000G	8F DD 0025C	PUSHL	#BACKUP\$ PROCINDEX	
00000000G	00		05 FB 00262	CALLS	#5, LIB\$SIGNAL	
			57 D4 00269	CLRL	EOF	1506
		50 00000000'	EF D0 0026B	MOVL	FAST_BUFFER, R0	1508
		02 00000000'	EF 91 00272	CMPB	FAST_STRUCLEV, #2	1507
			0C 12 00279	BNEQ	15\$	
50	021C	C0	10 9C 0027B	ROTL	#16, 540(R0), R0	1508
		57 FF	A0 9E 00281	MOVAB	-1(R0), EOF	
			0B 11 00285	BRB	16\$	
		02 0C	A0 91 00287	CMPB	12(R0), #2	1509
			05 12 0028B	BNEQ	16\$	
57	10	AE	10 9C 0028D	ROTL	#16, STATBLK+4, EOF	1510
		53 00000000'FF48	DE 00292	MOVAL	@FAST_IMAP_SIZE[RVN], R3	1517
53	FC	A3	07 78 0029A	ASHL	#7, -4(R3), R3	
		52	53 D0 0029F	MOVL	R3, J	
			39 11 002A2	BRB	19\$	
		50 00000000'FF48	DE 002A4	MOVAL	@FAST_IMAP[RVN], R0	1519
		50 FC B042	D0 002AC	MOVL	@-4(R0)[J], R0	
			2A 13 002B1	BEQL	19\$	
53		52	05 78 002B3	ASHL	#5, J, R3	1523
			50 DD 002B7	PUSHL	R0	1524
		00000000G	01 FB 002B9	CALLS	#1, LEFT ONE	
51		53	50 C1 002C0	ADDL3	R0, R3, R1	1523
		50 00000000'FF48	DE 002C4	MOVAL	@FAST_HDR_OFFSET[RVN], R0	1525
		51 FC	A0 C0 002CC	ADDL2	-4(R0), RT	
		57	51 D1 002D0	CMPB	R1, EOF	1526
			03 1E 002D3	BGEQU	18\$	
		51	57 D0 002D5	MOVL	EOF, R1	
		57	51 D0 002D8	MOVL	R1, EOF	1522
			03 11 002DB	BRB	20\$	1521
		C4	52 F4 002DD	SOBGEQ	J, 17\$	1517
74	00000000'	EF	03 E1 002E0	BBC	#3, QUAL+10, 23\$	1532
			7E 7C 002E8	CLRQ	-(SP)	1544
		7E	01 7D 002EA	MOVQ	#1, -(SP)	
		7E 0200	8F 3C 002ED	MOVZWL	#512, -(SP)	
7E	00000000'	EF 00000200	8F C1 002F2	ADDL3	#512, FAST_BUFFER, -(SP)	

			7E	7C	002FE	CLRQ	-(SP)				
		40	AE	9F	00300	PUSHAB	IOSB				
			31	DD	00303	PUSHL	#49				
		00000000'	EF	DD	00305	PUSHL	INPUT_CHAN				
			7E	D4	00308	CLRL	-(SP)				
	00000000G	00	0C	FB	0030D	CALLS	#12, SYSSQIOW				
		5A	50	DO	00314	MOVL	RO, STATUS				
		07	5A	E9	00317	BLBC	STATUS, 21\$	1545			
		5A	20	AE	3C 0031A	MOVZWL	IOSB, STATUS				
		1B	5A	E8	0031E	BLBS	STATUS, 22\$	1546			
		0500	8F	BB	00321	PUSHR	#^M<R8,R10>	1548			
	7E 00000000'	EF	10	C1	00325	ADDL3	#16, INPUT_QUAL, -(SP)				
			02	DD	0032D	PUSHL	#2				
		00000000G	8F	DD	0032F	PUSHL	#BACKUP\$ PROCINDEX				
		00000000G	00	05	FB	00335	CALLS	#5, LIB\$SIGNAL			
		50	00000000'	FF	48	DE	0033C	22\$:	MOVAL	@FAST_BOOT_LBN[RVN], RO	1553
		FC	A0	01	CE	0C344	MNEGL	#1, -4(R0)			
			51	00000000'	EF	DO	00348	MOVL	FAST_BUFFER, R1	1554	
			0205	C1	95	0034F	TSTB	517(R1)			
			07	12	00353	BNEQ	23\$				
	FC	A0	0204	C1	10	9C	00355	ROTL	#16, 516(R1), -4(R0)	1556	
					7E	7C	0035C	23\$:	CLRQ	-(SP)	1564
					7E	7C	0035E	CLRQ	-(SP)		
					7E	7C	00360	CLRQ	-(SP)		
					7E	7C	00362	CLRQ	-(SP)		
			7E		34	7D	00364	MOVQ	#52, -(SP)		
			00000000'	EF	DD	00367	PUSHL	INPUT_CHAN			
				7E	D4	0036D	CLRL	-(SP)			
		00000000G	00	0C	FB	0036F	CALLS	#12, SYSSQIOW			
	03	00000000'	EF	03	E0	00376	BBS	#3, QUAL+10, 24\$	1567		
				0107	31	0037E	BRW	30\$			
	0040	8F	00	6E	00	2C	00381	24\$:	MGC5	#0, (SP), #0, #64, FIB	1573
				30	AE		00388				
			30	AE	56	DO	0038A	MOVL	ACCTL, FIB	1574	
			34	AE	8F	DO	0038E	MOVL	#131074, FIB+4	1575	
			38	AE	58	B0	00396	MOVW	RVN, FIB+8	1577	
					7E	7C	0039A	CLRQ	-(SP)	1582	
					7E	7C	0039C	CLRQ	-(SP)		
					7E	D4	0039E	CLRL	-(SP)		
			3C	AE	9F	003A0	PUSHAB	FIB_DESC			
				7E	7C	003A3	CLRQ	-(SP)			
			40	AE	9F	003A5	PUSHAB	IOSB			
			7E	72	8F	9A	003A8	MOVZBL	#114, -(SP)		
			00000000'	EF	DD	003AC	PUSHL	INPUT_CHAN			
				7E	D4	003B2	CLRL	-(SP)			
		00000000G	00	0C	FB	003B4	CALLS	#12, SYSSQIOW			
			5A	50	DO	003BB	MOVL	RO, STATUS			
			07	5A	E9	003BE	BLBC	STATUS, 25\$	1583		
			5A	20	AE	3C 003C1	MOVZWL	IOSB, STATUS			
			1B	5A	E8	003C5	BLBS	STATUS, 26\$	1584		
			0500	8F	BB	003C8	PUSHR	#^M<R8,R10>	1586		
	7E 00000000'	EF	10	C1	003CC	ADDL3	#16, INPUT_QUAL, -(SP)				
			02	DD	003D4	PUSHL	#2				
		00000000G	8F	DD	003D6	PUSHL	#BACKUP\$ PROCINDEX				
			00	05	FB	003DC	CALLS	#5, LIB\$SIGNAL			
				7E	7C	003E3	CLRQ	-(SP)	1597		
			7E	01	7D	003E5	MOVQ	#1, -(SP)			

	7E	0200	8F	3C	003E8	MOVZWL	#512, -(SP)					
	7E	00000000'	EF	00000400	8F	C1	003ED	ADDL3	#1024, FAST_BUFFER, -(SP)			
				40	7E	7C	003F9	CLRQ	-(SP)			
					AE	9F	003FB	PUSHAB	10'B			
					31	DD	003FE	PUSHL	#49			
		00000000'	EF	DD	00400	PUSHL	INPUT_CHAN					
			7E	D4	00406	CLRL	-(SP)					
	00000000G		00	OC	FB	00408	CALLS	#12, SYSSQIOW				
			5A	50	DO	0040F	MOVL	RO, STATUS				
			07	5A	E9	00412	BLBC	STATUS, 27\$	1598			
			5A	20	AE	3C	00415	MOVZWL	IOSB, STATUS			
			1B	5A	EB	00419	BLBS	STATUS, 28\$	1599			
	7E	00000000'	EF	0500	8F	BB	0041C	PUSHR	#*M<R8,R10>	1601		
					10	C1	00420	ADDL3	#16, INPUT_QUAL, -(SP)			
					02	DD	00428	PUSHL	#2			
	00000000G		00	00000000G	8F	DD	0042A	PUSHL	#BACKUP\$ PROCINDEX			
					05	FB	00430	CALLS	#5, LIB\$SIGNAL			
					7E	7C	00437	CLRQ	-(SP)	1608		
					7E	7C	00439	CLRQ	-(SP)			
					7E	7C	0043B	CLRQ	-(SP)			
					7E	7C	0043D	CLRQ	-(SP)			
			7E		34	7D	0043F	MOVQ	#52, -(SP)			
		00000000'	EF	DD	00442	PUSHL	INPUT_CHAN					
			7E	D4	00448	CLRL	-(SP)					
	00000000G		00	OC	FB	0044A	CALLS	#12, SYSSQIOW				
			0B	00000000'	EF	E9	00451	BLBC	QUAL+14, 29\$	1613		
	5B	00000000'	EF	0B	00	ED	00458	CMPZV	#0, #8, QUAL+79, RVN			
					25	12	00461	BNEQ	30\$			
			50	00000000'	FF	48	DE	00463	MOVAL	@FAST HDR OFFSET[RVN], RO	1619	
			7E	57	FC	A0	C3	0046B	SUBL3	-4(RO), EOF, -(SP)		
			50	00000000'	EF	D0	00470	MOVL	FAST_BUFFER, RO	1618		
				0400	C0	9F	00477	PUSHAB	1024(RO)			
				0200	C0	9F	0047B	PUSHAB	512(RO)	1617		
					50	DD	0047F	PUSHL	RO	1616		
	0040	8F	00	00000000G	00	04	FB	00481	CALLS	#4, VOLUME_ATTRIBUTES		
					6E	00	2C	00488	MOVCS	#0, (SP), #0, #64, FIB	1625	
				30	AE		0048F					
				34	AE	00010001	56	DO	00491	MOVL	ACCTL, FIB	1626
				38	AE		8F	DO	00495	MOVL	#65537, FIB+4	1627
							58	B0	0049D	MOVW	RVN, FIB+8	1629
							7E	7C	004A1	CLRQ	-(SP)	1634
							7E	7C	004A3	CLRQ	-(SP)	
							7E	D4	004A5	CLRL	-(SP)	
				3C	AE	9F	004A7	PUSHAB	FIB_DESC			
					7E	7C	004AA	CLRQ	-(SP)			
				40	AE	9F	004AC	PUSHAB	IOSB			
			7E	72	8F	9A	004AF	MOVZBL	#114, -(SP)			
		00000000'	EF	DD	004B3	PUSHL	INPUT_CHAN					
					7E	D4	004B9	CLRL	-(SP)			
	00000000G		00	OC	FB	004BB	CALLS	#12, SYSSQIOW				
			5A	50	DO	004C2	MOVL	RO, STATUS				
			07	5A	E9	004C5	BLBC	STATUS, 31\$	1635			
			5A	20	AE	3C	004C8	MOVZWL	IOSB, STATUS			
			1B	5A	EB	004CC	BLBS	STATUS, 32\$	1636			
				0500	8F	BB	004CF	PUSHR	#*M<R8,R10>	1638		
	7E	00000000'	EF	10	C1	004D3	ADDL3	#16, INPUT_QUAL, -(SP)				
				02	DD	004DB	PUSHL	#2				

		00000000G	00	00000000G	8F DD 004DD	PUSHL	#BACKUP\$ PROCINDEX		
					05 FB 004E3	CALLS	#5, LIB\$SIGNAL		
					00000000'	EF 94 004EA	32\$: CLR	DIR STATUS	1644
					50 00000000'	FF48 DE 004F0	MOVAL	@FAST HDR OFFSET[RVN], R0	1645
59	FC		A0		01 C1 004F8	ADDL3	#1, -4(ROT) VBN		
			6E	01	A7 9E 004FD	MOVAB	1(R7), (SP)		1655
			57		59 D1 00501	33\$: CML	VBN, EOF		1646
					03 1B 00504	BLEQU	34\$		
				01B4	31 00506	BRW	58\$		
50			6E		59 C3 00509	34\$: SUBL3	VBN, (SP), R0		1655
		00000040	8F		50 D1 0050D	CML	R0, #64		
					04 1B 00514	BLEQU	35\$		
			50	40	8F 9A 00516	MOVZBL	#64, R0		
			5B		50 D0 0051A	35\$: MOVL	R0, READ_COUNT		
					7E 7C 0051D	CLRQ	-(SP)		1662
					7E D4 0051F	CLRL	-(SP)		
					59 DD 00521	PUSHL	VBN		
7E			5B		09 78 00523	ASHL	#9, READ_COUNT, -(SP)		
					00000000'	EF DD 00527	PUSHL	FAST_BUFFER	
					7E 7C 0052D	CLRQ	-(SP)		
				40	AE 9F 0052F	PUSHAB	IOSB		
					31 DD 00532	PUSHL	#49		
					00000000'	EF DD 00534	PUSHL	INPUT_CHAN	
					7E D4 0053A	CLRL	-(SP)		
		00000000G	00		0C FB 0053C	CALLS	#12, SYSSQIOW		
			5A		50 D0 00543	MOVL	R0, STATUS		
			07		5A E9 00546	BLBC	STATUS, 36\$		1663
			5A	20	AE 3C 00549	MOVZWL	IOSB, STATUS		
			4E		5A EB 0054D	BLBS	STATUS, 40\$		1669
			56		01 CE 00550	36\$: MNEGL	#1, XVBN		1672
					45 11 00553	BRB	39\$		
50			56		09 78 00555	37\$: ASHL	#9, XVBN, R0		1678
52			50	00000000'	EF C1 00559	ADDL3	FAST_BUFFER, R0, HEADER		
					7E 7C 00561	CLRQ	-(SP)		1685
					7E D4 00563	CLRL	-(SP)		
					6649 9F 00565	PUSHAB	(XVBN)[VBN]		
			7E	0200	8F 3C 00568	MOVZWL	#512, -(SP)		
					52 DD 0056D	PUSHL	HEADER		
					7E 7C 0056F	CLRQ	-(SP)		
				40	AE 9F 00571	PUSHAB	IOSB		
					31 DD 00574	PUSHL	#49		
					00000000'	EF DD 00576	PUSHL	INPUT_CHAN	
					7E D4 0057C	CLRL	-(SP)		
		00000000G	00		0C FB 0057E	CALLS	#12, SYSSQIOW		
			5A		50 D0 00585	MOVL	R0, STATUS		
			07		5A E9 00588	BLBC	STATUS, 38\$		1686
			5A	20	AE 3C 0058B	MOVZWL	IOSB, STATUS		
			08		5A EB 0058F	BLBS	STATUS, 39\$		1687
0200	8F		00		6E 00 2C 00592	38\$: MOVCS	#0, (SP), #0, #512, (HEADER)		1689
					62 00599				
			B7		56 5B F2 0059A	39\$: AOBLSS	READ_COUNT, XVBN, 37\$		1672
					52 00000000'	EF D0 0059E	40\$: MOVL	FAST_BUFFER, HEADER	1697
					54 01 CE 005A5	MNEGL	#1, XVBN		1698
					0102 31 005A8	BRW	55\$		
51			59		54 C1 005AB	41\$: ADDL3	XVBN, VBN, R1		1708
			50	00000000'	FF48 DE 005AF	MOVAL	@FAST HDR OFFSET[RVN], R0		
53			51	FC	A0 C3 005B7	SUBL3	-4(ROT), RT, FILE_NUMBER		

50	53	04	AE		53	B0	005BC	MOVW	FILE_NUMBER, FILE_ID	1709	
		08			10	EF	005C0	EXTZV	#16, #8, FILE_NUMBER, R0	1710	
		09	AE		50	90	005C5	MOVW	R0, FILE_ID+5		
		02		00000000'	EF	91	005C9	CMPB	FAST_STRUCLEV, #2	1711	
					07	12	005D0	BNEQ	42\$		
		06	AE	0A	A2	B0	005D2	MOVW	10(HEADER), FILE_ID+2	1712	
					05	11	005D7	BRB	43\$		
		06	AE	04	A2	B0	005D9	MOVW	4(HEADER), FILE_ID+2	1713	
		08	AE		58	90	005DE	MOVW	RVN, FILE_ID+4	1714	
		25		00000000'	EF	03	E1	005E2	BBC	#3, QUAL+T0, 45\$	1720
					54	D5	005EA	TSTL	XVBN	1721	
					21	12	005EC	BNEQ	45\$		
			08	00000000'	EF	E9	005EE	BLBC	QUAL+14, 44\$	1722	
			08		00	ED	005F5	CMPZV	#0, #8, QUAL+79, RVN		
					0F	12	005FE	BNEQ	45\$		
				0108	8F	BB	00600	PUSHR	#*M<R3,R8>	1724	
				0804	8F	BB	00604	PUSHR	#*M<R2,R11>		
			00	00000000G	04	FB	00608	CALLS	#4, GEN_FID_RECORD		
					04	AE	9F	0060F	PUSHAB	FILE_ID	1732
					52	DD	00612	PUSHL	HEADER		
			0000V		02	FB	00614	CALLS	#2, VERIFY_HEADER		
					50	D0	00619	MOVL	R0, STATUS		
					01	EF	91	0061C	CMPB	FAST_STRUCLEV, #1	1734
					13	12	00623	BNEQ	46\$		
					50	DE	00625	MOVAL	@FAST_IMAP[RVN], R0	1735	
					51	A3	9E	0062D	MOVAB	-1(R3), R1	
			02	FC	B0	51	E0	00631	BBS	R1, @-4(R0), 46\$	
					5A	D4	00636	CLRL	STATUS	1737	
					50	DE	00638	MOVAL	@FAST_IMAP[RVN], R0	1743	
					53	D7	00640	DECL	R3		
			00	FC	B0	53	E5	00642	BBCC	R3, @-4(R0), 47\$	
					5A	E9	00647	BLBC	STATUS, 54\$	1746	
					51	9A	0064A	MOVZBL	1(HEADER), R1	1753	
					51	3E	0064E	MOVAW	(HEADER)[R1], MAP_AREA		
					55	D4	00652	CLRL	R5	1756	
					02	EF	91	00654	CMPB	FAST_STRUCLEV, #2	
					07	12	0065B	BNEQ	48\$		
					55	D6	0065D	INCL	R5		
					04	A2	B5	0065F	TSTW	4(HEADER)	1757
					02	11	00662	BRB	49\$		
					61	95	00664	TSTB	(MAP_AREA)	1758	
					40	12	00666	BNEQ	54\$		
			00	FC	B0	53	E2	00668	BBSS	R3, @-4(R0), 50\$	1769
					55	E9	0066D	BLBC	R5, 51\$	1774	
					50	A2	9E	00670	MOVAB	52(R2), FCH	1775
					04	11	00674	BRB	52\$		
					50	A2	9E	00676	MOVAB	12(R2), FCH	1776
					60	B5	0067A	TSTW	(FCH)	1785	
					1D	19	0067C	BLSS	53\$		
					52	DD	0067E	PUSHL	HEADER	1790	
					01	FB	00680	CALLS	#1, INIT_ATTR		
			19	00000000'	EF	03	E0	00687	BBS	#3, QUAL+T0, 54\$	1791
					01	DD	0068F	PUSHL	#1	1793	
					01	FB	00691	CALLS	#1, SELECT_INPUT_FILE		
					0D	E8	00698	BLBS	R0, 54\$		
					50	DE	0069B	MOVAL	@FAST_IMAP[RVN], R0	1797	
			00	FC	B0	53	E4	006A3	BBSC	R3, @-4(R0), 54\$	

02	52	0200	C2	9E	006A8	54\$:	MOVAB	512(R2), HEADER	1802
	54		5B	F2	006AD	55\$:	AOBLS	READ_COUNT, XVBN, 56\$	1698
			03	11	006B1		BRB	57\$	
			FEF5	31	006B3	56\$:	BRW	41\$	
	59	40	A9	9E	006B6	57\$:	MOVAB	64(R9), VBN	1806
			FE44	31	006BA		BRW	33\$	1646
			7E	7C	006BD	58\$:	CLRQ	-(SP)	1814
			7E	7C	006BF		CLRQ	-(SP)	
			7E	7C	006C1		CLRQ	-(SP)	
			7E	7C	006C3		CLRQ	-(SP)	
	7E		34	7D	006C5		MOVQ	#52, -(SP)	
		00000000'	EF	DD	006C8		PUSHL	INPUT_CHAN	
			7E	D4	006CE		CLRL	-(SP)	
	00000000G	00	0C	FB	006D0		CALLS	#12, SYSSQIOW	
			58	D6	006D7		INCL	RVN	1817
58	00000000'	EF	08	00	ED	006D9	CMPZV	#0, #8, COM_I_SETCOUNT, RVN	1819
			03	1F	006E2		BLSSU	59\$	
			F95J	31	006E4		BRW	1\$	
		00000000'	EF	DD	006E7	59\$:	PUSHL	FAST_BUFFER	1824
		00000000'	EF	DD	006ED		PUSHL	FAST_BUFFER_SIZE	
	00000000G	00	02	FB	006F3		CALLS	#2, FREE_VM	
		00000000'	EF	7C	006FA		CLRQ	FAST_BUFFER	1825
	00000000'	EF	40	8F	8A	00700	BICB2	#64, COM_FLAGS	1826
		53	00000000'	EF	9A	00708	MOVZBL	COM_I_SETCOUNT, R3	1831
				52	D4	0070F	CLRL	RVN	
				07	11	00711	BRB	61\$	
				52	DD	00713	60\$:	PUSHL	RVN
	0000V	CF	01	FB	00715		CALLS	#1, DIR_SCAN	
F5		52	53	F3	0071A	61\$:	AOBLEQ	R3, RVN, 60\$	
	00000000'	EF	0200	8F	3C	0071E	MOVZWL	#512, FAST_BUFFER_SIZE	1836
		7E	0200	8F	3C	00727	MOVZWL	#512, -(SP)	1837
	00000000G	00	01	FB	0072C		CALLS	#1, GET_VM	
	00000000'	EF	50	D0	00733		MOVL	R0, FAST_BUFFER	
03	00000000'	EF	03	E0	0073A		BBS	#3, QUAL+10, 62\$	1842
			01CA	31	00742		BRW	75\$	
		59	00000000'	EF	9A	00745	62\$:	MOVZBL	COM_I_SETCOUNT, R9
				56	D4	0074C	CLRL	RVN	1845
				01B8	31	0074E	BRW	74\$	
		50	00000000'	FF46	DE	00751	63\$:	MOVAL	@FAST_IMAP_SIZE[RVN], R0
57	FC	A0		0C	78	00759	ASHL	#12, =4(R0), R7	1850
				58	D4	0075E	CLRL	FILE_NUMBER	
				01A0	31	00760	64\$:	BRW	73\$
		50	00000000'	FF46	DE	00763	65\$:	MOVAL	@FAST_IMAP[RVN], R0
		51	FF	A8	9E	00768	MOVAB	-1(R8), R1	1852
		EC	FC	B0	51	E1	0076F	BBC	R1, @-4(R0), 64\$
0040	8F	00		6E	00	2C	00774	MOVCS	#0, (SP), #0, #64, FIB
					30	AE	0077B		1864
		30	AE	00200000	8F	D0	0077D	MOVL	#2097152, FIB
		34	AE	00010001	8F	D0	00785	MOVL	#65537, FIB+4
		38	AE		56	B0	0078D	MOVW	RVN, FIB+8
					7E	7C	00791	CLRQ	-(SP)
					7E	7C	00793	CLRQ	-(SP)
					7E	D4	00795	CLRL	-(SP)
			3C	AE	9F	00797	PUSHAB	FIB_DESC	
				7E	7C	0079A	CLRQ	-(SP)	
			40	AE	9F	0079C	PUSHAB	IOSB	
		7E	72	8F	9A	0079F	MOVZBL	#114, -(SP)	

		00000000'	EF DD 007A3	PUSHL	INPUT_CHAN	
			7E D4 007A9	CLRL	-(SP)	
	00000000G	00	OC FB 007AB	CALLS	#12, SYSSQIOW	
		5A	50 D0 007B2	MOVL	R0, STATUS	
		07	5A E9 007B5	BLBC	STATUS, 66\$	1874
		5A	20 AE 3C 007B8	MOVZWL	IOSB, STATUS	
		1B	5A E8 007BC	BLBS	STATUS, 67\$	1875
		0440	8F BB 007BF	PUSHR	#*M<R6,R10>	1877
7E	00000000'	EF	10 C1 007C3	ADDL3	#16, INPUT_QUAL, -(SP)	
			02 DD 007CB	PUSHL	#2	
	00000000G	00	8F DD 007CD	PUSHL	#BACKUP\$ PROCINDEX	
			05 FB 007D3	CALLS	#5, LIB\$SIGNAL	
			7E 7C 007DA	CLRQ	-(SP)	1888
			7E D4 007DC	CLRL	-(SP)	
	50	00000000'	FF46 DE 007DE	MOVVAL	@FAST_HDR_OFFSET[RVN], R0	
			FC B048 9F 007E6	PUSHAB	@-4(R0)[FILE_NUMBER]	
	7E	0200	8F 3C 007EA	MOVZWL	#512, -(SP)	
		00000000'	EF DD 007EF	PUSHL	FAST_BUFFER	
			7E 7C 007F5	CLRQ	-(SP)	
		40	AE 9F 007F7	PUSHAB	IOSB	
			31 DD 007FA	PUSHL	#49	
		00000000'	EF DD 007FC	PUSHL	INPUT_CHAN	
			7E D4 00802	CLRL	-(SP)	
	00000000G	00	OC FB 00804	CALLS	#12, SYSSQIOW	
		5A	50 D0 0080B	MOVL	R0, STATUS	
		07	5A E9 0080E	BLBC	STATUS, 68\$	1889
		5A	20 AE 3C 00811	MOVZWL	IOSB, STATUS	
		1B	5A E8 00815	BLBS	STATUS, 69\$	1890
		0440	8F BB 00818	PUSHR	#*M<R6,R10>	1892
7E	00000000'	EF	10 C1 0081C	ADDL3	#16, INPUT_QUAL, -(SP)	
			02 DD 00824	PUSHL	#2	
	00000000G	00	8F DD 00826	PUSHL	#BACKUP\$ PROCINDEX	
			05 FB 0082C	CALLS	#5, LIB\$SIGNAL	
			7E 7C 00833	CLRQ	-(SP)	1899
			7E 7C 00835	CLRQ	-(SP)	
			7E 7C 00837	CLRQ	-(SP)	
			7E 7C 00839	CLRQ	-(SP)	
	7E	00000000'	34 7D 0083B	MOVQ	#52, -(SP)	
			EF DD 0083E	PUSHL	INPUT_CHAN	
			7E D4 00844	CLRL	-(SP)	
	00000000G	00	OC FB 00846	CALLS	#12, SYSSQIOW	
		50	00000000'	MOVL	INPUT_NAM, R0	1904
	24	A0	58 B0 00854	MOVW	FILE_NUMBER, 36(R0)	
		08	10 EF 00858	EXTZV	#16, #8, FILE_NUMBER, R1	1905
	29	A0	51 90 0085D	MOVB	R1, 41(R0)	
	28	A0	56 90 00861	MOVB	RVN, 40(R0)	1906
		2A	A0 D4 00865	CLRL	42(R0)	1907
		2E	A0 B4 00868	CLRW	46(R0)	1909
	51	00000000'	EF D0 0086B	MOVL	FAST_BUFFER, R1	1921
	02	00000000'	EF 91 00872	CMPB	FAST_STRUCLEV, #2	1915
			21 12 00879	BNEQ	71\$	
	26	A0	0A A1 B0 0087B	MOVW	10(R1), 38(R0)	1921
		50	61 9A 00880	MOVZBL	(R1), R0	1923
		52	6140 3E 00883	MOVAV	(R1)[R0], *AME_ADDRESS	1924
		53	14 D0 00887	MOVL	#20, NAME_LENGTH	1925
	62	14	20 3A 0088A	LOCC	#32, #20, *(NAME_ADDRESS)	1926
			02 12 0088E	BNEQ	70\$	

				51	D4	00890		CLRL	R1		
				51	D5	00892	70\$:	TSTL	P		1927
				23	13	00894		BEQL	72\$		
53		51		52	C3	00896		SUBL3	NAME_ADDRESS, P, NAME_LENGTH		
				1D	11	0089A		BRB	72\$		1915
	26	A0	04	A1	B0	0089C	71\$:	MOVW	4(R1), 38(R0)		1931
		52	04	AE	9E	008A1		MOVAB	FILENAME, NAME_ADDRESS		1932
			04	AE	9F	008A5		PUSHAB	FILENAME		1933
		50		61	9A	008A8		MOVZBL	(R1), R0		1934
			FA	A140	3F	008AB		PUSHAW	-6(R1)[R0]		1935
	00000000G	00		02	FB	008AF		CALLS	#2, MAKE_STRING		
		53		50	3C	008B6		MOVZWL	R0, NAME_LENGTH		1936
	18	AE	FF	8F	9A	008B9	72\$:	MOVZBL	#255, RSA_DESC		1938
		50	00000000'	EF	D0	008BE		MOVL	INPUT_NAME, R0		1939
		1C	AE	04	A0	008C5		MOVL	4(R0), RSA_DESC+4		
				52	DD	008CA		PUSHL	NAME_ADDRESS		1945
				53	DD	008CC		PUSHL	NAME_LENGTH		
7E	00000000'	EF		10	C1	008CE		ADDL3	#16, INPUT_QUAL, -(SP)		
				24	AE	008D6		PUSHAB	RSA_DESC		
				28	AE	008D9		PUSHAB	RSA_DESC		
			F718	CF	9F	008DC		PUSHAB	P.AAB		
	00000000G	00		06	FB	008E0		CALLS	#6, SYS\$FAO		
		50	00000000'	EF	D0	008E7		MOVL	INPUT_NAME, R0		1946
		03	A0	18	AE	008EE		MOVB	RSA_DESC, 3(R0)		
				50	DD	008F3		PUSHL	R0		1947
	00000000G	00		01	FB	008F5		CALLS	#1, INIT_NAMEBLOCK		
	00000000G	00		00	FB	008FC		CALLS	#0, SAVE_ONE_FILE		1952
FESA		58		01	F1	00903	73\$:	ACBL	R7, #1, FILE_NUMBER, 65\$		1850
FE42		56		01	F1	00909	74\$:	ACBL	R9, #1, RVN, 63\$		1845
			00000000'	EF	DD	0090F	75\$:	PUSHL	FAST_BUFFER		1961
			00000000'	EF	DD	00915		PUSHL	FAST_BUFFER_SIZE		
	00000000G	00		02	FB	0091B		CALLS	#2, FREE_VM		
			00000000'	EF	7C	00922		CLRQ	FAST_BUFFER		1962
		54	00000000'	EF	9A	00928		MOVZBL	COM_I SETCOUNT, R4		1967
		53	00000000'	EF	D0	0092F		MOVL	FAST_IMAP, R3		1969
				52	D4	00936		CLRL	RVN		
				23	11	00938		BRB	77\$		
			FC	A342	DD	0093A	76\$:	PUSHL	-4(R3)[RVN]		
		50	00000000'	FF42	DE	0093E		MOVAL	@FAST_IMAP_SIZE[RVN], R0		
7E		A0		09	78	00946		ASHL	#9, -4(R0), -(SP)		
	FC	00		02	FB	0094B		CALLS	#2, FREE_VM		
	00000000G	53	00000000'	EF	D0	00952		MOVL	FAST_IMAP, R3		1970
			FC	A342	D4	00959		CLRL	-4(R3)[RVN]		
D9		52		54	F3	0095D	77\$:	AOBLEQ	R4, RVN, 76\$		1967
		52	00000000'	EF	9E	00961		MOVAB	FAST_VOL_BEG, R2		1976
		53	00000000'	EF	9E	00968		MOVAB	FAST_VOL_END-4, R3		
				16	11	0096F		BRB	79\$		
				62	DD	00971	78\$:	PUSHL	(A)		1978
		50	00000000'	EF	9A	00973		MOVZBL	COM_I SETCOUNT, R0		
7E		50		02	78	0097A		ASHL	#2, R0, -(SP)		
	00000000G	00		02	FB	0097E		CALLS	#2, FREE_VM		
				82	D4	00985		CLRL	(A)+		1979
				52	D1	00987	79\$:	CMPL	A, R3		1976
		53		E5	1B	0098A		BLEQU	78\$		
				04	0098C			RET			1981

: Routine Size: 2445 bytes, Routine Base: CODE + 0018

FASTSCAN
V04-000

Fast file scan
FAST_FILE_SCAN - fast file scan main routine

^{E 5}
15-Sep-1984 23:56:53
14-Sep-1984 11:53:52

VAX-11 Bliss-32 V4.0-742
[BACKUP.SRC]FASTSCAN.B32;1

Page 34
(3)

```

: 880 1982 1 %SBTTL 'SLOW FILE_SCAN - slow file scan main routine'
: 881 1983 1 GLOBAL ROUTINE SLOW_FILE_SCAN: NOVALUE=
: 882 1984 1
: 883 1985 1 !++
: 884 1986 1
: 885 1987 1 FUNCTIONAL DESCRIPTION:
: 886 1988 1 This routine is the driver for the slow file scan.
: 887 1989 1
: 888 1990 1 INPUT PARAMETERS:
: 889 1991 1 NONE
: 890 1992 1
: 891 1993 1 IMPLICIT INPUTS:
: 892 1994 1 NONE
: 893 1995 1
: 894 1996 1 OUTPUT PARAMETERS:
: 895 1997 1 NONE
: 896 1998 1
: 897 1999 1 IMPLICIT OUTPUTS:
: 898 2000 1 NONE
: 899 2001 1
: 900 2002 1 ROUTINE VALUE:
: 901 2003 1 NONE
: 902 2004 1
: 903 2005 1 SIDE EFFECTS:
: 904 2006 1 NONE
: 905 2007 1
: 906 2008 1 --
: 907 2009 1
: 908 2010 2 BEGIN
: 909 2011 2 DIR_SCAN(1);
: 910 2012 1 END;

```

```

                                0000 0000      .ENTRY SLOW_FILE_SCAN, Save nothing      : 1983
                                01 DD 0002      PUSHL #1                          : 2011
                                01 FB 0004      CALLS #1, DIR_SCAN                          :
                                04 0009      RET                                      : 2012

```

; Routine Size: 10 bytes. Routine Base: CODE + 09A5

```

: 912 2013 1 %SBTTL 'READ_HOMEBLOCK - read home block from index file'
: 913 2014 1 ROUTINE READ_HOMEBLOCK(RVN,BUFFER): NOVALUE=
: 914 2015 1
: 915 2016 1 :++
: 916 2017 1
: 917 2018 1 : FUNCTIONAL DESCRIPTION:
: 918 2019 1 : This routine reads the first good home block of the currently open
: 919 2020 1 : index file into the buffer supplied.
: 920 2021 1
: 921 2022 1 : INPUT PARAMETERS:
: 922 2023 1 :     RVN             - Relative volume number.
: 923 2024 1 :     BUFFER          - Pointer to buffer.
: 924 2025 1
: 925 2026 1 : IMPLICIT INPUTS:
: 926 2027 1 :     NONE
: 927 2028 1
: 928 2029 1 : OUTPUT PARAMETERS:
: 929 2030 1 :     NONE
: 930 2031 1
: 931 2032 1 : IMPLICIT OUTPUTS:
: 932 2033 1 :     BUFFER          - Contains a valid home block.
: 933 2034 1 :     COM_1_SETCOUNT - Count of volumes in volume set.
: 934 2035 1 :     FAST_STRUCLEV  - Structure level (1 or 2) of the volume set.
: 935 2036 1
: 936 2037 1 : ROUTINE VALUE:
: 937 2038 1 :     NONE
: 938 2039 1
: 939 2040 1 : SIDE EFFECTS:
: 940 2041 1 :     NONE
: 941 2042 1
: 942 2043 1 :--
: 943 2044 1
: 944 2045 2 BEGIN
: 945 2046 2 MAP
: 946 2047 2     BUFFER:          REF BBLOCK;      ! Pointer to buffer
: 947 2048 2 LOCAL
: 948 2049 2     STATUS,          ! General status value
: 949 2050 2     IOSB:           VECTOR[4,WORD], ! I/O status block
: 950 2051 2     OLD_STATUS;      ! Save status for error message
: 951 2052 2
: 952 2053 2
: 953 2054 2 ! We keep reading until we get a block that reads without errors and looks
: 954 2055 2 ! like a home block. Track any error status for the eventual error message.
: 955 2056 2
: 956 2057 2 OLD STATUS = $$$ ABORT;
: 957 2058 2 INCR VBN FROM 2 TO 100 DO
: 958 2059 3     BEGIN
: 959 2060 3         STATUS = $QIOW(
: 960 2061 3             FUNC=IOS READVBLK,
: 961 2062 3             CHAN=.INPUT_CHAN,
: 962 2063 3             IOSB=IOSB,
: 963 2064 3             P1=.BUFFER,
: 964 2065 3             P2=512,
: 965 2066 3             P3=.VBN);
: 966 2067 3     IF .STATUS THEN STATUS = .IOSB[0];
: 967 2068 3
: 968 2069 3     IF NOT .STATUS

```

P
P
P
P
P

```

969 2070 3 THEN
970 2071 OLD_STATUS = .STATUS
971 2072 ELSE
972 2073 IF
973 2074 .BUFFER[HM2$B_STRUCLEV] EQL 2 AND
974 2075 .BUFFER[HM2$W_HOMEVBN] EQL .VBN AND
975 2076 .BUFFER[HM2$L_ALTIDXLBN] NEQ 0 AND
976 2077 .BUFFER[HM2$W_CLUSTER] NEQ 0 AND
977 2078 .BUFFER[HM2$W_HOMEVBN] NEQ 0 AND
978 2079 .BUFFER[HM2$W_ALHOMEVBN] NEQ 0 AND
979 2080 .BUFFER[HM2$W_ALTIDXVBN] NEQ 0 AND
980 2081 .BUFFER[HM2$W_IBMAPVBN] NEQ 0 AND
981 2082 .BUFFER[HM2$L_IBMAPLBN] NEQ 0 AND
982 2083 .BUFFER[HM2$L_MAXFILES] NEQ 0 AND
983 2084 .BUFFER[HM2$W_IBMAPSIZE] NEQ 0 AND
984 2085 .BUFFER[HM2$W_RESFILES] NEQ 0 AND
985 2086 CHECKSUM2(.BUFFER, $BYTEOFFSET(HM2$W_CHECKSUM1)) AND
986 2087 CHECKSUM2(.BUFFER, $BYTEOFFSET(HM2$W_CHECKSUM2))
987 2088 THEN
988 2089 BEGIN
989 2090 IF .RVN EQL 1
990 2091 THEN
991 2092 BEGIN
992 2093 FAST_STRUCLEV = 2;
993 2094 IF .BUFFER[HM2$W_SETCOUNT] GTRU MAX_VOLUMES
994 2095 THEN SIGNAL(BACKUP$ MAXVOLS, 1, INPUT_QUAL[QUAL_DEV_DESC]);
995 2096 COM_I_SETCOUNT = .BUFFER[HM2$W_SETCOUNT];
996 2097 IF .COM_I_SETCOUNT EQL 0 THEN COM_I_SETCOUNT = 1;
997 2098 IF
998 2099 .QUAL[QUAL_VOLU] AND
999 2100 .QUAL[QUAL_VOLU_VALUE] GTRU .COM_I_SETCOUNT
1000 2101 THEN
1001 2102 SIGNAL(
1002 2103 BACKUP$ NOSUCHRVN,
1003 2104 2, .QUAL[QUAL_VOLU_VALUE], .COM_I_SETCOUNT);
1004 2105 CHSMOVE(
1005 2106 HM2$S_STRUCNAME,
1006 2107 BUFFER[HM2$T_STRUCNAME],
1007 2108 COM_I_STRUCNAME);
1008 2109 END;
1009 2110 RETURN;
1010 2111 END
1011 2112 ELSE IF
1012 2113 .RVN EQL 1 AND
1013 2114 .BUFFER[HM2$B_STRUCLEV] EQL 1 AND
1014 2115 .BUFFER[HM1$W_CLUSTER] EQL 1 AND
1015 2116 .BUFFER[HM1$L_IBMAPLBN] NEQ 0 AND
1016 2117 .BUFFER[HM1$W_MAXFILES] NEQ 0 AND
1017 2118 .BUFFER[HM1$W_IBMAPSIZE] NEQ 0 AND
1018 2119 CHECKSUM2(.BUFFER, $BYTEOFFSET(HM1$W_CHECKSUM1)) AND
1019 2120 CHECKSUM2(.BUFFER, $BYTEOFFSET(HM1$W_CHECKSUM2))
1020 2121 THEN
1021 2122 BEGIN
1022 2123 FAST_STRUCLEV = 1;
1023 2124 COM_I_SETCOUNT = 1;
1024 2125 IF
1025 2126 .QUAL[QUAL_VOLU] AND

```

```

1026 2127 4        .QUAL[QUAL_VOLU_VALUE] GTRU 1
1027 2128 4        THEN
1028 2129 4        SIGNAL(
1029 2130 4        BACKUP$ NOSUCHRVN,
1030 2131 4        2, .QUAC[QUAL_VOLU_VALUE], .COM_I_SETCOUNT);
1031 2132 4        RETURN;
1032 2133 3        END;
1033 2134 2        END;
1034 2135 2
1035 2136 2
1036 2137 2  ! No good home block found. Report failure.
1037 2138 2
1038 2139 2 SIGNAL(BACKUP$ PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .OLD_STATUS);
1039 2140 1 END;

```

OFFC 00000 READ_HOMEBLOCK:

SB	00000000G	00	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	2014
5A	00000000'	EF	9E	00009	MOVAB	CHECKSUM2, R11	
5E		08	C2	00010	MOVAB	COM_I_SETCOUNT, R10	
59		2C	D0	00013	SUBL2	#8, SP	2057
56	08	AC	D0	00016	MOVL	#4, OLD STATUS	2066
57		02	D0	0001A	MOVL	#2, VBN	
		7E	7C	0001D	1\$: CLRQ	-(SP)	
		7E	D4	0001F	CLRL	-(SP)	
		57	DD	00021	PUSHL	VBN	
7E	0200	8F	3C	00023	MOVZWL	#512, -(SP)	
		56	DD	00028	PUSHL	R6	
		7E	7C	0002A	CLRL	-(SP)	
		20	AE	9F	PUSHAB	IOSB	
		31	DD	0002F	PUSHL	#49	
		46	AA	DD	PUSHL	INPUT_CHAN	
		7E	D4	00034	CLRL	-(SP)	
	00000000G	00	0C	FB	CALLS	#12, SYSSQIOW	
58		50	D0	0003D	MOVL	R0, STATUS	
06		58	E9	00040	BLBC	STATUS, 2\$	2067
58		6E	3C	00043	MOVZWL	IOSB, STATUS	
06		58	E8	00046	BLBS	STATUS, 3\$	2069
59		58	D0	00049	2\$: MOVL	STATUS, OLD_STATUS	2071
		01	11	31	BRW	12\$	
02	0D	A6	91	0004F	3\$: CMPB	13(R6), #2	2074
		06	12	00053	BNEQ	4\$	
57	10	A6	00	ED	CMPZV	#0, #16, 16(R6), VBN	2075
		03	13	0005B	4\$: BEQL	6\$	
		00A1	31	0005D	5\$: BRW	11\$	
		08	A6	D5	6\$: TSTL	8(R6)	2076
			F8	13	BEQL	5\$	
		0E	A6	B5	TSTW	14(R6)	2077
			F3	13	BEQL	5\$	
		10	A6	B5	TSTW	16(R6)	2078
			EE	13	BEQL	5\$	
		12	A6	B5	TSTW	18(R6)	2079
			E9	13	BEQL	5\$	

			14	A6	B5	00074		TSTW	20(R6)		2080
				E4	13	00077		BEQL	5\$		
			16	A6	B5	00079		TSTW	22(R6)		2081
				DF	13	0007C		BEQL	5\$		
			18	A6	D5	0007E		TSTL	24(R6)		2082
				7E	13	00081		BEQL	11\$		
			1C	A6	D5	00083		TSTL	28(R6)		2083
				79	13	00086		BEQL	11\$		
			20	A6	B5	00088		TSTW	32(R6)		2084
				74	13	0008B		BEQL	11\$		
			22	A6	B5	0008D		TSTW	34(R6)		2085
				6F	13	00090		BEQL	11\$		
				3A	DD	00092		PUSHL	#58		2086
				56	DD	00094		PUSHL	R6		
		6B		02	FB	00096		CALLS	#2, CHECKSUM2		
		65		50	E9	00099		BLBC	R0, 11\$		
		7E	01FE	8F	3C	0009C		MOVZWL	#510, -(SP)		2087
				56	DD	000A1		PUSHL	R6		
		6B		02	FB	000A3		CALLS	#2, CHECKSUM2		
		58		50	E9	000A6		BLBC	R0, 11\$		
		01	04	AC	D1	000A9		CMPL	RVN, #1		2090
				01	13	000AD		BEQL	7\$		
					04	000AF		RET			
		45	AA	02	90	000B0	7\$:	MOVB	#2, FAST_STRUCLEV		2093
		00FF	8F	28	A6	B1	000B4	CMPL	40(R6), #255		2094
					14	1B	000BA	BLEQU	8\$		
		7E	5A	AA	10	C1	000BC	ADDL3	#16, INPUT_QUAL, -(SP)		2095
					01	DD	000C1	PUSHL	#1		
					8F	DD	000C3	PUSHL	#BACKUP\$ MAXVOLS		
		00000000G	00	03	FB	000C9		CALLS	#3, LIB\$SIGNAL		
			6A	28	A6	90	000D0	8\$:	MOVB	40(R6), COM_I_SETCOUNT	2096
					03	12	000D4	BNEQ	9\$		2097
			6A	01	90	000D6		MOVB	#1, COM_I_SETCOUNT		
			1C	90	AA	E9	000D9	9\$:	BLBC	QUAL+14, TOS	2099
			6A	D1	AA	91	000DD	CMPL	QUAL+79, COM_I_SETCOUNT		2100
					16	1B	000E1	BLEQU	10\$		
			7E	6A	9A	000E3		MOVZBL	COM_I_SETCOUNT, -(SP)		2104
			7E	D1	AA	9A	000E6	MOVZBL	QUAL+79, -(SP)		
					02	DD	000EA	PUSHL	#2		2102
					8F	DD	000EC	PUSHL	#BACKUP\$ NOSUCHRVN		
		00000000G	00	04	FB	000F2		CALLS	#4, LIB\$SIGNAL		
			02	AA	01CC	C6	0C	28	000F9	10\$:	2107
							04	00100		RET	2089
			01	04	AC	D1	00101	11\$:	CMPL	RVN, #1	2113
					59	12	00105	BNEQ	12\$		
			01	0D	A6	91	00107	CMPL	13(R6), #1		2114
					53	12	0010B	BNEQ	12\$		
			01	08	A6	B1	0010D	CMPL	8(R6), #1		2115
					4D	12	00111	BNEQ	12\$		
					02	A6	D5	00113	TSTL	2(R6)	2116
					48	13	00116	BEQL	12\$		
			06	A6	B5	00118		TSTW	6(R6)		2117
					43	13	0011B	BEQL	12\$		
					66	B5	0011D	TSTW	(R6)		2118
					3F	13	0011F	BEQL	12\$		
					3A	DD	00121	PUSHL	#58		2119
					56	DD	00123	PUSHL	R6		

	6B		02	FB	00125	CALLS	#2, CHECKSUM2	:	
	35		50	E9	00128	BLBC	R0, 12\$:	
	7E	01FE	8F	3C	0012B	MOVZWL	#510, -(SP)	:	2120
			56	DD	00130	PUSHL	R6	:	
	6B		02	FB	00132	CALLS	#2, CHECKSUM2	:	
	28		50	E9	00135	BLBC	R0, 12\$:	
45	AA		01	90	00138	MOVB	#1, FAST_STRUCLEV	:	2123
	6A		01	90	0013C	MOVB	#1, COM_I_SETCOUNT	:	2124
	40	90	AA	E9	0013F	BLBC	QUAL+14, T3\$:	2126
	01	D1	AA	91	00143	CMPB	QUAL+79, #1	:	2127
			3A	1B	00147	BLEQU	13\$:	
	7E		6A	9A	00149	MOVZBL	COM_I_SETCOUNT, -(SP)	:	2131
	7E	D1	AA	9A	0014C	MOVZBL	QUAL+79, -(SP)	:	
			02	DD	00150	PUSHL	#2	:	2129
	00000000G	00	8F	DD	00152	PUSHL	#BACKUP\$ NOSUCHRVN	:	
			04	FB	00158	CALLS	#4, LIB\$SIGNAL	:	
			04	0015F		RET		:	2122
FEB3	57	01	00000064	8F	F1	00160	12\$: ACBL	#100, #1, VBN, 1\$	2058
				59	DD	0016A	PUSHL	OLD_STATUS	2139
			04	AC	DD	0016C	PUSHL	RVN	
	7E	5A	AA	10	C1	0016F	ADDL3	#16, INPUT_QUAL, -(SP)	
				02	DD	00174	PUSHL	#2	
	00000000G	00	8F	DD	00176	PUSHL	#BACKUP\$ PROCINDEX	:	
			05	FB	0017C	CALLS	#5, LIB\$SIGNAL	:	
			04	00183	13\$:	RET		:	2140

; Routine Size: 388 bytes. Routine Base: CODE + 09AF

```

1041 2141 1 %SBTTL 'VERIFY_HEADER - verify file header'
1042 2142 1 ROUTINE VERIFY_HEADER(HEADER,FILE_ID)=
1043 2143 1
1044 2144 1 |++
1045 2145 1 |
1046 2146 1 | FUNCTIONAL DESCRIPTION:
1047 2147 1 | This routine determines if the block given it is a valid file header.
1048 2148 1 |
1049 2149 1 | INPUT PARAMETERS:
1050 2150 1 | HEADER - Pointer to header.
1051 2151 1 | FILE_ID - Purported file ID.
1052 2152 1 |
1053 2153 1 | IMPLICIT INPUTS:
1054 2154 1 | NONE
1055 2155 1 |
1056 2156 1 | OUTPUT PARAMETERS:
1057 2157 1 | NONE
1058 2158 1 |
1059 2159 1 | IMPLICIT OUTPUTS:
1060 2160 1 | NONE
1061 2161 1 |
1062 2162 1 | ROUTINE VALUE:
1063 2163 1 | 0 if invalid file header
1064 2164 1 | 1 if valid file header
1065 2165 1 | 2 if deleted file header
1066 2166 1 |
1067 2167 1 | SIDE EFFECTS:
1068 2168 1 | NONE
1069 2169 1 |
1070 2170 1 | --
1071 2171 1 |
1072 2172 2 BEGIN
1073 2173 2 MAP
1074 2174 2 HEADER: REF BBLOCK, ! file header arg
1075 2175 2 FILE_ID: REF BBLOCK; ! file ID arg
1076 2176 2
1077 2177 2
1078 2178 2 ! First check the structure level.
1079 2179 2 !
1080 2180 2 IF .HEADER[FH2$B_STRUCLEV] NEQ .FAST_STRUCLEV
1081 2181 2 THEN
1082 2182 2 RETURN 0;
1083 2183 2
1084 2184 2
1085 2185 2 IF .FAST_STRUCLEV EQL 2
1086 2186 2 THEN
1087 2187 2 BEGIN
1088 2188 2
1089 2189 2 ! Check the area offsets and the retrieval pointer use counts for
1090 2190 2 ! consistency.
1091 2191 2 !
1092 2192 2 IF
1093 2193 2 .HEADER[FH2$B_IDOFFSET] LSSU $BYTEOFFSET (FH2$L_HIGHWATER)/2 OR
1094 2194 2 .HEADER[FH2$B_MPOFFSET] LSSU .HEADER[FH2$B_IDOFFSET] OR
1095 2195 2 .HEADER[FH2$B_ACOFFSET] LSSU .HEADER[FH2$B_MPOFFSET] OR
1096 2196 2 .HEADER[FH2$B_RSOFFSET] LSSU .HEADER[FH2$B_ACOFFSET] OR
1097 2197 2 .HEADER[FH2$B_MAP_INUSE] GTRU .HEADER[FH2$B_ACOFFSET] - .HEADER[FH2$B_MPOFFSET]

```

```

: 1098      2198      THEN
: 1099      2199      RETURN 0;
: 1100      2200
: 1101      2201
: 1102      2202      ! At this point, we have verified that the block at least once was a
: 1103      2203      ! valid file header.
: 1104      2204
: 1105      2205      ! Look at the file number in the header. If zero, this is a
: 1106      2206      ! deleted header.
: 1107      2207
: 1108      2208      IF
: 1109      2209      .HEADER[FH2$W_FID_NUM] EQL 0 AND
: 1110      2210      .HEADER[FH2$B_FID_NMX] EQL 0
: 1111      2211      THEN
: 1112      2212      RETURN 2;
: 1113      2213
: 1114      2214
: 1115      2215      ! Now compute the header checksum.
: 1116      2216
: 1117      2217      IF NOT CHECKSUM(.HEADER)
: 1118      2218      THEN
: 1119      2219      RETURN 2;
: 1120      2220
: 1121      2221
: 1122      2222      ! Check file number and file sequence number.
: 1123      2223
: 1124      2224      IF
: 1125      2225      .HEADER[FH2$W_FID_NUM] NEQ .FILE_ID[FID$W_NUM] OR
: 1126      2226      .HEADER[FH2$B_FID_NMX] NEQ .FILE_ID[FID$B_NMX] OR
: 1127      2227      .HEADER[FH2$W_FID_SEQ] NEQ .FILE_ID[FID$W_SEQ]
: 1128      2228      THEN
: 1129      2229      RETURN 2;
: 1130      2230      END
: 1131      2231      ELSE
: 1132      2232      BEGIN
: 1133      2233      LOCAL
: 1134      2234      MAP_AREA:      REF BBLOCK;
: 1135      2235
: 1136      2236
: 1137      2237      ! Now point to the map area and make sure that the extension
: 1138      2238      ! RVN is zero. Also check the retrieval pointer format data.
: 1139      2239
: 1140      2240      MAP_AREA = .HEADER + .HEADER[FH1$B_MPOFFSET]*2;
: 1141      2241      IF
: 1142      2242      .MAP_AREA[FM1$B_EX_RVN] NEQ 0 OR
: 1143      2243      .MAP_AREA[FM1$B_COUNTSIZE] NEQ 1 OR
: 1144      2244      .MAP_AREA[FM1$B_LBNSIZE] NEQ 3
: 1145      2245      THEN
: 1146      2246      RETURN 0;
: 1147      2247
: 1148      2248
: 1149      2249      ! Check the retrieval pointer counts for consistency with the
: 1150      2250      ! available space.
: 1151      2251
: 1152      2252      IF
: 1153      2253      .MAP_AREA[FM1$B_INUSE] GTRU .MAP_AREA[FM1$B_AVAIL] OR
: 1154      2254      .MAP_AREA[FM1$B_AVAIL] GTRU 255 = (.MAP_AREA + FM1$C_POINTERS - .HEADER) / 2
```


	02	A2	03	A2	91	00035		CMPB	3(R2), 2(R2)	2196
				03	1E	0003A	1\$:	BGEQU	2\$	
				0092	31	0003C		BRW	9\$	
	51		02	A2	9A	0003F	2\$:	MOVZBL	2(R2), R1	2197
	50		01	A2	9A	00043		MOVZBL	1(R2), R0	
	51			50	C2	00047		SUBL2	R0, R1	
51		3A		08	00	ED	0004A	CMPZV	#0, #8, 58(R2), R1	
					7F	1A	00050	BGTRU	9\$	
			08	A2	B5	00052		TSTW	8(R2)	2209
				05	12	00055		BNEQ	3\$	
			0D	A2	95	00057		TSTB	13(R2)	2210
				6D	13	0005A		BEQL	7\$	
				52	DD	0005C	3\$:	PUSHL	R2	2217
	64			01	FB	0005E		CALLS	#1, CHECKSUM	
	65			50	E9	00061		BLBC	R0, 7\$	
	63		08	A2	B1	00064		CMPW	8(R2), (R3)	2225
				5F	12	00068		BNEQ	7\$	
05	A3		0D	A2	91	0006A		CMPB	13(R2), 5(R3)	2226
				58	12	0006F		BNEQ	7\$	
02	A3		0A	A2	B1	00071		CMPW	10(R2), 2(R3)	2227
				4F	11	00076		BRB	6\$	
	50		01	A2	9A	00078	4\$:	MOVZBL	1(R2), R0	2240
	50			6240	3E	0007C		MOVAV	(R2)[R0], MAP_AREA	
			01	A0	95	00080		TSTB	1(MAP_AREA)	2242
				4C	12	00083		BNEQ	9\$	
	01		06	A0	91	00085		CMPB	6(MAP_AREA), #1	2243
				46	12	00089		BNEQ	9\$	
	03		07	A0	91	0008B		CMPB	7(MAP_AREA), #3	2244
				40	12	0008F	5\$:	BNEQ	9\$	
09	A0		08	A0	91	00091		CMPB	8(MAP_AREA), 9(MAP_AREA)	2253
				39	1A	00096		BGTRU	9\$	
		51		52	50	C3	00098	SUBL3	MAP_AREA, R2, R1	2254
				51	0A	C2	0009C	SUBL2	#10, R1	
				51	02	C6	0009F	DIVL2	#2, R1	
			00FF	C1	9E	000A2		MOVAB	255(R1), R1	
51		09	A0	08	00	ED	000A7	CMPZV	#0, #8, 9(MAP_AREA), R1	
					22	1A	000AD	BGTRU	9\$	
			02	A2	B5	000AF		TSTW	2(R2)	2265
				15	13	000B2		BEQL	7\$	
				52	DD	000B4		PUSHL	R2	2272
	64			01	FB	000B6		CALLS	#1, CHECKSUM	
	0D			50	E9	000B9		BLBC	R0, 7\$	
	63		02	A2	B1	000BC		CMPW	2(R2), (R3)	2280
				07	12	000C0		BNEQ	7\$	
02	A3		04	A2	B1	000C2		CMPW	4(R2), 2(R3)	2281
				04	13	000C7	6\$:	BEQL	8\$	
	50			02	D0	000C9	7\$:	MOVL	#2, R0	2283
					04	000CC		RET		
	50			01	D0	000CD	8\$:	MOVL	#1, R0	2289
					04	000D0		RET		
				50	D4	000D1	9\$:	CLRL	R0	2290
					04	000D3		RET		

: Routine Size: 212 bytes, Routine Base: CODE + 0B33

```
1192 2291 1 %SBTTL 'PROCESS_FILE - process selected file'
1193 2292 1 ROUTINE PROCESS_FILE: NOVALUE=
1194 2293 1
1195 2294 1 :++
1196 2295 1
1197 2296 1 : FUNCTIONAL DESCRIPTION:
1198 2297 1 : This routine is called when the next file that matches the selection
1199 2298 1 : file specification has been located. It completes the tests that
1200 2299 1 : select files to be processed, and if these are passed, processes the
1201 2300 1 : file.
1202 2301 1
1203 2302 1 : INPUT PARAMETERS:
1204 2303 1 : NONE
1205 2304 1
1206 2305 1 : IMPLICIT INPUTS:
1207 2306 1 : INPUT_NAM - Contains resultant string and file ID.
1208 2307 1
1209 2308 1 : OUTPUT PARAMETERS:
1210 2309 1 : NONE
1211 2310 1
1212 2311 1 : IMPLICIT OUTPUTS:
1213 2312 1 : NONE
1214 2313 1
1215 2314 1 : ROUTINE VALUE:
1216 2315 1 : NONE
1217 2316 1
1218 2317 1 : SIDE EFFECTS:
1219 2318 1 : File processed if appropriate.
1220 2319 1
1221 2320 1 :--
1222 2321 1
1223 2322 2 BEGIN
1224 2323 2 LOCAL
1225 2324 2 FILE_NUMBER, ! Clean file number
1226 2325 2 RVN; ! Clean RVN
1227 2326 2
1228 2327 2
1229 2328 2 ! Get clean file ID.
1230 2329 2
1231 2330 2 FILE_NUMBER = .INPUT_NAM[NAM$W_FID_NUM];
1232 2331 2 FILE_NUMBER<16,8> = .INPUT_NAM[NAM$B_FID_NMX];
1233 2332 2 RVN = .INPUT_NAM[NAM$B_FID_RVN];
1234 2333 2
1235 2334 2
1236 2335 2 IF .QUAL[QUAL_FAST]
1237 2336 2 THEN
1238 2337 2 BEGIN
1239 2338 2
1240 2339 2 ! First, make sure the RVN is in range. Then, make sure the file number
1241 2340 2 ! is in range.
1242 2341 2
1243 2342 2 IF .RVN GTRU .COM I SETCOUNT THEN RETURN;
1244 2343 2 IF .FILE_NUMBER GTRD .FAST_IMAP_SIZE[.RVN-1]*4096 THEN RETURN;
1245 2344 2
1246 2345 2
1247 2346 2 ! See if file is selected.
1248 2347 2 !
```

```

: 1249      2348 3      IF .QUAL[QUAL_IMAG]
: 1250      2349 3      THEN
: 1251      2350 4      BEGIN
: 1252      2351 4      IF NOT .BITVECTOR[.FAST_IMAP[.RVN-1], .FILE_NUMBER-1]
: 1253      2352 4      THEN
: 1254      2353 4      RETURN;
: 1255      2354 4      END
: 1256      2355 3      ELSE
: 1257      2356 4      BEGIN
: 1258      2357 4      IF
: 1259      2358 4      NOT .BITVECTOR[.FAST_IMAP[.RVN-1], .FILE_NUMBER-1] AND
: 1260      2359 5      NOT (.DIR_STATUS[.STAT_SCANNED] AND .QUAL[QUAL_OSAV])
: 1261      2360 4      THEN
: 1262      2361 4      RETURN;
: 1263      2362 3      END;
: 1264      2363 2      END;
: 1265      2364 2
: 1266      2365 2
: 1267      2366 2      ! Finish evaluating selection criteria.
: 1268      2367 2
: 1269      2368 2      IF
: 1270      2369 2      NOT .QUAL[QUAL_IMAG] AND
: 1271      2370 3      NOT (.DIR_STATUS[.STAT_SCANNED] AND .QUAL[QUAL_OSAV])
: 1272      2371 2      THEN
: 1273      2372 2      IF NOT SELECT_INPUT_FILE(%B'010') THEN RETURN;
: 1274      2373 2
: 1275      2374 2
: 1276      2375 2      ! File is selected. Process it. If successfully processed in image mode,
: 1277      2376 2      ! clear the bitmap bit to avoid processing its synonyms.
: 1278      2377 2
: 1279      2378 2      IF SAVE_ONE_FILE()
: 1280      2379 2      THEN
: 1281      2380 2      IF .QUAL[QUAL_IMAG]
: 1282      2381 2      THEN
: 1283      2382 2      BITVECTOR[.FAST_IMAP[.RVN-1], .FILE_NUMBER-1] = FALSE;
: 1284      2383 1      END;

```

		001C 00000 PROCESS_FILE:						
					.WORD	Save R2,R3,R4	: 2292	
		54	00000000'	EF 9E 00002	MOVAB	QUAL+8, R4	: 2330	
		50	00C8	C4 D0 00009	MOVL	INPUT_NAM, R0	: 2331	
52	08	52	24	A0 3C 0000E	MOVZWL	36(R0), FILE_NUMBER	: 2332	
		10	29	A0 F0 00012	INSV	41(R0), #16, #8, FILE_NUMBER	: 2335	
		53	28	A0 9A 00018	MOVZBL	40(R0), RVN	: 2342	
		A4		06 E1 0001C	BBC	#6, QUAL+9, 2\$: 2343	
53	76	08		00 ED 00021	CMPZV	#0, #8, COM_I_SETCOUNT, RVN	: 2351	
				6D 1F 00027	BLSSU	5\$: 2351	
		50	0234 D443	DE 00029	MOVAL	@FAST_IMAP_SIZE[RVN], R0	: 2351	
		A0		0C 78 0002F	ASHL	#12, =4(R0), R0	: 2351	
		50		52 D1 00034	CMPL	FILE_NUMBER, R0	: 2351	
				5D 1A 00037	BGTRU	5\$: 2351	
		50	0238 D443	DE 00039	MOVAL	@FAST_IMAP[RVN], R0	: 2351	
		51	FF	A2 9E 0003F	MOVAB	-1(R2), R1	: 2351	

FASTSCAN
V04-000

Fast file scan
PROCESS_FILE - process selected file

E 6
15-Sep-1984 23:56:53
14-Sep-1984 11:53:52

VAX-11 Bliss-32 V4.0-742
[BACKUP.SRC]FASTSCAN.B32;1

Page 47
(7)

06	02	A4	03	E1	00043	BBC	#3, QUAL+10, 1\$:	2348
11	FC	B0	51	E0	00048	BBS	R1, @-4(R0), 2\$:	2351
				04	0004D	RET		:	2353
08	FC	B0	51	E0	0004E	BBS	R1, @-4(R0), 2\$:	2358
3D	0297	C4	02	E1	00053	BBC	#2, DIR_STATUS, 5\$:	2359
			07	A4	95 00059	TSTB	QUAL+15	:	
			38	18	0005C	BGEQ	5\$:	
17	02	A4	03	E0	0005E	BBS	#3, QUAL+10, 4\$:	2369
05	0297	C4	02	E1	00063	BBC	#2, DIR_STATUS, 3\$:	2370
			07	A4	95 00069	TSTB	QUAL+15	:	
			0C	19	0006C	BLSS	4\$:	
			02	DD	0006E	PUSHL	#2	:	2372
	00000000G	00	01	FB	00070	CALLS	#1, SELECT_INPUT_FILE	:	
		1C	50	E9	00077	BLBC	R0, 5\$:	
	00000000G	00	00	FB	0007A	CALLS	#0, SAVE_ONE_FILE	:	2378
		12	50	E9	00081	BLBC	R0, 5\$:	
0D	02	A4	03	E1	00084	BBC	#3, QUAL+10, 5\$:	2380
		50	0238	D443	DE 00089	MOVAL	@FAST_IMAP(RVN), R0	:	2382
				52	D7 0008F	DECL	R2	:	
00	FC	B0	52	E5	00091	BBCC	R2, @-4(R0), 5\$:	
			04	00096	5\$:	RET		:	2383

; Routine Size: 151 bytes, Routine Base: CODE + 0C07

```

: 1286      2384 1 %SBTTL 'DIR_SCAN - scan a directory'
: 1287      2385 1 ROUTINE DIR_SCAN(RVN): NOVALUE=
: 1288      2386 1
: 1289      2387 1 !++
: 1290      2388 1
: 1291      2389 1 : FUNCTIONAL DESCRIPTION:
: 1292      2390 1 :   This routine is the driver for a directory scan.
: 1293      2391 1
: 1294      2392 1 : INPUT PARAMETERS:
: 1295      2393 1 :   RVN           - Relative volume number.
: 1296      2394 1
: 1297      2395 1 : IMPLICIT INPUTS:
: 1298      2396 1 :   NONE
: 1299      2397 1
: 1300      2398 1 : OUTPUT PARAMETERS:
: 1301      2399 1 :   NONE
: 1302      2400 1
: 1303      2401 1 : IMPLICIT OUTPUTS:
: 1304      2402 1 :   NONE
: 1305      2403 1
: 1306      2404 1 : ROUTINE VALUE:
: 1307      2405 1 :   NONE
: 1308      2406 1
: 1309      2407 1 : SIDE EFFECTS:
: 1310      2408 1 :   NONE
: 1311      2409 1
: 1312      2410 1 :--
: 1313      2411 1
: 1314      2412 2 BEGIN
: 1315      2413 2 FAST_RVN = .RVN;
: 1316      2414 2 INIT_DIR_SCAN(
: 1317      2415 2   .INPUT_CHAN,
: 1318      2416 2   .INPUT_NAM,
: 1319      2417 2   INPUT_QUAL[QUAL_DEV_DESC],
: 1320      2418 2   INPUT_QUAL[QUAL_EXP_DESC],
: 1321      2419 2   .QUAL[QUAL_IMAG] + .QUAL[QUAL_OSAV]^2,
: 1322      2420 2   .RVN,
: 1323      2421 2   0);
: 1324      2422 2 WHILE FIND_NEXT() DO PROCESS_FILE();
: 1325      2423 2 FREE_DIR_DATA();
: 1326      2424 1 END;

```

0004 0000 DIR_SCAN:

```

: 2385      .WORD   Save R2
: 2413      MOVAB   INPUT_QUAL, R2
: 2418      MOVB    RVN, FAST_RVN
: 2420      CLRL    -(SP)
: 2419      PUSHL   RVN
:          EXTZV  #3, #1, QUAL+10, R1
:          EXTZV  #7, #1, QUAL+15, R0
:          PUSHAL (R1)[R0]
: 2418      ADDL3   #8, INPUT_QUAL, -(SP)
: 2417      ADDL3   #16, INPUT_QUAL, -(SP)

```

FASTSCAN
V04-000

Fast file scan
DIR_SCAN - scan a directory

G 6
15-Sep-1984 23:56:53
14-Sep-1984 11:53:52

VAX-11 Bliss-32 V4.0-742
[BACKUP.SRC]FASTSCAN.B32;1

Page 49
(8)

		FB	A2	DD	0002D		PUSHL	INPUT_NAM	:	2418
		EC	A2	DD	00030		PUSHL	INPUT_CHAN	:	
0000V	CF		07	FB	00033		CALLS	#7, INIT_DIR_SCAN	:	
0000V	CF		00	FB	00038	1\$:	CALLS	#0, FIND_NEXT	:	2422
	07		50	E9	0003D		BLBC	R0, 2\$:	
FF24	CF		00	FB	00040		CALLS	#0, PROCESS_FILE	:	
			F1	11	00045		BRB	1\$:	
0000V	CF		00	FB	00047	2\$:	CALLS	#0, FREE_DIR_DATA	:	2423
			04	0004C			RET		:	2424

; Routine Size: 77 bytes, Routine Base: CODE + 0C9E

```

1328 2425 1 %SBTTL 'INIT DIR SCAN - initialize directory scan'
1329 2426 1 GLOBAL ROUTINE INIT_DIR_SCAN(CHAN,NAM,DEV_DESC,SEL_DESC,FLAGS,RVN,LIMIT): NOVALUE=
1330 2427 1
1331 2428 1 !++
1332 2429 1
1333 2430 1 ! FUNCTIONAL DESCRIPTION:
1334 2431 1 ! This routine initializes context for a directory scan.
1335 2432 1
1336 2433 1 ! INPUT PARAMETERS:
1337 2434 1 ! CHANNEL - Channel assigned to volume set.
1338 2435 1 ! NAM - Pointer to name block.
1339 2436 1 ! DEV_DESC - Pointer to device name descriptor.
1340 2437 1 ! SEL_DESC - Pointer to selection filespec descriptor.
1341 2438 1 ! FLAGS - Bit 0 true to request an image scan.
1342 2439 1 ! Bit 1 true to request immediate return on terminator.
1343 2440 1 ! Bit 2 true to request return of scanned directories.
1344 2441 1 ! RVN - Relative volume number.
1345 2442 1 ! LIMIT - Pointer to vector of ODS-1 scan limits or 0.
1346 2443 1
1347 2444 1 ! IMPLICIT INPUTS:
1348 2445 1 ! NONE
1349 2446 1
1350 2447 1 ! OUTPUT PARAMETERS:
1351 2448 1 ! NONE
1352 2449 1
1353 2450 1 ! IMPLICIT OUTPUTS:
1354 2451 1 ! NONE
1355 2452 1
1356 2453 1 ! ROUTINE VALUE:
1357 2454 1 ! NONE
1358 2455 1
1359 2456 1 ! SIDE EFFECTS:
1360 2457 1 ! NONE
1361 2458 1
1362 2459 1 !--
1363 2460 1
1364 2461 2 BEGIN
1365 2462 2 MAP
1366 2463 2 DEV_DESC : REF VECTOR, ! Device name descriptor
1367 2464 2 NAM : REF BBLOCK; ! Pointer to name block
1368 2465 2 LOCAL
1369 2466 2 STATUS, ! General status value
1370 2467 2 LOCAL_FAB : $FAB_DECL, ! FAB for $PARSE
1371 2468 2 LOCAL_NAM : $NAM_DECL; ! NAM for $PARSE
1372 2469 2
1373 2470 2
1374 2471 2 ! Initialize the impure area.
1375 2472 2 !
1376 2473 2 CH$FILL(0, DIR_END-DIR_BEG, DIR_BEG);
1377 2474 2 DIR_FLAGS[D_INITIAL] = TRUE;
1378 2475 2
1379 2476 2
1380 2477 2 ! Save the parameters.
1381 2478 2 !
1382 2479 2 DIR_CHAN = .CHAN;
1383 2480 2 DIR_NAM = .NAM;
1384 2481 2 DIR_NAM[NAM$W_DID_NUM] = FIDSC_MFD;

```

```

: 1385      2482  2 DIR_NAM[NAMSW_DID_SEQ] = FIDSC_MFD;
: 1386      2483  2 DIR_NAM[NAMSW_DID_RVN] = .RVN;
: 1387      2484  2 DIR_DEV_DESC = .DEV_DESC;
: 1388      2485  2
: 1389      2486  2 ! Determine if a rooted directory is being used. If so, do another
: 1390      2487  2 ! parse to get the root directory ID.
: 1391      2488  2
: 1392      2489  2
: 1393      2490  2 IF .NAM[NAMSV_ROOT_DIR]
: 1394      2491  2 THEN
: 1395      2492  2 BEGIN
: 1396      2493  2   $FAB_INIT (FAB = LOCAL_FAB,
: 1397      2494  2             NAM = LOCAL_NAM,
: 1398      2495  2             FNS = .DEV_DESC[0],
: 1399      2496  2             FNA = .DEV_DESC[1],
: 1400      2497  2             DNM = '[000000]')
: 1401      2498  2   );
: 1402      2499  2   $NAM_INIT (NAM = LOCAL_NAM,
: 1403      2500  2             NOP = NOCONCEAL
: 1404      2501  2   );
: 1405      2502  2   STATUS = $PARSE (FAB = LOCAL_FAB);
: 1406      2503  2   IF NOT .STATUS
: 1407      2504  2   THEN SIGNAL (BACKUP$_OPENIN+STSS$_SEVERE, 1, .DEV_DESC, .STATUS, .LOCAL_FAB[FAB$_STV]);
: 1408      2505  2
: 1409      2506  2   DIR_NAM[NAMSW_FID_NUM] = .LOCAL_NAM[NAMSW_DID_NUM];
: 1410      2507  2   DIR_NAM[NAMSW_FID_SEQ] = .LOCAL_NAM[NAMSW_DID_SEQ];
: 1411      2508  2   DIR_NAM[NAMSW_FID_RVN] = .LOCAL_NAM[NAMSW_DID_RVN];
: 1412      2509  2   IF .DIR_NAM[NAMSB_FID_RVN] EQL 0 THEN DIR_NAM[NAMSB_FID_RVN] = .RVN;
: 1413      2510  2   END
: 1414      2511  2
: 1415      2512  2
: 1416      2513  2 ! If no root directory is used, establish the MFD as the root.
: 1417      2514  2
: 1418      2515  2 ELSE
: 1419      2516  2 BEGIN
: 1420      2517  2   DIR_NAM[NAMSW_FID_NUM] = FIDSC_MFD;
: 1421      2518  2   DIR_NAM[NAMSW_FID_SEQ] = FIDSC_MFD;
: 1422      2519  2   DIR_NAM[NAMSW_FID_RVN] = .RVN;
: 1423      2520  2   END;
: 1424      2521  2
: 1425      2522  2 ! Initialize the result string in the name block with the name of
: 1426      2523  2 ! the MFD. Note that in the case of a rooted directory, we are lying
: 1427      2524  2 ! about its name. However, this is consistent with RMS behavior and
: 1428      2525  2 ! is generally all to the best.
: 1429      2526  2
: 1430      2527  2
: 1431      2528  2 CH$COPY (.DEV_DESC[0], .DEV_DESC[1],
: 1432      2529  2          %CHARCOUNT ('[000000]000000.DIR;1'),
: 1433      2530  2          UPLIT BYTE ('[000000]000000.DIR;1'),
: 1434      2531  2          .NAM[NAMSB_RSS],
: 1435      2532  2          .NAM[NAMSL_RSA]
: 1436      2533  2          );
: 1437      2534  2
: 1438      2535  2 NAM[NAMSB_RSL] = MINU (.NAM[NAMSB_RSS], .DEV_DESC[0] + %CHARCOUNT ('[000000]000000.DIR;1'));
: 1439      2536  2 INIT_NAMEBLOCK (.NAM);
: 1440      2537  2
: 1441      2538  2 ! Initialize the level stack with the root of the selected RVN.

```

```

: 1442 2539 2 !
: 1443 2540 2 DIR_SP = DIR_STACK;
: 1444 2541 2 DIR_LEVELS = -1;
: 1445 2542 2 RBLOCK[DIR_STACK[D_FID], FIDSW_NUM] = .DIR_NAM[NAMSW_FID_NUM];
: 1446 2543 2 JBLOCK[DIR_STACK[D_FID], FIDSW_SEQ] = .DIR_NAM[NAMSW_FID_SEQ];
: 1447 2544 2 BBLOCK[DIR_STACK[D_FID], FIDSW_RVN] = .DIR_NAM[NAMSW_FID_RVN];
: 1448 2545 2 DIR_STACK[D_VBN] = -1;
: 1449 2546 2
: 1450 2547 2 ! Establish the selection file specification. This will also set the
: 1451 2548 2 ! terminator and status bits for the root level.
: 1452 2549 2
: 1453 2550 2 RESET_DIR_SPEC(.SEL_DESC, .FLAGS);
: 1454 2551 2 IF .LIMIT NEQ 0 THEN CHSMOVE(D_K_NLEVELS*XUPVAL, .LIMIT, DIR_SCANLIMIT);
: 1455 2552 2
: 1456 2553 2 ! Special cases for root directories:
: 1457 2554 2 ! the root directory is not saved initially.
: 1458 2555 2 ! If the directory pointed to by DIR_SEL_DIR is '000000' then
: 1459 2556 2 ! we want to save the MFD only.
: 1460 2557 2
: 1461 2558 2
: 1462 2559 2 IF .NAM[NAMSV_ROOT_DIR]
: 1463 2560 2 THEN
: 1464 2561 2 BEGIN
: 1465 2562 2 IF
: 1466 2563 2 CHSEQL ( %CHARCOUNT ('000000'),
: 1467 2564 2 UPLIT BYTE ('000000'),
: 1468 2565 2 .DIR_SEL_DIR [DSCSW_LENGTH],
: 1469 2566 2 .DIR_SEL_DIR [DSCSA_POINTER],
: 1470 2567 2 %C ' ')
: 1471 2568 2 THEN DIR_FLAGS[D_ROOT_MFD] = TRUE ;
: 1472 2569 2 DIR_FLAGS[D_INITIAL] = FALSE;
: 1473 2570 2 END;
: 1474 2571 2
: 1475 2572 1 END;

```

2E	30	30	30	30	30	30	30	30	30	30	30	30	5B	00CEB	P.AAD:	.ASCII	\[000000]\	:
													5B	00CF3	P.AAE:	.ASCII	\[000000]000000.DIR;1\	:
														00D02				:
														00D07	P.AAF:	.ASCII	\000000\	:
																.EXTRN	SYSSPARSE	:
														07FC	00000	.ENTRY	INIT_DIR_SCAN, Save R2,R3,R4,R5,R6,R7,R8,-	2426
																MOVAB	R9,R10	:
																MOVAB	DIR_NAM, R10	:
03CC	8F		00													MOVAB	-178(SP), SP	:
																MOVCS	#0, (SP), #0, #972, DIR_BEG	2473
																		:
				1A	AA											BISB2	#16, DIR_FLAGS	2474
				FC	AA		04									MOVL	CHAN, DIR_CHAN	2479
					58		08									MOVL	NAM, R8	2480
					6A											MOVL	R8, DIR_NAM	:
					56											MOVL	DIR_NAM, R6	2481
				2A	A6		00040004									MOVL	#262148, 42(R6)	:
				2E	A6		18									MOVW	RVN, 46(R6)	2483

0050	8F	7B 00	04 35	57 AA A8 6E	0C	AC 57 05 00	DO DO E1 2C	00037 00038 0003F 00044	MOVL MOVL BBC MOVCS	DEV_DESC, R7 R7, DIR_DEV_DESC #5, 53(R8), -2\$ #0, (SP), #0, #80, \$RMS_PTR	2484 2490 2498
			60 76 7F D8 DC E0 E4 E5	AE AE AE AD AD AD AD AD	60 5003	8F 02 02 6E A7 CF 67 08	BO 90 90 9E DO 9E 90 90	0004B 0004D 00053 00057 0005B 0005F 00064 0006A 0006E	MOVW MOVB MOVB MOVAB MOVL MOVAB MOVB MOVB	#20483, \$RMS_PTR #2, \$RMS_PTR+22 #2, \$RMS_PTR+31 LOCAL_NAM, \$RMS_PTR+40 4(R7), \$RMS_PTR+44 P.AAD, \$RMS_PTR+48 (R7), \$RMS_PTR+52 #8, \$RMS_PTR+53	
0060	8F	00		6E AE	6002	8F 10	BO 90	0007A 0007F	MOVW MOVB	#24578, \$RMS_PTR #16, \$RMS_PTR+8	2501
		00000000G		00 16	60	AE 01 50	9F FB EB	00083 00086 0008D	PUSHAB CALLS BLBS	LOCAL_FAB #1, SYSSPARSE STATUS, 1\$	2502 2503 2504
					6C	AE 50 57 01	DD DD DD	00090 00093 00095 00097	PUSHL PUSHL PUSHL	LOCAL_FAB+12 STATUS R7 #1	
		00000000G		00	00000000G	8F	DD	00099	PUSHL	#BACKUP\$ OPENIN+4	
			24 28	A0 A0	2A 2E 28	6A AE A0	DO DO 95	000A6 000A9 000AE	CALLS MOVL MOVW	#5, LIBSSIGNAL DIR_NAM, R0 LOCAL_NAM+42, 36(R0) LOCAL_NAM+46, 40(R0)	2506 2508 2509
			28	A0	18	14 AC	12 90	000B6 000B8	TSTB BNEQ	40(R0) 3\$	
			28	A6 A6 59	00040004 18	0D 8F AC	11 DO BO	000BD 000BF 000C7	MOVW BRB MOVL	RVN, 40(R0) 3\$ #262148, 36(R6)	2490 2517 2519
59		20	04	B7	04	A8 A8	9A DO	000CC 000D0	MOVZBL MOVL	2(R8), R9 4(R8), R6	2532 2533
						67 66 0E	2C 000DA 18	000D4 000DA 000DB	MOVCS BGEQ	(R7), R6 4\$	
				56 59		67 67	C0 C2	000DD 000E0	ADDL2 SUBL2	(R7), R6 (R7), R9	
59		20	FEFE	CF		14 66	2C 000EA	000E3 000EA	MOVCS	#20, P.AAE, #32, R9, (R6)	
				67 50 51		14 A8	C1 9A	000EB 000EF	ADDL3 MOVZBL	#20, (R7), R1 2(R8), R0	2535
				51	02	50 03	D1 1B	000F3 000F6	CMP BLEQU	R0, R1 5\$	
				50 03		51 50	DO 90	000F8 000FB	MOVL MOVB	R1, R0 R0, 3(R8)	
						58	DD	000FF	PUSHL	R8	2536
		00000000G		0G		01	FB	00101	CALLS	#1, INIT_NAMEBLOCK	
		03C0		CA	015C	CA	9E	00108	MOVAB	DIR_STACK, DIR_SP	2540
		19		AA		01	90	0010F	MOVB	#1, DIR_LEVELS	2541
				50		6A	DO	00113	MOVL	DIR_NAM, R0	2542
				0178	24	A0	DO	00116	MOVL	36(R0), DIR_STACK+28	
				017C	28	A0	BO	0011C	MOVW	40(R0), DIR_STACK+32	2544

FASTSCAN
V04-000

Fast file scan
INIT_DIR_SCAN - initialize directory scan

L 6
15-Sep-1984 23:56:53
14-Sep-1984 11:53:52

VAX-11 Bliss-32 V4.0-742
[BACKUP.SRC]FASTSCAN.B32;1

Page 54
(9)

			0170	CA		01	D0	00122		MOVL	#1, DIR_STACK+20	:	2545
				7E	10	AC	7D	00127		MOVQ	SEL_DEST, -(SP)	:	2550
			0000V	CF		02	FB	0012B		CALLS	#2, RESEf_DIR_SPEC	:	
					1C	AC	D5	00130		TSTL	LIMIT	:	2551
						07	13	00133		BEQL	6\$:	
		03C8	CA	1C	BC	24	28	00135		MOVCL	#36, @LIMIT, DIR_SCANLIMIT	:	
			14	35	AB	05	E1	0013C	6\$:	BBC	#5, 53(R8), 8\$:	2559
08	AA		20	FEB4	CF	06	2D	00141		CMPC5	#6, P.AAF, #32, DIR_SEL_DIR, @DIR_SEL_DIR+4	:	2563
					0C	BA		00149				:	
						04	12	0014B		BNEQ	7\$:	
			1A	AA		20	88	0014D		BISB2	#32, DIR_FLAGS	:	2568
			1A	AA		10	8A	00151	7\$:	BICB2	#16, DIR_FLAGS	:	2569
						04	00155	8\$:		RET		:	2572

; Routine Size: 342 bytes, Routine Base: CODE + 0D0D


```

1477 2573 1 %SBTTL 'RESET DIR SPEC - reinitialize directory context'
1478 2574 1 GLOBAL ROUTINE RESET_DIR_SPEC(SEL_DESC,FLAGS): NOVALUE=
1479 2575 1
1480 2576 1 !++
1481 2577 1
1482 2578 1 FUNCTIONAL DESCRIPTION:
1483 2579 1 This routine reinitializes context to change the selection file spec.
1484 2580 1
1485 2581 1 INPUT PARAMETERS:
1486 2582 1 SEL_DESC - Pointer to selection filespec descriptor.
1487 2583 1 FLAGS - Bit 0 true to request an image scan.
1488 2584 1 Bit 1 true to request immediate return on terminator.
1489 2585 1 Bit 2 true to request return of scanned directories.
1490 2586 1
1491 2587 1 IMPLICIT INPUTS:
1492 2588 1 NONE
1493 2589 1
1494 2590 1 OUTPUT PARAMETERS:
1495 2591 1 NONE
1496 2592 1
1497 2593 1 IMPLICIT OUTPUTS:
1498 2594 1 NONE
1499 2595 1
1500 2596 1 ROUTINE VALUE:
1501 2597 1 NONE
1502 2598 1
1503 2599 1 SIDE EFFECTS:
1504 2600 1 NONE
1505 2601 1
1506 2602 1 --
1507 2603 1
1508 2604 2 BEGIN
1509 2605 2 INIT SEL_INFO(.SEL_DESC, DIR_SEL_DIR, DIR_SEL_NTV, DIR_SEL_LATEST);
1510 2606 2 DIR_FLAGS[D_IMAGE_SCAN] = .FLAGS<0,1>;
1511 2607 2 DIR_FLAGS[D_HARD_STOP] = .FLAGS<1,1>;
1512 2608 2
1513 2609 2 DIR_FLAGS[D_SCANNED_DIRS] = FALSE;
1514 2610 2 IF
1515 2611 2 NOT .QUAL[QUAL INTE] AND
1516 2612 2 .DIR_SEL_NTV[DSC$W_LENGTH] EQL 5 AND
1517 2613 2 CH$EQL(5, .DIR_SEL_NTV[DSC$A_POINTER], 5, UPLIT BYTE ('*.*;*))
1518 2614 2 THEN
1519 2615 2 DIR_FLAGS[D_SCANNED_DIRS] = .FLAGS<2,1>;
1520 2616 2
1521 2617 2 IF .DIR_FLAGS[D_IMAGE_SCAN] THEN DIR_SEL_LATEST = +1;
1522 2618 2 CH$FILL(7-1, D_K_NLEVELS*%UPVAL, DIR_SCANLIMIT);
1523 2619 2
1524 2620 2
1525 2621 2 INCRA D FROM DIR_STACK TO .DIR_SP BY D_S_ENTRY DO
1526 2622 2 BEGIN
1527 2623 2 MAP
1528 2624 2 D: REF BBLOCK; ! Pointer to stack entry
1529 2625 2 LOCAL
1530 2626 2 STATUS, ! Status return
1531 2627 2 DESC: VECTOR[2]; ! Descriptor for directory string
1532 2628 2
1533 2629 2

```

```

: 1534      2630      3      | Establish the descriptor for the directory string at the current level.
: 1535      2631      3      |
: 1536      2632      3      IF .D EQLA DIR_STACK
: 1537      2633      3      THEN
: 1538      2634      4      BEGIN
: 1539      2635      4      DESC[0] = %CHARCOUNT('000000');
: 1540      2636      4      DESC[1] = UPLIT BYTE('000000');
: 1541      2637      4      END
: 1542      2638      3      ELSE
: 1543      2639      4      BEGIN
: 1544      2640      4      IF .D EQLA .DIR_SP
: 1545      2641      4      THEN DESC[0] = .DIR_STRING[0]
: 1546      2642      4      ELSE DESC[0] = .BLOCK[D + D_S_ENTRY, D_SAV_LEN];
: 1547      2643      4      DESC[1] = DIR_STRING[1];
: 1548      2644      3      END;
: 1549      2645      3      |
: 1550      2646      3      | Allocate the dynamic areas for this level.
: 1551      2647      3      |
: 1552      2648      3      |
: 1553      2649      3      BBLOCK[D[D_TERM_DESC], DCSA_POINTER] = GET_VM(DIR$$_NAME);
: 1554      2650      3      BBLOCK[D[D_NAME_DESC], DCSA_POINTER] = GET_VM(DIR$$_NAME);
: 1555      2651      3      |
: 1556      2652      3      |
: 1557      2653      3      | Establish the match bit and the terminator specification for this level.
: 1558      2654      3      |
: 1559      2655      3      STATUS = MATCH_DIRECTORY(
: 1560      2656      3      DESC,
: 1561      2657      3      DIR_SEL DIR,
: 1562      2658      3      D[D_TERM_DESC],
: 1563      2659      3      D[D_TERM_VER],
: 1564      2660      3      DIR_SEL NTV);
: 1565      2661      3      IF .STATUS<0,1> THEN D[D_DIR_MATCHES] = TRUE;
: 1566      2662      3      IF .STATUS<2,1> THEN D[D_WILD_TERM] = TRUE;
: 1567      2663      3      IF .STATUS<3,1> THEN D[D_NON_TERM] = TRUE;
: 1568      2664      3      |
: 1569      2665      2      END;
: 1570      2666      1      END;

```

```

30 30 30 30 30 30 00E63 P.AAG: .ASCII \*.**\
00E68 P.AAH: .ASCII \000000\

```

				00FC 0000	.ENTRY	RESET DIR SPEC, Save R2,R3,R4,R5,R6,R7	: 2574
	57	00000000G	00	9E 00002	MOVAB	GET_VM, R7	:
	56	00000000'	EF	9E 00009	MOVAB	DIR_FLAGS, R6	:
	5E		08	C2 00010	SUBL2	#8, SP	:
		03AA	C6	9F 00013	PUSHAB	DIR_SEL_LATEST	: 2605
		F6	A6	9F 00017	PUSHAB	DIR_SEL_NTV	:
		EE	A6	9F 0001A	PUSHAB	DIR_SEL_DIR	:
		04	AC	DD 0001D	PUSHL	SEL_DESC	:
		00000000G	00	04 FB 00020	CALLS	#4, INIT_SEL_INFO	:
66		01	00	08 AC FO 00027	INSV	FLAGS, #0, #T, DIR_FLAGS	: 2606
50	08	AC	01	01 EF 0002D	EXTZV	#1, #1, FLAGS, R0	: 2607
66		01	02	50 FO 00033	INSV	R0, #2, #1, DIR_FLAGS	:

			66		08	8A	00038	BICB2	#8, DIR_FLAGS	2609		
	19	FD70	C6		01	E0	0003B	EBS	#1, QUAC+14, 1\$	2611		
			05	F6	A6	B1	00041	(MPW	DIR_SEL_NTV, #5	2612		
					13	12	00045	BNEQ	1\$	2613		
	AB	AF	FA	B6	05	29	00047	CMFC3	#5, @DIR_SEL_NTV+4, P.AAG	2613		
					0B	12	0004D	BNEQ	1\$	2615		
50	08	AC	01		02	EF	0004F	EXTZV	#2, #1, FLAGS, R0	2615		
66		01	03		50	F0	00055	INSV	R0, #3, #1, DIR_FLAGS	2617		
			05		66	E9	0005A	1\$:	BLBC	DIR_FLAGS, 2\$	2617	
			03AA		01	D0	0005D	MOV	#1, DIR_SEL_LATEST	261E		
24	FF	8F	6E		00	2C	00062	2\$:	MOVCS	#0, (SPT), #=1, #36, DIR_SCANLIMIT	2621	
				03AE	C6		00068					
			52	0142	C6	9E	0006B	MOVAB	DIR_STACK, R2	2621		
			53	03A6	C6	D0	00070	MOVL	DIR_SP, R3	2632		
					72	11	00075	BRB	11\$	2632		
			50	0142	C6	9E	00077	3\$:	MOVAB	DIR_STACK, R0	2632	
			50		52	D1	0007C	CMPL	D, R0	2635		
			6E		0B	12	0007F	BNEQ	4\$	2636		
			04	AE	FF72	CF	9E	00084	MOV	#6, DESC	2636	
					16	11	0008A	MOVAB	P.AAH, DESC+4	2632		
			03A6	C6		52	D1	0008C	4\$:	CMPL	D, DIR_SP	2640
			6E	02	A6	9A	00093	BNEQ	5\$	2641		
					04	11	00097	MOVZBL	DIR_STRING, DESC	2642		
			6E	66	A2	9A	00099	5\$:	BRB	6\$	2642	
			04	AE	03	A6	9E	0009D	6\$:	MOVZBL	102(D), DESC	2643
			7E	50	8F	9A	000A2	7\$:	MOVAB	DIR_STRING+1, DESC+4	2649	
			67		01	FB	000A6	MOVZBL	#80, -(SP)	2650		
			28	A2	50	D0	000A9	CALLS	#1, GET_VM	2650		
			7E		50	8F	9A	000AD	MOVL	R0, 40(D)	2650	
			67		01	FB	000B1	MOVZBL	#80, -(SP)	2650		
			30	A2	50	D0	000B4	CALLS	#1, GET_VM	2659		
					F6	A6	9F	000B8	MOVL	R0, 48(D)	2659	
					38	A2	9F	000BB	PUSHAB	DIR_SEL_NTV	2659	
					24	A2	9F	000BE	PUSHAB	56(D)	2658	
					EE	A6	9F	000C1	PUSHAB	36(R2)	2658	
					10	AE	9F	000C4	PUSHAB	DIR_SEL_DIR	2655	
			00000000G	00	05	FB	000C7	PUSHAB	DESC	2659		
				04	50	E9	000CE	CALLS	#5, MATCH_DIRECTORY	2659		
			23	A2	02	88	000D1	BLBC	STATUS, 8\$	2661		
04			50		02	E1	000D5	8\$:	BISB2	#2, 35(D)	2662	
			23	A2	04	88	000D9	BBC	#2, STATUS, 9\$	2662		
04			50		03	E1	000DD	9\$:	BISB2	#4, 35(D)	2663	
			23	A2	08	88	000E1	BBC	#3, STATUS, 10\$	2663		
			52		44	A2	9E	000E5	10\$:	BISB2	#8, 35(D)	2621
			53		52	D1	000E9	11\$:	MOVAB	68(R2), D	2621	
					89	1B	000EC	CMPL	D, R3	2621		
					04	000EE		BLEQU	3\$	2666		
								RET		2666		

; Routine Size: 239 bytes, Routine Base: CODE + 0E6E

```
: 1572 2667 1 %SBTTL 'FIND NEXT - find next file'
: 1573 2668 1 GLOBAL ROUTINE FIND_NEXT=
: 1574 2669 1
: 1575 2670 1 |++
: 1576 2671 1 |
: 1577 2672 1 | FUNCTIONAL DESCRIPTION:
: 1578 2673 1 | This routine searches for the next file matching the specified
: 1579 2674 1 | selection filespec.
: 1580 2675 1 |
: 1581 2676 1 | INPUT PARAMETERS:
: 1582 2677 1 | NONE
: 1583 2678 1 |
: 1584 2679 1 | IMPLICIT INPUTS:
: 1585 2680 1 | Directory scan context.
: 1586 2681 1 |
: 1587 2682 1 | OUTPUT PARAMETERS:
: 1588 2683 1 | NONE
: 1589 2684 1 |
: 1590 2685 1 | IMPLICIT OUTPUTS:
: 1591 2686 1 | Directory scan context. If a file was found, the name block contains
: 1592 2687 1 | the resultant string and file ID.
: 1593 2688 1 |
: 1594 2689 1 | ROUTINE VALUE:
: 1595 2690 1 | True if a file was found, false otherwise.
: 1596 2691 1 |
: 1597 2692 1 | SIDE EFFECTS:
: 1598 2693 1 | NONE
: 1599 2694 1 |
: 1600 2695 1 | --
: 1601 2696 1 |
: 1602 2697 2 BEGIN
: 1603 2698 2 LOCAL
: 1604 2699 2 N: REF BBLOCK, ! Local copy of DIR_NAM
: 1605 2700 2 D: REF BBLOCK; ! Local copy of DIR_SP
: 1606 2701 2
: 1607 2702 2
: 1608 2703 2 N = .DIR_NAM;
: 1609 2704 2 D = .DIR_SP;
: 1610 2705 2 DIR_STATOS = DIR_VERLIMIT = 0;
: 1611 2706 2
: 1612 2707 2
: 1613 2708 2 ! If the directory stack is now empty, we have completed the MFD.
: 1614 2709 2 |
: 1615 2710 2 IF .DIR_LEVELS LEQ 0
: 1616 2711 2 THEN
: 1617 2712 2 BEGIN
: 1618 2713 2 N[NAM$B_RSL] = 0;
: 1619 2714 2 N[NAM$W_FID_NUM] = N[NAM$W_FID_SEQ] = N[NAM$W_FID_RVN] = 0;
: 1620 2715 2 N[NAM$W_DID_NUM] = N[NAM$W_DID_SEQ] = N[NAM$W_DID_RVN] = 0;
: 1621 2716 2 RETURN FALSE;
: 1622 2717 2 END;
: 1623 2718 2
: 1624 2719 2
: 1625 2720 2 ! Return the MFD if requested.
: 1626 2721 2 |
: 1627 2722 2 IF TESTBITSC(DIR_FLAGS[D_INITIAL])
: 1628 2723 2 THEN
```

```

: 1629      2724 2   IF .DIR_FLAGS[D_SCANNED_DIRS] AND .D[D_NON_TERM]
: 1630      2725 2   THEN
: 1631      2726 2     BEGIN
: 1632      2727 2     DIR_STATUS[D_STAT_VALID] = DIR_STATUS[D_STAT_SCANNED] = TRUE;
: 1633      2728 2     RETURN TRUE;
: 1634      2729 2     END;
: 1635      2730 2
: 1636      2731 2
: 1637      2732 2   ! Loop until we find something or traverse past a possible match.
: 1638      2733 2   !
: 1639      2734 2   WHILE TRUE DO
: 1640      2735 2     BEGIN
: 1641      2736 2     LOCAL
: 1642      2737 2       R:          REF BBLOCK,      ! Local copy of D_REC
: 1643      2738 2       V:          REF BBLOCK;      ! Local copy of D_VER
: 1644      2739 2
: 1645      2740 2
: 1646      2741 2     ! Push down to a lower directory if necessary. This logic depends upon the
: 1647      2742 2     ! resultant string in the name block being unmodified from the previous
: 1648      2743 2     ! call.
: 1649      2744 2     !
: 1650      2745 2     IF NOT (.DIR_FLAGS[D_SCAN_FAILED] OR .DIR_FLAGS[D_ROOT_MFD])
: 1651      2746 2     THEN IF TESTBITSC(D[D_DIR_SCAN])
: 1652      2747 2     THEN
: 1653      2748 2       BEGIN
: 1654      2749 2         !
: 1655      2750 2         ! Push down the directory stack.
: 1656      2751 2         !
: 1657      2752 2         DIR_LEVELS = .DIR_LEVELS + 1;
: 1658      2753 2         DIR_SP = D = .D + D_S_ENTRY;
: 1659      2754 2         BBLOCK[D[D_FID], FID$W_NUM] = .N[NAM$W_FID_NUM];
: 1660      2755 2         BBLOCK[D[D_FID], FID$W_SEQ] = .N[NAM$W_FID_SEQ];
: 1661      2756 2         BBLOCK[D[D_FID], FID$W_RVN] = .N[NAM$W_FID_RVN];
: 1662      2757 2         D[D_SAV_LEN] = .DIR_STRING[0];
: 1663      2758 2         D[D_VBN] = 1;
: 1664      2759 2         !
: 1665      2760 2         !
: 1666      2761 2         ! Generate the new directory string. If this is not the top level,
: 1667      2762 2         ! append a dot, and then append the directory name.
: 1668      2763 2         !
: 1669      2764 2         IF .DIR_STRING[0] NEQ 0 THEN DIR_STRING[0] = .DIR_STRING[0] + 1;
: 1670      2765 2         DIR_STRING[0] = .DIR_STRING[0] + .N[NAM$B_NAME];
: 1671      2766 2         END;
: 1672      2767 2         !
: 1673      2768 2         !
: 1674      2769 2         ! Get a new chunk of directory if necessary.
: 1675      2770 2         !
: 1676      2771 2         !
: 1677      2772 2         IF
: 1678      2773 2           .D[D_REC] GEQA .D[D_BUF_LIM] AND
: 1679      2774 2           (.D[D_VBN] EQL 1 OR .D[D_VBN] LEQU .D[D_DIR_LEN])
: 1680      2775 2         THEN
: 1681      2776 2           BEGIN
: 1682      2777 2             LITERAL
: 1683      2778 2               ATR_LENGTH= MAXU(
: 1684      2779 2                 $BYTEOFFSET(FH1$W_RECATTR)+32,
: 1685      2780 2                 $BYTEOFFSET(FH2$W_FILEPROT)+2);
:           LOCAL

```

```

: 1686      2781  4      FIB:          BBLOCK[FIB$C_LENGTH],  ! FIB
: 1687      2782  4      FIB_DESC:    VECTOR[2],          ! Descriptor for FIB
: 1688      2783  4      STATUS,      ! Status variable
: 1689      2784  4      IOSB:        VECTOR[4,WORD],      ! I/O status block
: 1690      2785  4      HEADER:      BBLOCK[ATR_LENGTH],  ! Beginning of file header
: 1691      2786  4      ATR_DESC:    BBLOCK[12];      ! ACP attributes list
: 1692      2787  4
: 1693      2788  4
: 1694      2789  4      ! Access the current directory file.
: 1695      2790  4
: 1696      2791  4      FIB_DESC[0] = FIB$C_LENGTH;
: 1697      2792  4      FIB_DESC[1] = FIB;
: 1698      2793  4      CH$FILL (0, FIB$C_LENGTH, FIB);
: 1699      2794  4      FIB[FIB$S_ACCTL] = FIB$M_NORECORD;
: 1700      2795  4      IF .QUAL[QUAL_IGNORE_INTE] THEN FIB[FIB$S_ACCTL] = FIB$M_NOLOCK OR FIB$M_NORECORD;
: 1701      2796  4      FIB[FIB$W_FID_NUM] = .BBLOCK[D[D_FID], FID$W_NUM];
: 1702      2797  4      FIB[FIB$W_FID_SEQ] = .BBLOCK[D[D_FID], FID$W_SEQ];
: 1703      2798  4      FIB[FIB$W_FID_RVN] = .BBLOCK[D[D_FID], FID$W_RVN];
: 1704      2799  4      ATR_DESC[0,0,16,0] = ATR_LENGTH;
: 1705      2800  4      ATR_DESC[2,0,16,0] = ATR$C_HEADER;
: 1706      2801  4      ATR_DESC[4,0,32,0] = HEADER;
: 1707      2802  4      ATR_DESC[8,0,32,0] = 0;
: 1708      2803  4      STATUS = C$QIOW(
: 1709      2804  4          FUNC=IOS_ACCESS OR IOSM_ACCESS,
: 1710      2805  4          CHAN=.DIR_CHAN,
: 1711      2806  4          IOSB=IOSB,
: 1712      2807  4          P1=FIB_DESC,
: 1713      2808  4          P5=ATR_DESC);
: 1714      2809  4      IF .STATUS THEN STATUS = .IOSB[0];
: 1715      2810  4
: 1716      2811  4
: 1717      2812  4      ! If a privilege violation occurred and the file specification is
: 1718      2813  4      ! nonwild, try an ACP call in case the problem is an execute-only
: 1719      2814  4      ! directory.
: 1720      2815  4
: 1721      2816  4      IF
: 1722      2817  4          .STATUS<0,16> EQL SSS_NOPRIV AND
: 1723      2818  4          NOT .COM_FLAGS[COM_STANDALONE] AND
: 1724      2819  4          NOT .D[D_WILD_TERM] AND
: 1725      2820  4          .D[BUF_ADDR] EQL 0
: 1726      2821  4      THEN
: 1727      2822  5          BEGIN
: 1728      2823  5              LOCAL
: 1729      2824  5                  FIB_DESC:    VECTOR[2],          ! Descriptor for FIB
: 1730      2825  5                  FIB:        BBLOCK[FIB$C_LENGTH],  ! FIB
: 1731      2826  5                  FNA_DESC:    VECTOR[2],          ! Descriptor for FNA
: 1732      2827  5                  FNA:        VECTOR[86, BYTE];      ! Buffer for 'n.t:v'
: 1733      2828  5
: 1734      2829  5
: 1735      2830  5                  D[BUF_LEN] = 512;
: 1736      2831  5                  D[BUF_ADDR] = GET_VM(512);
: 1737      2832  5
: 1738      2833  5
: 1739      2834  5                  ! Initialize the FIB.
: 1740      2835  5                  !
: 1741      2836  5                  FIB_DESC[0] = FIB$C_LENGTH;
: 1742      2837  5                  FIB_DESC[1] = FIB;

```

```

: 1743      2838  5      CHSFILL(0, FIBSC_LENGTH, FIB);
: 1744      2839  5      FIB[FIBSW_DID_NUM] = .BBLOCK[D[D_FID], FIDSW_NUM];
: 1745      2840  5      FIB[FIBSW_DID_SEQ] = .BBLOCK[D[D_FID], FIDSW_SEQ];
: 1746      2841  5      FIB[FIBSW_DID_RVN] = .BBLOCK[D[D_FID], FIDSW_RVN];
: 1747      2842  5
: 1748      2843  5
: 1749      2844  5      ! Initialize the filename.
: 1750      2845  5
: 1751      2846  5      FNA_DESC[0] = %ALLOCATION(FNA);
: 1752      2847  5      FNA_DESC[1] = FNA;
: 1753      2848  5      $FAD(
P      2849  5          $DESCRIPTOR('!AS;!UW'),
P      2850  5          FNA_DESC,
P      2851  5          FNA_DESC,
: 1757      2852  5          D[D_TERM_DESC], .D[D_TERM_VER]);
: 1758      2853  5
: 1759      2854  5
: 1760      2855  5      ! Execute the lookup.
: 1761      2856  5
: 1762      2857  5      STATUS = $QIOW(
P      2858  5          FUNC=IOS_ACCESS,
P      2859  5          CHAN=.DIR_CHAN,
P      2860  5          IOSB=IOSB,
P      2861  5          P1=FIB_DESC,
: 1766      2862  5          P2=FNA_DESC);
: 1767      2863  5      IF .STATUS THEN STATUS = .IOSB[0];
: 1768      2864  5
: 1769      2865  5
: 1770      2866  5
: 1771      2867  5      ! If a no such file error occurred, simulate an empty directory.
: 1772      2868  5      ! Otherwise, simulate a single block directory that contains the
: 1773      2869  5      ! desired entry.
: 1774      2870  5
: 1775      2871  5      IF .STATUS<0,16> EQL SSS_NOSUCHFILE
: 1776      2872  5      THEN
: 1777      2873  6          BEGIN
: 1778      2874  6              D[D_DIR_LEN] = 0;
: 1779      2875  6              STATUS = SSS_NORMAL;
: 1780      2876  6          END
: 1781      2877  5      ELSE
: 1782      2878  6          BEGIN
: 1783      2879  6              LOCAL
: 1784      2880  6                  L,                ! Length of name and type
: 1785      2881  6                  P:                ! Pointer to entry
: 1786      2882  6                  REF BBLOCK;
: 1787      2883  6
: 1788      2884  6          ! Initialize directory stack context.
: 1789      2885  6
: 1790      2886  6          DIR_STRUCLEV = 2;
: 1791      2887  6          D[D_VBN] = 2;
: 1792      2888  6          D[D_DIR_LEN] = 1;
: 1793      2889  6
: 1794      2890  6
: 1795      2891  6          ! Initialize simulated directory entry.
: 1796      2892  6
: 1797      2893  6          L = (.BBLOCK[D[D_TERM_DESC], DSCSW_LENGTH] + 1) AND NOT 1;
: 1798      2894  6          P = .D[D_BUF_ADDR];
: 1799      2895  6          P[DIRSW_SIZE] = .L + DIRSC_LENGTH - 2 + DIRSC_VERSION;

```

```

: 1800      2895  6      P[DIR$W_VERLIMIT] = .FIB[FIB$W_VERLIMIT];
: 1801      2896  6      P[DIR$B_FLAGS] = 0;
: 1802      2897  6      P[DIR$B_NAMECOUNT] = .BBLOCK[D[D_TERM_DESC], DSC$W_LENGTH];
: 1803      2898  6      P = CH$COPY(
: 1804      2899  6          .BBLOCK[D[D_TERM_DESC], DSC$W_LENGTH],
: 1805      2900  6          .BBLOCK[D[D_TERM_DESC], DSC$A_POINTER],
: 1806      2901  6          0,
: 1807      2902  6          .L, P[DIR$T_NAME]);
: 1808      2903  6      P[DIR$W_VERSION] = .D[D_TERM_VER];
: 1809      2904  6      P[DIR$W_FID_NUM] = .FIB[FIB$W_FID_NUM];
: 1810      2905  6      P[DIR$W_FID_SEQ] = .FIB[FIB$W_FID_SEQ];
: 1811      2906  6      P[DIR$W_FID_RVN] = .FIB[FIB$W_FID_RVN];
: 1812      2907  6
: 1813      2908  6
: 1814      2909  6      D[D_BUF_LIM] = .P + DIR$C_VERSION;
: 1815      2910  5      END;
: 1816      2911  4      END;
: 1817      2912  4
: 1818      2913  4
: 1819      2914  4      IF NOT .STATUS
: 1820      2915  4      THEN
: 1821      2916  5          BEGIN
: 1822      2917  5              ! Report failure to access the directory.
: 1823      2918  5              !
: 1824      2919  5              ! SIGNAL(
: 1825      2920  5              !     BACKUP$_OPENDIR,
: 1826      2921  5              !     2,
: 1827      2922  5              !     .DIR_DEV_DESC,
: 1828      2923  5              !     (IF .DIR_STRING[0] EQL 0 THEN MFD ELSE DIR_STRING),
: 1829      2924  5              !     .STATUS);
: 1830      2925  5
: 1831      2926  5
: 1832      2927  5
: 1833      2928  5              ! Readjust context so that processing of the directory is
: 1834      2929  5              ! aborted.
: 1835      2930  5              !
: 1836      2931  5              ! D[D_DIR_LEN] = 0;
: 1837      2932  5              ! END
: 1838      2933  4      ELSE
: 1839      2934  5          BEGIN
: 1840      2935  5
: 1841      2936  5              ! If there is not currently a buffer, this is the first chunk of
: 1842      2937  5              ! this directory. Determine whether the file is indeed a
: 1843      2938  5              ! directory, determine its size, and allocate the buffer.
: 1844      2939  5              !
: 1845      2940  5              ! IF .D[D_BUF_ADDR] EQL 0
: 1846      2941  5              ! THEN
: 1847      2942  6                  BEGIN
: 1848      2943  6
: 1849      2944  6              ! Ensure that the file is, in fact, a directory. At this point
: 1850      2945  6              ! we know only that the filename is ".DIR;1". Use the portion
: 1851      2946  6              ! of the file header that was obtained during the access.
: 1852      2947  6              ! Compute the file length if valid. If invalid, leave the file
: 1853      2948  6              ! length zero to avoid processing the file.
: 1854      2949  6              !
: 1855      2950  6              ! D[D_DIR_LEN] = 0;
: 1856      2951  6              ! DIR_STRUCLEV = .HEADER[FH2$B_STRUCLEV];

```



```

: 1857      2952  6      IF .DIR_STRUCLEV EQL 2
: 1858      2953  6      THEN
: 1859      2954  7      BEGIN
: 1860      2955  7      BIND
: 1861      2956  7      RECATTR=          HEADER[FH2$W_RECATTR]: BBLOCK;
: 1862      2957  7
: 1863      2958  7      IF .HEADER[FH2$V_DIRECTORY]
: 1864      2959  7      THEN
: 1865      2960  8      BEGIN
: 1866      2961  8      D[D_DIR_LEN] = ROT(.RECATTR[FAT$L_EFBLK], 16);
: 1867      2962  8      IF .RECATTR[FAT$W_FFBYTE] EQL 0
: 1868      2963  8          THEN D[D_DIR_LEN] = .D[D_DIR_LEN] - 1;
: 1869      2964  8      D[D_FPRO] = .HEADER[FH2$W_FI[EPROT]];
: 1870      2965  8      D[D_UIC] = .HEADER[FH2$L_FILEOWNER];
: 1871      2966  8      D[D_VERLIM] = .BBLOCK[HEADER[FH2$W_RECATTR], FAT$W_VERSIONS];
: 1872      2967  7      END;
: 1873      2968  7      END
: 1874      2969  6      ELSE
: 1875      2970  7      BEGIN
: 1876      2971  7      BIND
: 1877      2972  7      RECATTR=          HEADER[FH1$W_RECATTR]: BBLOCK;
: 1878      2973  7
: 1879      2974  7      IF
: 1880      2975  7          .RECATTR[FAT$B_RTYPE] EQL FAT$C_FIXED AND
: 1881      2976  7          .RECATTR[FAT$W_RSIZ] EQL NMB$C_DIRENTRY
: 1882      2977  7      THEN
: 1883      2978  8      BEGIN
: 1884      2979  8      D[D_DIR_LEN] = ROT(.RECATTR[FAT$L_EFBLK], 16);
: 1885      2980  8      IF .RECATTR[FAT$W_FFBYTE] EQL 0
: 1886      2981  8          THEN D[D_DIR_LEN] = .D[D_DIR_LEN] - 1;
: 1887      2982  8      D[D_FPRO] = .HEADER[FH1$W_FI[EPROT]];
: 1888      2983  8      D[D_UIC] = .HEADER[FH1$B_OICMEMBER];
: 1889      2984  8      (D[D_UIC])<16,16> = .HEADER[FH1$B_UICGROUP];
: 1890      2985  8      D[D_VERLIM] = .BBLOCK[HEADER[FH1$W_RECATTR], FAT$W_VERSIONS];
: 1891      2986  7      END;
: 1892      2987  6      END;
: 1893      2988  6
: 1894      2989  6
: 1895      2990  6      ! If the file looks like a directory and is not zero length,
: 1896      2991  6      ! allocate a buffer for it.
: 1897      2992  6      !
: 1898      2993  6      IF .D[D_DIR_LEN] NEQ 0
: 1899      2994  6      THEN
: 1900      2995  7      BEGIN
: 1901      2996  7
: 1902      2997  7          ! Compute buffer length. Try to read the entire directory
: 1903      2998  7          ! at one time, but no more than DIR_BUF_COUNT blocks.
: 1904      2999  7          ! However, for an ODS-1 directory on which a latest-version
: 1905      3000  7          ! scan is in progress, always read the entire directory.
: 1906      3001  7          !
: 1907      3002  7      D[D_BUF_LEN] = .D[D_DIR_LEN] * 512;
: 1908      3003  7      IF
: 1909      3004  7          .D[D_BUF_LEN] GTRU DIR_BUF_COUNT*512 AND
: 1910      3005  8          NOT (.DIR_STRUCLEV EQL 1 AND .DIR_SEL_LATEST LEQ 0)
: 1911      3006  7      THEN
: 1912      3007  7          D[D_BUF_LEN] = DIR_BUF_COUNT*512;
: 1913      3008  7

```

```

1914 3009 7
1915 3010 7
1916 3011 7
1917 3012 7
1918 3013 6
1919 3014 5
1920 3015 5
1921 3016 5
1922 3017 5
1923 3018 5
1924 3019 5
1925 3020 5
1926 3021 5
1927 3022 6
1928 3023 6
1929 3024 6
1930 3025 6
1931 3026 6
1932 3027 6
1933 3028 6
1934 3029 6
1935 3030 6
1936 3031 6
1937 3032 6
1938 3033 6
1939 3034 6
1940 3035 6
1941 3036 6
1942 3037 6
1943 3038 6
1944 3039 6
1945 3040 6
1946 3041 6
1947 3042 6
1948 3043 6
1949 3044 7
1950 3045 7
1951 3046 7
1952 3047 7
1953 3048 7
1954 3049 7
1955 3050 7
1956 3051 7
1957 3052 7
1958 3053 7
1959 3054 7
1960 3055 7
1961 3056 7
1962 3057 7
1963 P 3058 7
1964 P P 3059 7
1965 P P 3060 7
1966 P P 3061 7
1967 P 3062 7
1968 P 3063 7
1969 3064 7
1970 3065 7

! Allocate memory for buffer.
[DED_BUF_ADDR] = GET_VM(.D[DED_BUF_LEN]);
END;

END;

! If we are not yet beyond the end, size the portion of the
! directory to be read on this pass and read it.
IF .D[DED_VBN] LEQU .D[DED_DIR_LEN]
THEN
BEGIN
LOCAL
READ_ADDRESS, ! Address for current read QIO
PROC_LENGTH; ! Bytes to process this time

! Compute size of chunk to be processed on this iteration.
PROC_LENGTH = MINU(
.D[DED_BUF_LEN],
(.D[DED_DIR_LEN] - .D[DED_VBN] + 1) * 512);
DED_BUF_LIM = .D[DED_BUF_ADDR] + .PROC_LENGTH;
DED_REC = DED_VER = 0;

! Loop to read the chunk of directory that will be processed on
! this iteration. Because the chunk might exceed 65535 bytes,
! we must have a loop, though it is unlikely to be executed
! more than once.
READ_ADDRESS = .D[DED_BUF_ADDR];
WHILE .READ_ADDRESS -LSSA .D[DED_BUF_LIM] DO
BEGIN
LOCAL
READ_LENGTH; ! Size of current transfer in bytes

! Compute size of this transfer.
READ_LENGTH = MINU(
T27 * 512,
.D[DED_BUF_LIM] - .READ_ADDRESS);

! Read in the blocks.
STATUS = (SQIOW(
FUNC=IOS_READVBLK,
CHAN=.DIR_CHAN,
IOSB=IOSB,
P1=.READ_ADDRESS,
P2=.READ_LENGTH,
P3=.D[DED_VBN]));
READ_ADDRESS = .READ_ADDRESS + .READ_LENGTH;

```

```

: 1971 3066 7 D[D_VBN] = .D[D_VBN] + .READ_LENGTH/512;
: 1972 3067 7 IF .STATUS THEN STATUS = .IOSB[0];
: 1973 3068 7 IF NOT .STATUS
: 1974 3069 7 THEN
: 1975 3070 8 BEGIN
: 1976 3071 8 SIGNAL(
: 1977 3072 8 BACKUP$_READDIR,
: 1978 3073 8 2,
: 1979 3074 8 .DIR_DEV_DESC,
: 1980 3075 8 (IF .DIR_STRING[0] EQL 0 THEN MFD ELSE DIR_STRING),
: 1981 3076 8 .STATUS);
: 1982 3077 8
: 1983 3078 8
: 1984 3079 8 ! Readjust context so that processing of the directory
: 1985 3080 8 ! is aborted.
: 1986 3081 8
: 1987 3082 8
: 1988 3083 8 D[D_DIR_LEN] = 0;
: 1989 3084 8 D[D_BUF_LIM] = 0;
: 1990 3085 7 EXITLOOP;
: 1991 3086 6 END;
: 1992 3087 5 END;
: 1993 3088 5
: 1994 3089 5
: 1995 3090 5 ! Deaccess the directory file.
: 1996 3091 5
: 1997 P 3092 5 C$QIOW(
: 1998 P 3093 5 FUNC=IOS$ DEACCESS,
: 1999 3094 5 CHAN=.DIR_CHAN);
: 2000 3095 4 END;
: 2001 3096 4 END;
: 2002 3097 4
: 2003 3098 4
: 2004 3099 4 ! If no more directory is available, pop up to the higher directory.
: 2005 3100 4
: 2006 3101 4 IF .D[D_REC] GEQA .D[D_BUF_LIM]
: 2007 3102 4 THEN
: 2008 3103 4 BEGIN
: 2009 3104 4
: 2010 3105 4 ! Deallocate dynamic memory.
: 2011 3106 4
: 2012 3107 4
: 2013 3108 4 IF .D[D_BUF_ADDR] NEQ 0
: 2014 3109 4 THEN
: 2015 3110 4 FREE_VM(.D[D_BUF_LEN], .D[D_BUF_ADDR]);
: 2016 3111 4 IF .BBLOCK[D[D_TERM_DESC], DSC$_PTR] NEQ 0
: 2017 3112 4 THEN
: 2018 3113 4 FREE_VM(DIR$_NAME, .BBLOCK[D[D_TERM_DESC], DSC$_PTR]);
: 2019 3114 4 IF .BBLOCK[D[D_NAME_DESC], DSC$_PTR] NEQ 0
: 2020 3115 4 THEN
: 2021 3116 4 FREE_VM(DIR$_NAME, .BBLOCK[D[D_NAME_DESC], DSC$_PTR]);
: 2022 3117 4
: 2023 3118 4 ! Pop the directory stack.
: 2024 3119 4
: 2025 3120 4 DIR_STRING[0] = .D[D_SAV_LEN];
: 2026 3121 4 CH$FILL(0, D_S_ENTRY, .D);
: 2027 3122 4 DIR_SP = D = .D - D_S_ENTRY;

```



```

2085 3180 7 IF .DIR_SEL_LATEST LEQ 0
2086 3181 7 THEN
2087 3182 7   INCRA P
2088 3183 7   FROM .D[D_BUF_ADDR]
2089 3184 7   TO .D[D_BUF_ADDR] + .D[D_BUF_LEN] - NMBSC_DIRENTRY
2090 3185 7   BY NMBSC_DIRENTRY DO
2091 3186 8   BEGIN
2092 3187 8   MAP
2093 3188 8   P:          REF BBLOCK;
2094 3189 8
2095 3190 8   IF
2096 3191 8     .(P[NMBSW_NAME]) EQL .(R[NMBSW_NAME]) AND
2097 3192 8     .(P[NMBSW_NAME]+4) EQL .(R[NMBSW_NAME]+4) AND
2098 3193 8     .P[NMBSW_VERSION] GTRU .R[NMBSW_VERSION]
2099 3194 8   THEN
2100 3195 9   BEGIN
2101 3196 9     D[D_VER_COUNT] = -32768;
2102 3197 9   EXITLOOP;
2103 3198 8   END;
2104 3199 7   END;
2105 3200 7
2106 3201 7   ! Exit the loop with a valid entry, and D_VER_COUNT set.
2107 3202 7   !
2108 3203 7   !
2109 3204 7   !
2110 3205 6   !
2111 3206 5   !
2112 3207 5   !
2113 3208 4   !
2114 3209 5   !
2115 3210 5   !
2116 3211 6   !
2117 3212 6   !
2118 3213 6   !
2119 3214 6   !
2120 3215 6   !
2121 3216 6   !
2122 3217 6   !
2123 3218 6   !
2124 3219 6   !
2125 3220 6   !
2126 3221 6   !
2127 3222 6   !
2128 3223 6   !
2129 3224 6   !
2130 3225 7   !
2131 3226 7   !
2132 3227 7   !
2133 3228 7   !
2134 3229 7   !
2135 3230 6   !
2136 3231 6   !
2137 3232 6   !
2138 3233 6   !
2139 3234 6   !
2140 3235 6   !
2141 3236 6   !

```

```

: 2142      3237  6
: 2143      3238  6
: 2144      3239  6
: 2145      3240  7
: 2146      3241  7
: 2147      3242  7
: 2148      3243  7
: 2149      3244  7
: 2150      3245  7
: 2151      3246  7
: 2152      3247  6
: 2153      3248  7
: 2154      3249  7
: 2155      3250  7
: 2156      3251  7
: 2157      3252  7
: 2158      3253  7
: 2159      3254  7
: 2160      3255  8
: 2161      3256  8
: 2162      3257  8
: 2163      3258  8
: 2164      3259  8
: 2165      3260  8
: 2166      3261  8
: 2167      3262  8
: 2168      3263  8
: 2169      3264  8
: 2170      3265  9
: 2171      3266  9
: 2172      3267  9
: 2173      3268  9
: 2174      3269  9
: 2175      3270  9
: 2176      3271  8
: 2177      3272  7
: 2178      3273  8
: 2179      3274  8
: 2180      3275  8
: 2181      3276  8
: 2182      3277  8
: 2183      3278  8
: 2184      3279  8
: 2185      3280  8
: 2186      3281  8
: 2187      3282  8
: 2188      3283  8
: 2189      3284  8
: 2190      3285  8
: 2191      3286  8
: 2192      3287  8
: 2193      3288  7
: 2194      3289  7
: 2195      3290  7
: 2196      3291  7
: 2197      3292  7
: 2198      3293  7

IF .R[DIR$W_SIZE] EQL 65535
THEN
  BEGIN
    ! End of this block. Advance to next.
    !
    R = ((.R - .D[D_BUF_ADDR]) AND NOT 511) + .D[D_BUF_ADDR] + 512;
    V = 0;
  END
ELSE
  BEGIN
    ! Point to where next record should start. Make some
    ! validity tests on the entry we are looking at.
    !
    NEXT_RECORD = .R[DIR$W_SIZE] + .R + 2;
    IF
      BEGIN
        IF
          .NEXT_RECORD GEQA ((.R - .D[D_BUF_ADDR]) AND NOT 511) + .D[D_BUF_ADDR] + 512 OR
          ! Entry within block?
          .R[DIR$W_SIZE] OR
          ! Length even?
          .R[DIR$W_SIZE] LSSU DIR$C_LENGTH + DIR$C_VERSION
          ! Long enough?
        THEN
          TRUE
        ELSE
          BEGIN
            V = (.R + DIR$C_LENGTH + .R[DIR$B_NAMECOUNT] + 1) AND NOT 1;
            ! Proper type code?
            .R[DIR$V_TYPE] NEQ DIR$C_FID OR
            ! Version entry within block?
            .V GEQA ((.R - .D[D_BUF_ADDR]) AND NOT 511) + .D[D_BUF_ADDR] + 512 - DIR$C_VERSI
          END
        END
      END
    THEN
      BEGIN
        ! Directory format error. Report it and quit.
        !
        SIGNAL(
          BACKUP$_BADDR,
          2,
          .DIR_DEV_DESC,
          (IF .DIR_STRING[0] EQL 0 THEN MFD ELSE DIR_STRING));
        !
        ! Adjust context to avoid processing this directory.
        !
        R = .D[D_BUF_LIM];
        D[D_DIR_LEN] = 0;
      END;
    ! Found a valid entry. Exit the loop.
    !
    EXITLOOP;
  END

```

```
: 2199      3294  6      END;
: 2200      3295  5      END;
: 2201      3296  4      END;
: 2202      3297  4      D[D_REC] = .R;
: 2203      3298  4      D[D_VER] = .V;
: 2204      3299  3      END;
: 2205      3300  3
: 2206      3301  3
: 2207      3302  3
: 2208      3303  3      ! If an entry was found, finish processing it.
: 2209      3304  3      !
: 2210      3305  3      IF .R LSSA .D[D_BUF_LIM]
: 2211      3306  4      THEN
: 2212      3307  4          BEGIN
: 2213      3308  4              LOCAL
: 2214      3309  4                  RSA_DESC: VECTOR[2],          ! Descriptor for RSA area
: 2215      3310  4                  NAME: REF VECTOR[.BYTE],      ! Pointer to ASCII name.typ
: 2216      3311  4                  VERSION,                    ! Binary version number
: 2217      3312  4                  FILE_NAME: VECTOR[20,BYTE];    ! Area to build ODS-1 filename
: 2218      3313  4
: 2219      3314  4          ! Get a pointer to the name.typ string and the version number.
: 2220      3315  4          ! Initialize the file ID in the name block.
: 2221      3316  4          !
: 2222      3317  4          IF .DIR_STRUCLEV EQL 2
: 2223      3318  4          THEN
: 2224      3319  5              BEGIN
: 2225      3320  5                  D[D_VER_COUNT] = .D[D_VER_COUNT] - 1;
: 2226      3321  5                  IF
: 2227      3322  5                      .R[DIR$B_NAMECOUNT] NEQ .BBLOCK[D[D_NAME_DESC], DSC$W_LENGTH] OR
: 2228      3323  5                      CH$NEQ(
: 2229      3324  5                          .R[DIR$B_NAMECOUNT],
: 2230      3325  5                          R[DIR$T_NAME],
: 2231      3326  5                          .R[DIR$B_NAMECOUNT],
: 2232      3327  5                          .BBLOCK[D[D_NAME_DESC], DSC$A_POINTER])
: 2233      3328  5                  THEN
: 2234      3329  6                      BEGIN
: 2235      3330  6                          D[D_VER_COUNT] = 0;
: 2236      3331  6                          BBLOCK[D[D_NAME_DESC], DSC$W_LENGTH] = .R[DIR$B_NAMECOUNT];
: 2237      3332  6                          CH$MOVE(
: 2238      3333  6                              .R[DIR$B_NAMECOUNT],
: 2239      3334  6                              R[DIR$T_NAME],
: 2240      3335  6                              .BBLOCK[D[D_NAME_DESC], DSC$A_POINTER]);
: 2241      3336  5                          END;
: 2242      3337  5                          NAME = R[DIR$B_NAMECOUNT];
: 2243      3338  5                          VERSION = .V[DIR$W_VERSION];
: 2244      3339  5                          DIR_VERLIMIT = .R[DIR$W_VERLIMIT];
: 2245      3340  5                          N[NAM$W_FID_NUM] = .V[DIR$W_FID_NUM];
: 2246      3341  5                          N[NAM$W_FID_SEQ] = .V[DIR$W_FID_SEQ];
: 2247      3342  5                          N[NAM$W_FID_RVN] = .V[DIR$W_FID_RVN];
: 2248      3343  5                          IF .N[NAM$B_FID_RVN] EQL 0
: 2249      3344  5                              THEN N[NAM$B_FID_RVN] = .BBLOCK[D[D_FID], FID$B_RVN];
: 2250      3345  5                      END
: 2251      3346  4          ELSE
: 2252      3347  5              BEGIN LOCAL T;
: 2253      3348  5                  T = MAKE_STRING(.R, FILE_NAME[1]);
: 2254      3349  5                  FILE_NAME[0] = CH$FIND_CR(T, FILE_NAME[1], '%C';') - FILE_NAME[1];
: 2255      3350  5                  NAME = FILE_NAME;
```

```

: 2256      3351 5      VERSION = .R[NMBSW_VERSION];
: 2257      3352 5      N[NAMSW_FID_NUM] = .R[NMBSW_FID_NUM];
: 2258      3353 5      N[NAMSW_FID_SEQ] = .R[NMBSW_FID_SEQ];
: 2259      3354 5      N[NAMSW_FID_RVN] = 1;
: 2260      3355 4      END;
: 2261      3356 4
: 2262      3357 4
: 2263      3358 4      ! Initialize the resultant string in the name block.
: 2264      3359 4      !
: 2265      3360 4      RSA_DESC[0] = .N[NAMSB_RSS];
: 2266      3361 4      RSA_DESC[1] = .N[NAMSL_RSA];
: 2267      P 3362 4      $FAO(
: 2268      P 3363 4          $DESCRIPTOR('!ASC!AC!AC;!UW'),
: 2269      P 3364 4          RSA_DESC,
: 2270      P 3365 4          RSA_DESC,
: 2271      P 3366 4          .DIR_DEV_DESC,
: 2272      P 3367 4          (IF .DIR_STRING[0] EQL 0 THEN MFD ELSE DIR_STRING),
: 2273      P 3368 4          .NAME,
: 2274      3369 4          .VERSION);
: 2275      3370 4      N[NAMSB_RSL] = .RSA_DESC[0];
: 2276      3371 4      INIT_NAMEBLOCK(.N);
: 2277      3372 4
: 2278      3373 4
: 2279      3374 4      ! Initialize the directory ID in the name block.
: 2280      3375 4      !
: 2281      3376 4      N[NAMSW_DID_NUM] = .BBLOCK[D[D_FID], FIDSW_NUM];
: 2282      3377 4      N[NAMSW_DID_SEQ] = .BBLOCK[D[D_FID], FIDSW_SEQ];
: 2283      3378 4      N[NAMSW_DID_RVN] = .BBLOCK[D[D_FID], FIDSW_RVN];
: 2284      3379 4
: 2285      3380 4
: 2286      3381 4      ! Set the directory scan bits for the next iteration.
: 2287      3382 4      !
: 2288      3383 4      DIR_STATUS = 0;
: 2289      3384 4      IF
: 2290      3385 4          .DIR_LEVELS LEQ D_K_NLEVELS-1 AND
: 2291      3386 5          (IF .N[NAMSB_TYPE] EQL 4
: 2292      3387 5              THEN .N[NAMSL_TYPE] EQL '.DIR'
: 2293      3388 4              ELSE FALSE) AND
: 2294      3389 4          .VERSION EQL 1 AND
: 2295      3390 5          (.N[NAMSW_FID_NUM] NEQ FIDSC_MFD OR
: 2296      3391 5              .N[NAMSB_FID_NMX] NEQ 0)
: 2297      3392 4      THEN
: 2298      3393 5          BEGIN
: 2299      3394 5              LOCAL
: 2300      3395 5                  STATUS,          ! Status variable
: 2301      3396 5                  DIR_DESC: VECTOR[2]; ! Descriptor for new directory string
: 2302      3397 5
: 2303      3398 5
: 2304      3399 5          ! Note that this is a directory.
: 2305      3400 5          !
: 2306      3401 5          DIR_STATUS[D_STAT_VALID] = TRUE;
: 2307      3402 5
: 2308      3403 5
: 2309      3404 5          ! Generate the new directory string. If this is not the top level.
: 2310      3405 5          ! append a dot, and then append the directory name.
: 2311      3406 5          !
: 2312      3407 5          DIR_DESC[0] = .DIR_STRING[0];

```



```
2313 3408 5 DIR_DESC[1] = DIR_STRING[1];
2314 3409 5 IF .DIR_DESC[0] NEQ 0
2315 3410 5 THEN
2316 3411 6 BEGIN
2317 3412 6 DIR_DESC[0] = .DIR_DESC[0] + 1;
2318 3413 6 DIR_STRING[.DIR_DESC[0]] = %C'.';
2319 3414 5 END;
2320 3415 5 CH$MOVE(
2321 3416 5 .N[NAM$B_NAME],
2322 3417 5 .N[NAM$L_NAME],
2323 3418 5 DIR_STRING[.DIR_DESC[0]+1]);
2324 3419 5 DIR_DESC[0] = .DIR_DESC[0] + .N[NAM$B_NAME];
2325 3420 5
2326 3421 5
2327 3422 5 ! Temporarily push down the directory stack.
2328 3423 5 !
2329 3424 5 D = .D + D_S_ENTRY;
2330 3425 5
2331 3426 5
2332 3427 5 ! Allocate the dynamic areas for this level.
2333 3428 5 !
2334 3429 5 BBLOCK[D[D_TERM_DESC], DSC$A_POINTER] = GET_VM(DIR$S_NAME);
2335 3430 5 BBLOCK[D[D_NAME_DESC], DSC$A_POINTER] = GET_VM(DIR$S_NAME);
2336 3431 5
2337 3432 5
2338 3433 5 ! Match the directory specification against the pattern string to
2339 3434 5 ! determine if this directory should be scanned. If it should not,
2340 3435 5 ! pop the directory stack. Otherwise, finish initializing context.
2341 3436 5 !
2342 3437 5 IF .DIR_FLAGS[D_IMAGE_SCAN]
2343 3438 5 THEN
2344 3439 5 STATUS = %B'111'
2345 3440 5 ELSE
2346 3441 5 STATUS = MATCH_DIRECTORY(
2347 3442 5 DIR_DESC,
2348 3443 5 DIR_SEL_DIR,
2349 3444 5 D[D_TERM_DESC],
2350 3445 5 D[D_TERM_VER],
2351 3446 5 DIR_SEL_NTV);
2352 3447 5
2353 3448 5
2354 3449 5 IF .STATUS<1,1>
2355 3450 5 THEN
2356 3451 6 BEGIN
2357 3452 6 DIR_SP[D DIR_SCAN] = DIR_STATUS[D STAT_SCANNED] = TRUE;
2358 3453 6 IF .STATUS<0,1> THEN D[D_DIR_MATCHES] = DIR_STATUS[D_STAT_FILE_SEL] = TRUE;
2359 3454 6 IF .STATUS<2,1> THEN D[D_WILD_TERM] = TRUE;
2360 3455 6 IF .STATUS<3,1> THEN D[D_NON_TERM] = TRUE;
2361 3456 6 END
2362 3457 5 ELSE
2363 3458 6 BEGIN
2364 3459 6 FREE_VM(DIR$S_NAME, .BBLOCK[D[D_TERM_DESC], DSC$A_POINTER]);
2365 3460 6 FREE_VM(DIR$S_NAME, .BBLOCK[D[D_NAME_DESC], DSC$A_POINTER]);
2366 3461 6 CH$FILL(0, D_S_ENTRY, .D);
2367 3462 5 END;
2368 3463 5
2369 3464 5
```

```
2370 3465 5      ! Pop the directory stack.
2371 3466 5      !
2372 3467 5      D = .DIR_SP;
2373 3468 4      END;
2374 3469 4
2375 3470 4
2376 3471 4      ! If this is image mode, the file always matches.
2377 3472 4      !
2378 3473 4      IF
2379 3474 4          .DIR_FLAGS[D_IMAGE_SCAN] AND
2380 3475 5          (NOT .DIR_FLAGS[D_SCANNED_DIRS] OR
2381 3476 5              .N[NAM$W_FID_NUM] NEQ FID$C_MFD OR
2382 3477 5              .N[NAM$B_FID_NMX] NEQ 0)
2383 3478 4      THEN
2384 3479 5          BEGIN
2385 3480 5              IF .DIR_STATUS[D_STAT_VALID]
2386 3481 5                  THEN DIR_STATUS[D_STAT_DIR_SEL] = TRUE;
2387 3482 5          EXITLOOP;
2388 3483 4          END;
2389 3484 4
2390 3485 4
2391 3486 4      ! Test for scan termination.
2392 3487 4      !
2393 3488 4      IF
2394 3489 5          BEGIN
2395 3490 5              IF .DIR_STRUCLEV EQL 1
2396 3491 5                  THEN
2397 3492 5                      .V GTRU .DIR_SCANLIMIT[.DIR_LEVELS-1]
2398 3493 5                  ELSE
2399 3494 5                      IF .BBLOCK[D[D_TERM_DESC], DSC$W_LENGTH] EQL 0
2400 3495 5                          THEN
2401 3496 5                              FALSE
2402 3497 5                          ELSE
2403 3498 5                              TERMINATE_SCAN(
2404 3499 5                                  .NAME[0], NAME[1],
2405 3500 5                                  .VERSION,
2406 3501 5                                  .BBLOCK[D[D_TERM_DESC], DSC$W_LENGTH],
2407 3502 5                                  .BBLOCK[D[D_TERM_DESC], DSC$A_POINTER],
2408 3503 5                                  .D[D_TERM_VER])
2409 3504 5          END
2410 3505 4      THEN
2411 3506 5          BEGIN
2412 3507 5              IF .DIR_FLAGS[D_HARD_STOP]
2413 3508 5                  THEN
2414 3509 6                  BEGIN
2415 3510 6                      DIR_FLAGS[D_SCAN_FAILED] = TRUE;
2416 3511 6                      RETURN FALSE;
2417 3512 5                  END;
2418 3513 5
2419 3514 5
2420 3515 5      ! Adjust context to skip rest of this directory.
2421 3516 5      !
2422 3517 5      D[D_REC] = .D[D_BUF_LIM];
2423 3518 5      D[D_DIR_LEN] = 0;
2424 3519 4      END;
2425 3520 4
2426 3521 4
```

```

: 2427 3522 4      : Match the selection file specification with the resultant string.
: 2428 3523 4      :
: 2429 3524 4      IF
: 2430 3525 4      .D[D_REC] LSSA .D[D_BUF_LIM] AND
: 2431 3526 4      .D[D_DIR_MATCHES] AND
: 2432 3527 4      (.DIR_SEL_LATEST GTR 0 OR .DIR_SEL_LATEST EQL .D[D_VER_COUNT]) AND
: 2433 3528 5      (NOT .DIR_FLAGS[D_SCANNED_DIRS] OR
: 2434 3529 5      .N[NAM$W_FID_NUM] NEQ FID$C_MFD OR
: 2435 3530 5      .N[NAM$B_FID_NMX] NEQ 0)
: 2436 3531 4      THEN
: 2437 3532 5      BEGIN
: 2438 3533 5      LOCAL
: 2439 3534 5      NTV_DESC:      VECTOR[2];      ! Descriptor for n.t;v
: 2440 3535 5
: 2441 3536 5
: 2442 3537 5      ! Match the file specification.
: 2443 3538 5      :
: 2444 3539 5      NTV_DESC[0] = .N[NAM$L_RSA] + .N[NAM$B_RSL] - .N[NAM$L_NAME];
: 2445 3540 5      NTV_DESC[1] = .N[NAM$L_NAME];
: 2446 3541 5      IF MATCH_FILENAME(NTV_DESC, DIR_SEL_NTV)
: 2447 3542 5      THEN
: 2448 3543 6      BEGIN
: 2449 3544 6      IF .DIR_STATUS[D_STAT_VALID]
: 2450 3545 6      THEN DIR_STATUS[D_STAT_DIR_SEL] = TRUE;
: 2451 3546 6      EXITLOOP;
: 2452 3547 5      END;
: 2453 3548 4      END;
: 2454 3549 4
: 2455 3550 4
: 2456 3551 4      ! If scanned directories are requested, and the directory was not
: 2457 3552 4      ! selected above, return it anyway.
: 2458 3553 4      :
: 2459 3554 4      IF .DIR_FLAGS[D_SCANNED_DIRS] AND .D[D_DIR_SCAN] AND .D[D_NON_TERM]
: 2460 3555 4      THEN
: 2461 3556 4      EXITLOOP;
: 2462 3557 3      END;
: 2463 3558 2      END;
: 2464 3559 2
: 2465 3560 2
: 2466 3561 2 TRUE
: 2467 3562 1 END;

```

```

57 55 21 3B 53 41 21 00F5D P.AAJ: .ASCII \!AS;!UW\
00000007 00F64 P.AAI: .LONG 7
00000000' 00F68 .ADDRESS P.AAJ
57 55 21 3B 43 41 21 5D 43 41 21 5B 53 41 21 00F6C P.AAL: .ASCII \!AS[!AC]!AC;!UW\
00F7B .BLKB 1
0000000F 00F7C P.AAK: .LONG 15
00000000' 00F80 .ADDRESS P.AAL

.EXTRN STA_QIOW

OFFC 0000 .ENTRY FIND_NEXT, Save R2,R3,R4,R5,R6,R7,R8,R9,-
R10,R11
SE FEB4 CE 9E 0002 MOVAB -332(SP), SP
:
:
:
:
: 2668
:
:

```

		58	00000000'	EF	D0	00007	MOVL	DIR_NAM, N	2703										
		56	00000000'	EF	D0	0000E	MOVL	DIR_SP, D	2704										
			00000000'	EF	B4	00015	CLRW	DIR_VERLIMIT	2705										
			00000000'	EF	94	0001B	CLRB	DIR_STATUS											
			00000000'	EF	95	00021	TSTB	DIR_LEVELS	2710										
					03	12	00027	BNEQ	1\$										
					0490	31	00029	BRW	44\$										
17	00000000'	EF		04	E5	0002C	1\$:	BBCC	#4, DIR_FLAGS, 2\$	2722									
OF	00000000'	EF		03	E1	00034		BBC	#3, DIR_FLAGS, 2\$	2724									
OA	23	A6		03	E1	0003C		BBC	#3, 35(D), 2\$										
	00000000'	EF		05	88	00041		BISB2	#5, DIR_STATUS	2727									
					08B7	31	00048	BRW	96\$	2728									
4D	00000000'	EF		01	E0	0004B	2\$:	BBS	#1, DIR_FLAGS, 4\$	2745									
45	00000000'	EF		05	E0	00053		BBS	#5, DIR_FLAGS, 4\$										
40	23	A6		00	E5	0005B		BBCC	#0, 35(D), 4\$	2746									
			00000000'	EF	96	00060		INCB	DIR_LEVELS	2752									
				56	44	A6	9E	00066	MOVAB	68(R6), D	2753								
	00000000'	EF		56	D0	0006A		MOVL	D, DIR_SP										
				50	1C	A6	9E	00071	MOVAB	28(D), -R0	2754								
				60	24	A8	D0	00075	MOVL	36(N), (R0)									
	04	A0		28	A8	B0	00C79	MOVW	40(N), 4(R0)	2756									
	22	A6	00000000'	EF	90	0007E		MOVB	DIR_STRING, 34(D)	2757									
	14	A6		01	D0	00086		MOVL	#1, -20(D)	2758									
			00000000'	EF	95	0008A		TSTB	DIR_STRING	2764									
				06	13	00090		BEQL	3\$										
			00000000'	EF	96	00092		INCB	DIR_STRING										
	00000000'	EF	38	A8	80	00098	3\$:	ADDB2	59(N), DIR_STRING	2765									
				5A	10	A6	9E	000A0	4\$:	MOVAB	16(D), R10	2772							
				6A		66	D1	000A4		CMPL	(D), (R10)								
					03	1E	000A7		BGEQU	6\$									
					03B3	31	000A9	5\$:	BRW	40\$									
				01	14	A6	D1	000AC	6\$:	CMPL	20(D), #1	2773							
						07	13	000B0		BEQL	7\$								
	18	A6		14	A6	D1	000B2		CMPL	20(D), 24(D)									
					F0	1A	000B7		BGTRU	5\$									
	B8	AD	40	8F	9A	000B9	7\$:	MOVZBL	#64, FIB_DESC	2791									
	BC	AD	C0	AD	9E	000BE		MOVAB	FIB, FIB_DESC+4	2792									
0040	8F			00	00	2C	000C3		MOVCS	#0, (SP), #0, #64, FIB	2793								
										000CA									
										8F	D0	000CC					MOVL	#2097152, FIB	2794
										02	E1	000D4					BBC	#2, QUAL+10, 8\$	2795
08	00000000'	AD	00200000	8F	D0	000DC				8F	D0	000DC					MOVL	#3145728, FIB	
										57	1C	A6	9E	000E4	8\$:		MOVAB	28(D), R7	2796
												67	D0	000E8			MOVL	(R7), FIB+4	
												A7	B0	000EC			MOVW	4(R7), FIB+8	2798
												8F	D0	000F1			MOVL	#655426, ATR_DESC	2799
												CD	9E	000FA			MOVAB	HEADER, ATR_DESC+4	2801
												CD	D4	00101			CLRL	ATR_DESC+8	2802
												EF	D0	00105			MOVL	DIR_CHAN, R0	2808
												50	D1	0010C			CMPL	R0, #131071	
												23	1F	00113			BLSSU	9\$	
												7E	D4	00115			CLRL	-(SP)	
												FF60	CD	9F	00117		PUSHAB	ATR_DESC	
												7E	7C	0011B			CLRQ	-(SP)	
												7E	D4	0011D			CLRL	-(SP)	
												B8	AD	9F	0011F		PUSHAB	FIB_DESC	
												7E	7C	00122			CLRQ	-(SP)	

			B0	AD	9F	00124	PUSHAB	IOSB		
		7E	72	8F	9A	00127	MOVZBL	#114, -(SP)		
				50	DD	0012B	PUSHL	R0		
				7E	D4	0012D	CLRL	-(SP)		
	00000000G	00		0C	FB	0012F	CALLS	#12, STA_QIOW		
				21	11	00136	BRB	10\$		
				7E	D4	00138	CLRL	-(SP)		
			FF60	CD	9F	0013A	PUSHAB	ATR_DESC		
				7E	7C	0013E	CLRQ	-(SP)		
				7E	D4	00140	CLRL	-(SP)		
			B8	AD	9F	00142	PUSHAB	FIB_DESC		
				7E	7C	00145	CLRQ	-(SP)		
			B0	AD	9F	00147	PUSHAB	IOSB		
		7E	72	8F	9A	0014A	MOVZBL	#114, -(SP)		
				50	DD	0014E	PUSHL	R0		
				7E	D4	00150	CLRL	-(SP)		
	00000000G	9F		0C	FB	00152	CALLS	#12, @#SYSSQIOW		
		59		50	DD	00159	MOVL	R0, STATUS		2809
		04		59	E9	0015C	BLBC	STATUS, 11\$		
		59	B0	AD	3C	0015F	MOVZWL	IOSB, STATUS		2817
		24		59	B1	00163	CMPW	STATUS, #36		
				08	12	00166	BNEQ	12\$		2818
03	00000000'	EF		01	E1	00168	BBC	#1, COM_FLAGS, 13\$		
				00E3	31	00170	BRW	16\$		2819
F8		23	A6	02	E0	00173	BBS	#2, 35(D), 12\$		2820
				0C	A6	00178	TSTL	12(D)		
					F3	12	BNEQ	12\$		
		08	A6	0200	8F	3C	MOVZWL	#512, 8(D)		2830
		7E	0200	8F	3C	00183	MOVZWL	#512, -(SP)		2831
	00000000G	00		01	FB	00188	CALLS	#1, GET_VM		
		0C	A6	50	DD	0018F	MOVL	R0, 12(D)		
	00A4	CE	40	8F	9A	00193	MOVZBL	#64, FIB_DESC		2836
	FF5C	CD	64	AE	9E	00199	MOVAB	FIB, FIB_DESC+4		2837
0040	8F	00	6E	00	2C	0019F	MOVCS	#0, (SP), #0, #04, FIB		2838
				64	AE	001A6				
		6E	AE	67	DD	001A8	MOVL	(R7), FIB+10		2839
		72	AE	04	A7	BD	MOVW	4(R7), FIB+14		2841
		5C	AE	56	8F	9A	MOVZBL	#86, FNA_DESC		2846
		60	AE	04	AE	9E	MOVAB	FNA, FNA_DESC+4		2847
				38	A6	3C	MOVZWL	56(D), -(SP)		2852
				52	24	A6	MOVAB	36(D), R2		
					52	DD	PUSHL	R2		
				64	AE	9F	PUSHAB	FNA_DESC		
				68	AE	9F	PUSHAB	FNA_DESC		
				FE11	CF	9F	PUSHAB	P.AXI		
	00000000G	00		05	FB	001CF	CALLS	#5, SYSSFAO		
				7E	7C	001D6	CLRQ	-(SP)		2862
				7E	7C	001D8	CLRQ	-(SP)		
			6C	AE	9F	001DA	PUSHAB	FNA_DESC		
			FF58	CD	9F	001DD	PUSHAB	FIB_DESC		
				7E	7C	001E1	CLRQ	-(SP)		
			B0	AD	9F	001E3	PUSHAB	IOSB		
				32	DD	001E6	PUSHL	#50		
			00000000'	EF	DD	001E8	PUSHL	DIR_CHAN		
				7E	D4	001EE	CLRL	-(SP)		
	00000000G	00		0C	FB	001F0	CALLS	#12, SYSSQIOW		
		59		50	DD	001F7	MOVL	R0, STATUS		

		04		59	E9	001FA		BLBC	STATUS, 14\$		2863
		59	B0	AD	3C	001FD		MOVZWL	IOSB, STATUS		2870
	0910	8F		59	B1	00201	14\$:	CMPW	STATUS, #2320		2873
				08	12	00206		BNEQ	15\$		2874
				A6	D4	00208		CLRL	24(D)		2870
		59		01	D0	0020B		MOVL	#1, STATUS		2885
	00000000'	EF		46	11	0020E	15\$:	BRB	16\$		2886
	14	A6		02	90	00210		MOVW	#2, DIR_STRUCLEV		2887
	18	A6		02	D0	00217		MOVL	#2, 20(D)		2892
		50		01	D0	0021B		MOVL	#1, 24(D)		2893
				62	3C	0021F		MOVZWL	(R2), R0		2894
				50	D6	00222		INCL	R0		2895
		50		01	8A	00224		BICB2	#1, L		2896
	63	53	0C	A6	D0	00227		MOVL	12(D), P		2897
		50		0C	A1	0022B		ADDW3	#12, L, (P)		2902
		02	A3	0090	CE	0022F		MOVW	FIB+44, 2(P)		2903
				04	A3	00235		CLRB	4(P)		2904
	50	05	A3	62	90	00238		MOVW	(R2), 5(P)		2906
		04	B2	62	2C	0023C		MOVW	(R2), 5(P)		2907
				06	A3	00242		MOVW	(R2), 24(R2) #0, L, 6(P)		2902
		63	38	A6	B0	00244		MOVW	56(D), (P)		2903
		02	A3	68	AE	00248		MOVL	FIB+4, 2(P)		2904
		06	A3	6C	AE	0024D		MOVW	FIB+8, 6(P)		2906
			6A	08	A3	00252		MOVAB	8(R3), (R10)		2909
			35	59	E8	00256	16\$:	BLBS	STATUS, 19\$		2914
				59	DD	00259		PUSHL	STATUS		2915
		00000000'		EF	95	0025B		TSTB	DIR_STRING		2924
				07	12	00261		BNEQ	17\$		2925
		50	EE15	CF	9E	00263		MOVAB	MFD, R0		2926
				07	11	00268		BRB	18\$		2927
		50	00000000'	EF	9E	0026A	17\$:	MOVAB	DIR_STRING, R0		2928
				50	DD	00271	18\$:	PUSHL	R0		2929
			00000000'	EF	DD	00273		PUSHL	DIR_DEV_DESC		2930
				02	DD	00279		PUSHL	#2		2931
			00000000G	8F	DD	0027B		PUSHL	#BACKUP\$ OPENDIR		2932
		00000000G	00	05	FB	00281		CALLS	#5, LIB\$SIGNAL		2933
				18	A6	00288		CLRL	24(D)		2934
				01D1	31	0028B		BRW	40\$		2935
				0C	A6	0028E	19\$:	TSTL	12(D)		2936
				03	13	00291		BEQL	20\$		2937
				00A3	31	00293		BRW	27\$		2938
		50	18	A6	9E	00296	20\$:	MOVAB	24(D), R0		2939
				60	D4	0029A		CLRL	(R0)		2940
		00000000'	EF	FF73	CD	0029C		MOVW	HEADER+7, DIR_STRUCLEV		2941
		02	00000000'	EF	91	002A5		CMPB	DIR_STRUCLEV, #2		2942
				22	12	002AC		BNEQ	22\$		2943
	4E	A1	AD	05	E1	002AE		BBC	#5, HEADER+53, 24\$		2944
	60	88	AD	10	9C	002B3		ROTL	#16, RECATTR+8, (R0)		2945
				8C	AD	002B8		TSTW	RECATTR+12		2946
				02	12	002BB		BNEQ	21\$		2947
				60	D7	002BD		DECL	(R0)		2948
		3A	A6	AC	AD	002BF	21\$:	MOVW	HEADER+64, 58(D)		2949
		3C	A6	A8	AD	002C4		MOVL	HEADER+60, 60(D)		2950
		40	A6	9E	AD	002C9		MOVW	HEADER+50, 64(D)		2951
				31	11	002CE		BRB	24\$		2952
		01	FF7A	CD	91	002D0	22\$:	CMPB	RECATTR, #1		2953
				2A	12	002D5		BNEQ	24\$		2954

		10	FF7C	CD	B1	002D7		CMPW	RECATTR+2, #16	2976
				23	12	002DC		BNEQ	24\$	2977
60	82	AD		10	9C	002DE		ROTL	#16, RECATTR+8, (R0)	2979
			86	AD	B5	002E3		TSTW	RECATTR+12	2980
				02	12	002E6		BNEQ	23\$	2981
				60	D7	002E8		DECL	(R0)	2982
	3A	A6	FF76	CD	B0	002EA	23\$:	MOVW	HEADER+10, 58(D)	2983
	3C	A6	FF74	CD	9A	002F0		MOVZBL	HEADER+8, 60(D)	2984
	3E	A6	FF75	CD	9B	002F6		MOVZBW	HEADER+9, 62(D)	2985
	40	A6	98	AD	B0	002FC		MOVW	HEADER+44, 64(D)	2993
				60	D5	00301	24\$:	TSTL	(R0)	3002
				34	13	00303		BEQL	27\$	3003
08	A6	00002000		09	78	00305		ASHL	#9, (R0), 8(D)	3004
				8F	08	A6	D1	0030A	CMP	8(D), #8192
				01	00000000'	17	1B	00312	BLEQU	26\$
				08	00000000'	08	12	0031B	CMPB	DIR_STRUCLEV, #1
						08	D5	0031D	BNEQ	25\$
						06	15	00323	TSTL	DIR_SEL_LATEST
	08	A6	2000	8F	3C	00325	25\$:	MOVZWL	#8192, 8(D)	3007
			08	A6	DD	0032B	26\$:	PUSHL	8(D)	3012
	00000000G	00		01	FB	0032E		CALLS	#1, GET_VM	3020
	0C	A6		50	D0	00335		MOVL	R0, 12(D)	3032
	18	A6	14	A6	D1	00339	27\$:	CMP	20(D), 24(D)	3033
				2B	1A	0033E		BGTRU	30\$	3034
50	18	A6	14	A6	C3	00340		SUBL3	20(D), 24(D), R0	3035
50		50		09	78	00346		ASHL	#9, R0, R0	3036
		51	0200	C0	9E	0034A		MOVAB	512(R0), R1	3037
		50	08	A6	D0	0034F		MOVL	8(D), R0	3038
		51		50	D1	00353		CMP	R0, R1	3039
				03	1B	00356		BLEQU	28\$	3040
		50		51	D0	00358		MOVL	R1, R0	3041
		6A	0C	B640	9E	0035B	28\$:	MOVAB	@12(D)[PROC_LENGTH], (R10)	3042
				66	7C	00360		CLRQ	(D)	3043
		53	0C	A6	D0	00362		MOVL	12(D), READ_ADDRESS	3044
		6A		53	D1	00366	29\$:	CMP	READ_ADDRESS, (R10)	3045
				03	1F	00369		BLSSU	31\$	3046
				00B3	31	0036B	30\$:	BRW	38\$	3047
				53	C3	0036E	31\$:	SUBL3	READ_ADDRESS, (R10), R0	3048
50	0000FE00	8F		50	D1	00372		CMP	R0, #65024	3049
				05	1B	00379		BLEQU	32\$	3050
		50	FE00	8F	3C	0037B		MOVZWL	#65024, R0	3051
		52		50	D0	00380	32\$:	MOVL	R0, READ_LENGTH	3052
		50	00000000'	EF	D0	00383		MOVL	DIR_CHAN, R0	3053
	0001FFFF	8F		50	D1	0038A		CMP	R0, #131071	3054
				1F	1F	00391		BLSSU	33\$	3055
				7E	7C	00393		CLRQ	-(SP)	3056
				7E	D4	00395		CLRL	-(SP)	3057
			14	A6	DD	00397		PUSHL	20(D)	3058
				52	DD	0039A		PUSHL	READ_LENGTH	3059
				53	DD	0039C		PUSHL	READ_ADDRESS	3060
				7E	7C	0039E		CLRQ	-(SP)	3061
			B0	AD	9F	003A0		PUSHAB	IOSB	3062
				31	DD	003A3		PUSHL	#49	3063
				50	DD	003A5		PUSHL	R0	3064
				7E	D4	003A7		CLRL	-(SP)	3065
	00000000G	00		0C	FB	003A9		CALLS	#12, STA_QIOW	3066

			1D 11 003B0	BRB	34\$		
			7E 7C 003B2	CLRQ	-(SP)		
			7E D4 003B4	CLRL	-(SP)		
		14	A6 DD 003B6	PUSHL	20(D)		
			52 DD 003B9	PUSHL	READ_LENGTH		
			53 DD 003BB	PUSHL	READ_ADDRESS		
			7E 7C 003BD	CLRQ	-(SP)		
		B0	AD 9F 003BF	PUSHAB	IOSB		
			31 DD 003C2	PUSHL	#49		
			50 DD 003C4	PUSHL	R0		
			7E D4 003C6	CLRL	-(SP)		
00000000G	9F		0C FB 003C8	CALLS	#12, @#SYSS\$QIOW		
	59		50 D0 003CF	MOVL	R0, STATUS		
	53		52 C0 003D2	ADDL2	READ_LENGTH, READ_ADDRESS		3065
	52	00000200	8F C6 003D5	DIVL2	#512, R2		3066
	A6		52 C0 003DC	ADDL2	R2, 20(D)		
	0A		59 E9 003E0	BLBC	STATUS, 35\$		3067
	59	B0	AD 3C 003E3	MOVZWL	IOSB, STATUS		
	03		59 E9 003E7	BLBC	STATUS, 35\$		3068
			FF 79 31 003EA	BRW	29\$		
			59 DD 003ED	PUSHL	STATUS		3076
		00000000'	EF 95 003EF	TSTB	DIR_STRING		3075
			07 12 003F5	BNEQ	36\$		
	50	EC81	CF 9E 003F7	MOVAB	MFD, R0		
			07 11 003FC	BRB	37\$		
	50	00000000'	EF 9E 003FE	MOVAB	DIR_STRING, R0		
			50 DD 00405	PUSHL	R0		
		00000000'	EF DD 00407	PUSHL	DIR_DEV_DESC		3074
			02 DD 0040D	PUSHL	#2		3071
		00000000G	8F DD 0040F	PUSHL	#BACKUP\$ READDIR		
00000000G	00		05 FB 00415	CALLS	#5, LIB\$SIGNAL		
		18	A6 D4 0041C	CLRL	24(D)		3082
			6A D4 0041F	CLRL	(R10)		3083
	50	00000000'	EF D0 00421	MOVL	DIR_CHAN, R0		3094
0001FFFF	8F		50 D1 00428	CMPL	R0, #131071		
			18 1F 0042F	BLSSU	39\$		
			7E 7C 00431	CLRQ	-(SP)		
			7E 7C 00433	CLRQ	-(SP)		
			7E 7C 00435	CLRQ	-(SP)		
			7E 7C 00437	CLRQ	-(SP)		
	7E		34 7D 00439	MOVQ	#52, -(SP)		
			50 DD 0043C	PUSHL	R0		
			7E D4 0043E	CLRL	-(SP)		
00000000G	00		0C FB 00440	CALLS	#12, STA_QIOW		
			16 11 00447	BRB	40\$		
			7E 7C 00449	CLRQ	-(SP)		
			7E 7C 0044B	CLRQ	-(SP)		
			7E 7C 0044D	CLRQ	-(SP)		
			7E 7C 0044F	CLRQ	-(SP)		
	7E		34 7D 00451	MOVQ	#52, -(SP)		
			50 DD 00454	PUSHL	R0		
			7E D4 00456	CLRL	-(SP)		
00000000G	9F		0C FB 00458	CALLS	#12, @#SYSS\$QIOW		
	6A		66 D1 0045F	CMPL	(D), (R10)		3101
			6A 1F 00462	BLSSU	45\$		
		0C	A6 D5 00464	TSTL	12(D)		3107
			0B 13 00467	BEQL	41\$		

		7E	08	A6	7D	00469	MOVQ	8(D), -(SP)	3109	
		00		02	FB	0046D	CALLS	#2, FREE_VM		
			28	A6	D5	00474	41\$: TSTL	40(D)	3110	
				0E	13	00477	BEQL	42\$		
			28	A6	DD	00479	PUSHL	40(D)	3112	
		7E	50	8F	9A	0047C	MOVZBL	#80, -(SP)		
		00		02	FB	00480	CALLS	#2, FREE_VM		
			30	A6	D5	00487	42\$: TSTL	48(D)	3113	
				0E	13	0048A	BEQL	43\$		
			30	A6	DD	0048C	PUSHL	48(D)	3115	
		7E	50	8F	9A	0048F	MOVZBL	#80, -(SP)		
		00		02	FB	00493	CALLS	#2, FREE_VM		
0044	8F	00	22	A6	90	0049A	43\$: MOVB	34(D), DIR_STRING	3120	
				00	2C	004A2	MOVCS	#0, (SP), #0, #68, (D)	3121	
				66		004A9				
		56		3C	C2	004AA	SUBL2	#60, D	3122	
		EF		76	7E	004AD	MOVAQ	-(D), DIR_SP		
		00000000'		EF	97	004B4	DECB	DIR_LEVELS	3123	
				12	12	004BA	BNEQ	45\$	3128	
			03	A8	94	004BC	44\$: CLRB	3(N)	3131	
			26	A8	D4	004BF	CLRL	38(N)	3132	
			24	A8	B4	004C2	CLRW	36(N)		
			2C	A8	D4	004C5	CLRL	44(N)	3133	
			2A	A8	B4	004C8	CLRW	42(N)		
				03B2	31	004CB	BRW	87\$	3134	
		57		66	D0	004CE	45\$: MOVL	(D), R	3141	
		59	04	A6	D0	004D1	MOVL	4(D), V	3142	
		EF		01	E5	004D5	BBCC	#1, DIR_FLAGS, 46\$	3143	
03	00000000'			012B	31	004DD	BRW	63\$		
		01	00000000'	EF	91	004E0	46\$: CMPB	DIR_STRUCLEV, #1	3146	
				58	12	004E7	BNEQ	54\$		
				57	D5	004E9	47\$: TSTL	R	3154	
				06	12	004EB	BNEQ	48\$		
		57	0C	A6	D0	004ED	MOVL	12(D), R	3156	
				05	11	004F1	BRB	49\$		
		57		10	C0	004F3	48\$: ADDL2	#16, R	3159	
				59	D6	004F6	INCL	V	3160	
		10	A6	57	D1	004F8	49\$: CMPL	R, 16(D)	3166	
				6F	1E	004FC	BGEQU	57\$		
				67	B5	004FE	TSTW	(R)	3171	
				E7	13	00500	BEQL	47\$		
			42	A6	B4	00502	CLRW	66(D)	3179	
		00000000'		EF	D5	00505	TSTL	DIR_SEL_LATEST	3180	
				34	14	0050B	BGTR	53\$		
51	0C	A6	08	A6	C1	0050D	ADDL3	8(D), i2(D), R1	3184	
		51		10	C2	00513	SUBL2	#16, R1		
		50	0C	A6	D0	00516	MOVL	12(D), P	3191	
				20	11	0051A	BRB	52\$		
		06	A7	06	A0	D1	50\$: CMPL	6(P), 6(R)		
				16	12	00521	BNEQ	51\$		
		0A	A7	0A	A0	D1	CMPL	10(P), 10(R)	3192	
				0F	12	00528	BNEQ	51\$		
		0E	A7	0E	A0	B1	0052A	CMPW	14(P), 14(R)	3193
				08	1B	0052F	BLEQU	51\$		
		42	A6	8000	8F	B0	00531	MOVW	#-32768, 66(D)	3196
				08	11	00537	BRB	53\$	3195	
		50		10	C0	00539	51\$: ADDL2	#16, P	3182	

	51		50	D1	0053C	52\$:	CMPL	P, R1	
			DB	1B	0053F		BLEQU	50\$	
			00C0	31	00541	53\$:	BRW	62\$	
			57	D5	00544	54\$:	TSTL	R	3219
			06	12	00546		BNEQ	55\$	
	57	OC	A6	D0	00548		MOVL	12(D), R	3221
			1B	11	0054C		BRB	56\$	
			59	D5	0054E	55\$:	TSTL	V	3223
			17	13	00550		BEQL	56\$	
	50		67	3C	00552		MOVZWL	(R), R0	3226
	50	02	A740	9E	00555		MOVAB	2(R)[R0], NEXT_RECORD	
	59		08	C0	0055A		ADDL2	#8, V	3227
	51	F8	A0	9E	0055D		MOVAB	-8(R0), R1	3228
	51		59	D1	00561		CMPL	V, R1	
			DB	1B	00564		BLEQU	53\$	
	57		50	D0	00566		MOVL	NEXT_RECORD, R	3229
10	A6		57	D1	00569	56\$:	CMPL	R, 16(D)	3235
			D2	1E	0056D	57\$:	BGEQU	53\$	
	51		67	3C	0056F		MOVZWL	(R), R1	3238
FFFF	8F		51	B1	00572		CMPW	R1, #65535	
			17	12	00577		BNEQ	58\$	
52	57	OC	A6	C3	00579		SUBL3	12(D), R, R2	3244
	52	01FF	8F	AA	0057E		BICW2	#511, R2	
	52	OC	A6	C0	00583		ADDL2	12(D), R2	
	57	0200	C2	9E	00587		MOVAB	512(R2), R	
			59	D4	0058C		CLRL	V	3245
			B4	11	0058E		BRB	54\$	3238
	50	02	A741	9E	00590	58\$:	MOVAB	2(R)[R1], NEXT_RECORD	3253
52	57	OC	A6	C3	00595		SUBL3	12(D), R, R2	3257
	52	01FF	8F	AA	0059A		BICW2	#511, R2	
	52	OC	A6	C0	0059F		ADDL2	12(D), R2	
	52	0200	C2	9E	005A3		MOVAB	512(R2), R2	
	52		50	D1	005A8		CMPL	NEXT_RECORD, R2	
			23	1E	005AB		BGEQU	59\$	
	20		51	E8	005AD		BLBS	R1, 59\$	3259
	0E		51	B1	005B0		CMPW	R1, #14	3260
			1B	1F	005B3		BLSSU	59\$	
	50	05	A7	9A	005B5		MOVZBL	5(R), R0	3266
	50	07	A047	9E	005B9		MOVAB	7(R0)[R], R0	
59	50		01	CB	005BE		BICL3	#1, R0, V	
	07	04	A7	93	005C2		BITB	4(R), #7	3267
			08	12	005C6		BNEQ	59\$	
	52		08	C2	005C8		SUBL2	#8, R2	3268
	52		59	D1	005CB		CMPL	V, R2	
			34	1F	005CE		BLSSU	62\$	
		00000000'	EF	95	005D0	59\$:	TSTB	DIR_STRING	3261
			07	12	005D6		BNEQ	60\$	
	50	EAA0	CF	9E	005D8		MOVAB	MFD, R0	
			07	11	005DD		BRB	61\$	
	50	00000000'	EF	9E	005DF	60\$:	MOVAB	DIR_STRING, R0	
			50	DD	005E6	61\$:	PUSHL	R0	
		00000000'	EF	DD	005E8		PUSHL	DIR_DEV_DESC	3280
			02	DD	005EE		PUSHL	#2	3277
		00000000G	8F	DD	005F0		PUSHL	#BACKUP\$ BADDIR	
00000000G	00		04	FB	005F6		CALLS	#4, LIB\$SIGNAL	
	57	10	A6	D0	005FD		MOVL	16(D), R	3286
		18	A6	D4	00601		CLRL	24(D)	3287

			66		57	D0	00604	62\$:	MOVL	R, (D)	3297	
	04		A6		59	D0	00607		MOVL	V, 4(D)	3298	
	10		A6		57	D1	0060B	63\$:	CMPL	R, 16(D)	3304	
					03	1F	0060F		BLSSU	64\$		
					31	00611			BRW	2\$		
			5B	24	A8	9E	00614	64\$:	MOVAB	36(N), R11	3340	
			02	00000000'	EF	91	00618		CMPB	DIR_STRUCLEV, #2	3317	
					48	12	0061F		BNEQ	67\$		
					42	A6	B7	00621	DECW	66(D)	3320	
			55	05	A7	9A	00624		MOVZBL	5(R), R5	3322	
			54	2C	A6	9E	00628		MOVAB	44(D), R4		
			55		64	B1	0062C		CMPW	(R4), R5		
					08	12	0062F		BNEQ	65\$		
04	B4		06		55	29	00631		CMPC3	R5, 6(R), @4(R4)	3325	
					0C	13	00637		BEQL	66\$		
					42	A6	B4	00639	65\$:	CLRW	66(D)	3330
			64		55	B0	0063C		MOVW	R5, (R4)	3331	
04	B4		0C		55	28	0063F		MOV3	R5, 6(R), @4(R4)	3335	
			5A	05	A7	9E	00645	66\$:	MOVAB	5(R), NAME	3337	
			6E		69	32	00649		CVTTL	(V), VERSION	3338	
			00000000'		EF	02	A7	B0	0064C	MOVW	2(R), DIR_VERLIMIT	3339
					02	A9	B0	00654	MOVW	2(V), (R11)	3340	
			26		04	A9	D0	00658	MOVL	4(V), 38(N)	3341	
					28	A8	95	0065D	TSTB	40(N)	3343	
					39	12	00660		BNEQ	69\$		
			28		A6	9C	00662		MOVB	32(D), 40(N)	3344	
					32	11	00667		BRB	69\$	3317	
					E5	AD	9F	00669	67\$:	PUSHAB	FILE_NAME+1	3348
					57	DD	0066C		PUSHL	R		
E5	AD		00000000G	00	02	FB	0066E		CALLS	#2, MAKE_STRING		
				50	3B	3A	00675		LOCC	#59, T, FILE_NAME+1	3349	
					02	12	0067A		BNEQ	68\$		
					51	D4	0067C		CLRL	R1		
E4	AD			50	E5	AD	9E	0067E	68\$:	MOVAB	FILE_NAME+1, R0	
				51	50	83	00682		SUBB3	R0, R1, FILE_NAME		
				5A	E4	AD	9E	00687	MOVAB	FILE_NAME, NAME	3350	
				6E	0E	A7	32	0068B	CVTTL	14(R), VERSION	3351	
				6B	67	B0	0068F		MOVW	(R), (R11)	3352	
			26		A7	B0	00692		MOVW	2(R), 38(N)	3353	
			28		01	B0	00697		MOVW	#1, 40(N)	3354	
			F8		A8	9A	0069B	69\$:	MOVZBL	2(N), RSA_DESC	3360	
			FC		AD	D0	006A0		MOVL	4(N), RSA_DESC+4	3361	
					6E	DD	006A5		PUSHL	VERSION	3369	
					5A	DD	006A7		PUSHL	NAME		
					EF	95	006A9		TSTB	DIR_STRING		
					07	12	006AF		BNEQ	70\$		
			50	E9C7	CF	9E	006B1		MOVAB	MFD, R0		
					07	11	006B6		BRB	71\$		
			50	00000000'	EF	9E	006B8	70\$:	MOVAB	DIR_STRING, R0		
					50	DD	006BF	71\$:	PUSHL	R0		
					EF	DD	006C1		PUSHL	DIR_DEV_DESC		
					F8	AD	9F	006C7	PUSHAB	RSA_DESC		
					F8	AD	9F	006CA	PUSHAB	RSA_DESC		
					F927	CF	9F	006CD	PUSHAB	P.AAK		
			00000000G	00	07	FB	006D1		CALLS	#7, SYSSFAO		
			03		AD	90	006D8		MOVB	RSA_DESC, 3(N)	3370	
					58	DD	006DD		PUSHL	N	3371	

	00000000G	00		01	FB	006DF	CALLS	#1, INIT_NAMEBLOCK	
		50	1C	A6	9E	006E6	MOVAB	28(D), R0	3376
	2A	A8		60	D0	006EA	MOVL	(R0), 42(N)	
	2E	A8	04	A0	B0	006EE	MOVW	4(R0), 46(N)	3378
			00000000'	EF	94	006F3	CLRB	DIR_STATUS	3383
		08	00000000'	EF	91	006F9	CMPB	DIR_LEVELS, #8	3385
				1F	1A	00700	BGTRU	72\$	
		04	3C	A8	91	00702	CMPB	60(N), #4	3386
				19	12	00706	BNEQ	72\$	
	5249442E	8F	50	B8	D1	00708	CMPL	@80(N), #1380533294	3387
				0F	12	00710	BNEQ	72\$	
		01		6E	D1	00712	CMPL	VERSION, #1	3389
				0A	12	00715	BNEQ	72\$	
		04		6B	B1	00717	CMPW	(R11), #4	3390
				08	12	0071A	BNEQ	73\$	
			29	A8	95	0071C	TSTB	41(N)	3391
				03	12	0071F	BNEQ	73\$	
				00F0	31	00721	BRW	81\$	
	00000000'	EF		01	88	00724	BISB2	#1, DIR_STATUS	3401
				EF	9A	0072B	MOVZBL	DIR_STRING, DIR_DESC	3407
	DC	AD	00000000'	EF	9E	00733	MOVAB	DIR_STRING+1, DIR_DESC+4	3408
	E0	AD	00000000'	AD	D5	0073B	TSTL	DIR_DESC	3409
				0F	13	0073E	BEQL	74\$	
			DC	AD	D6	00740	INCL	DIR_DESC	3412
		50	00000000'	EF	9E	00743	MOVAB	DIR_STRING, R0	3413
	DC	BD40		2E	90	0074A	MOVB	#46, @DIR_DESC[R0]	
		51	3B	A8	9A	0074F	MOVZBL	59(N), R1	3416
		50	00000000'	EF	9E	00753	MOVAB	DIR_STRING+1, R0	3418
	DC	BD40		51	28	0075A	MOVW	R1, @76(N), @DIR_DESC[R0]	
		4C	B8	51	28	0075A	MOVZBL	59(N), R0	3419
				50	C0	00765	ADDL2	R0, DIR_DESC	
	DC	AD	3B	A8	9A	00761	MOVAB	68(R6), D	3424
		56	44	A6	9E	00769	MOVZBL	#80, -(SP)	3429
		7E	50	8F	9A	0076D	CALLS	#1, GET_VM	
	00000000G	00		01	FB	00771	MOVL	R0, 40(D)	
		28		50	D0	00778	MOVZBL	#80, -(SP)	3430
		7E	50	8F	9A	0077C	CALLS	#1, GET_VM	
	00000000G	00		01	FB	00780	MOVL	R0, 48(D)	
		30		50	D0	00787	BLBC	DIR_FLAGS, 75\$	3437
			00000000'	EF	E9	0078B	MOVL	#7, STATUS	3439
				07	D0	00792	BRB	76\$	
				1C	11	00795	PUSHAB	DIR_SEL_NTV	3445
			00000000'	EF	9F	00797	PUSHAB	56(D)	
				38	A6	0079D	PUSHAB	36(R6)	3444
				24	A6	007A0	PUSHAB	DIR_SEL_DIR	3441
			00000000'	EF	9F	007A3	PUSHAB	DIR_DESC	
				AD	9F	007A9	CALLS	#5, MATCH_DIRECTORY	3445
	00000000G	00		05	FB	007AC	BBC	#1, STATUS, 79\$	3449
		32		01	E1	007B3	MOVL	DIR_SP, R1	3452
			00000000'	EF	D0	007B7	BISB2	#4, DIR_STATUS	
				04	88	007BE	BISB2	#1, 35(Ri)	
		23		01	88	007C5	BLBC	STATUS, 77\$	3453
				50	E9	007C9	BISB2	#8, DIR_STATUS	
	00000000'	0B		08	88	007CC	BISB2	#2, 35(D)	
		23		02	88	007D3	BBC	#2, STATUS, 78\$	3454
				02	E1	007D7	BISB2	#4, 35(D)	
	04			04	88	007DB	BBC	#3, STATUS, 80\$	3455
		23		03	E1	007DF			
	2A			03	E1	007DF			

	23	A6		08	88	007E3		BISB2	#8, 35(D)			
				24	11	007E7		BRB	80\$		3449	
			28	A6	DD	007E9	79\$:	PUSHL	40(D)		3459	
		7E	50	8F	9A	007EC		MOVZBL	#80, -(SP)			
	00000000G	00		02	FB	007F0		CALLS	#2, FREE_VM			
			30	A6	DD	007F7		PUSHL	48(D)		3460	
		7E	50	8F	9A	007FA		MOVZBL	#80, -(SP)			
0044	8F	00	00000000G	00	02	FB	007FE	CALLS	#2, FREE_VM			
		6E		00	2C	00805		MOVCS	#0, (SP), #0, #68, (D)		3461	
				66		0080C						
		56	00000000'	EF	D0	0080D	80\$:	MOVL	DIR_SP, D		3467	
		15	00000000'	EF	E9	00814	81\$:	BLBC	DIR_FLAGS, 84\$		3474	
	03	00000000'		03	E0	0081B		BBS	#3, DIR_FLAGS, 83\$		3475	
				00B8	31	00823	82\$:	BRW	92\$			
		04		6B	B1	00826	83\$:	CMPW	(R11), #4		3476	
				F8	12	00829		BNEQ	82\$			
				29	A8	95	0082B	TSTB	41(N)		3477	
				F3	12	0082E		BNEQ	82\$			
		01	00000000'	EF	91	00830	84\$:	CMPB	DIR_STRUCLEV, #1		3490	
				13	12	00837		BNEQ	85\$			
		50	00000000'	EF	9A	00839		MOVZBL	DIR_LEVELS, R0		3492	
		00000000'	EF40	59	D1	00840		CMPL	V, DIR_SCANLIMIT-4[R0]			
				40	1B	00848		BLEQU	89\$			
				25	11	0084A		BRB	86\$			
		50		24	A6	9E	0084C	85\$:	MOVAB	36(D), R0	3494	
				60	B5	00850		TSTW	(R0)			
				36	13	00852		BEQL	89\$			
		7E		38	A6	3C	00854	MOVZWL	56(D), -(SP)		3503	
				04	A0	DD	00858	PUSHL	4(R0)		3502	
		7E		60	3C	0085B		MOVZWL	(R0), -(SP)		3501	
				0C	AE	DD	0085E	PUSHL	VERSION		3500	
				01	AA	9F	00861	PUSHAB	1(NAME)		3499	
		7E		6A	9A	00864		MOVZBL	(NAME), -(SP)			
		000000C0G		00	06	FB	00867	CALLS	#6, TERMINATE_SCAN			
				19	50	E9	0086E	BLBC	R0, 89\$			
	0A	00000000'		EF	02	E1	00871	86\$:	BBC	#2, DIR_FLAGS, 88\$	3507	
		00000000'		EF	02	88	00879		BISB2	#2, DIR_FLAGS	3510	
					50	D4	00880	87\$:	CLRL	R0	3511	
					04	00882		RET				
		66		10	A6	D0	00883	88\$:	MOVL	16(D), (D)	3517	
				18	A6	D4	00887		CLRL	24(D)	3518	
		10	A6		66	D1	0088A	89\$:	CMPL	(D), 16(D)	3525	
					5E	1E	0088E		BGEQU	93\$		
	59	23	A6		01	E1	00890		BBC	#1, 35(D), 93\$	3526	
			50	00000000'	EF	D0	00895		MOVL	DIR_SEL_LATEST, R0	3527	
					08	14	0089C		BGT?	90\$		
50		42	A6		00	EC	0089E		CMPV	#0, #16, 66(D), R0		
					48	12	008A4		BNEQ	93\$		
		0A	00000000'	EF	03	E1	008A6	90\$:	BBC	#3, DIR_FLAGS, 91\$	3528	
				04	6B	B1	008AE		CMPW	(R11), #4	3529	
					05	12	008B1		BNEQ	91\$		
				29	A8	95	008B3		TSTB	41(N)	3530	
					36	13	008B6		BEQL	93\$		
			50		03	A8	9A	008B8	91\$:	MOVZBL	3(N), R0	3539
			50		04	A8	C0	008BC		ADDL2	4(N), #0	
DC		AD		50	4C	A8	C3	008C0		SUBL3	76(N), R0, NTV_DESC	
			E0	AD	4C	A8	D0	0C8C6		MOVL	76(N), NTV_DESC+4	3540

		00000000'	EF	9F	008CB	PUSHAB	DIR_SEL_NTV	:	3541
		DC	AD	9F	008D1	PUSHAB	NTV_DESC	:	
00000000G	00		02	FB	008D4	CALLS	#2, MATCH_FILENAME	:	
	10		50	E9	008DB	BLBC	R0, 93\$:	
	1D	00000000'	EF	E9	008DE	92\$:	BLBC	DIR_STATUS, 96\$	3544
00000000'	EF		02	88	008E5	BISB2	#2, DIR_STATUS	:	3545
			14	11	008EC	BRB	96\$:	3543
03 00000000'	EF		03	E0	008EE	93\$:	BBS	#3, DIR_FLAGS, 95\$	3554
			F752	31	008F6	94\$:	BRW	2\$:
	F9	23	A6	E9	008F9	95\$:	BLBC	35(D), 94\$:
F4	23		03	E1	008FD		BBC	#3, 35(D), 94\$:
			50	01	00902	96\$:	MOVL	#1, R0	:
				04	00905		RET	:	3562

: Routine Size: 2310 bytes, Routine Base: CODE + 0F84

```

: 2469 3563 1 %SBTTL 'FREE DIR DATA - free directory scan context'
: 2470 3564 1 GLOBAL ROUTINE FREE_DIR_DATA: NOVALUE=
: 2471 3565 1
: 2472 3566 1 !++
: 2473 3567 1
: 2474 3568 1 FUNCTIONAL DESCRIPTION:
: 2475 3569 1 This routine deletes the directory scan context.
: 2476 3570 1
: 2477 3571 1 INPUT PARAMETERS:
: 2478 3572 1 NONE
: 2479 3573 1
: 2480 3574 1 IMPLICIT INPUTS:
: 2481 3575 1 Directory scan context.
: 2482 3576 1
: 2483 3577 1 OUTPUT PARAMETERS:
: 2484 3578 1 NONE
: 2485 3579 1
: 2486 3580 1 IMPLICIT OUTPUTS:
: 2487 3581 1 Directory scan context.
: 2488 3582 1
: 2489 3583 1 ROUTINE VALUE:
: 2490 3584 1 NONE
: 2491 3585 1
: 2492 3586 1 SIDE EFFECTS:
: 2493 3587 1 NONE
: 2494 3588 1
: 2495 3589 1 --
: 2496 3590 1
: 2497 3591 2 BEGIN
: 2498 3592 2
: 2499 3593 2 ! Free any dynamic storage that is currently allocated.
: 2500 3594 2
: 2501 3595 2 INCRA D FROM DIR_STACK TO DIR_STACK+D_K_NLEVELS*D_S_ENTRY-D_S_ENTRY BY D_S_ENTRY DO
: 2502 3596 3 BEGIN
: 2503 3597 3 MAP
: 2504 3598 3 D: REF BBLOCK; ! Pointer to directory stack entry
: 2505 3599 3
: 2506 3600 3 IF .D[D_BUF_ADDR] NEQ 0
: 2507 3601 3 THEN
: 2508 3602 3 FREE_VM(.D[D_BUF_LEN], .D[D_BUF_ADDR]);
: 2509 3603 3 IF .BBLOCK[D[D_TERM_DESC], DSC$A_POINTER] NEQ 0
: 2510 3604 3 THEN
: 2511 3605 3 FREE_VM(DIR$$NAME, .BBLOCK[D[D_TERM_DESC], DSC$A_POINTER]);
: 2512 3606 3 IF .BBLOCK[D[D_NAME_DESC], DSC$A_POINTER] NEQ 0
: 2513 3607 3 THEN
: 2514 3608 3 FREE_VM(DIR$$NAME, .BBLOCK[D[D_NAME_DESC], DSC$A_POINTER]);
: 2515 3609 2 END;
: 2516 3610 2
: 2517 3611 2
: 2518 3612 2 ! Reinitialize the impure storage.
: 2519 3613 2
: 2520 3614 2 (H$FILL(0, DIR_END-DIR_BEG, DIR_BEG);
: 2521 3615 1 END;

```

			00FC 00000	.ENTRY	FREE_DIR_DATA, Save R2,R3,R4,R5,R6,R7	: 3564
57	00000000'	EF	9E 00002	MOVAB	DIR_STACK, R7	:
56	00000000G	00	9E 00009	MOVAB	FREE_VM, R6	:
52		67	9E 00010	MOVAB	DIR_STACK, R2	: 3595
53	0220	C7	9E 00013	MOVAB	DIR_STACK+544, R3	:
		2E	11 00018	BRB	5\$:
	0C	A2	D5 0001A	TSTL	12(D)	: 3600
		07	13 0001D	BEQL	2\$:
7E	08	A2	7D 0001F	MOVQ	8(D), -(SP)	: 3602
66		02	FB 00023	CALLS	#2, FREE_VM	:
	28	A2	D5 00026	TSTL	40(D)	: 3603
		0A	13 00029	BEQL	3\$:
	28	A2	DD 0002B	PUSHL	40(D)	: 3605
7E	50	8F	9A 0002E	MOVZBL	#80, -(SP)	:
66		02	FB 00032	CALLS	#2, FREE_VM	:
	30	A2	D5 00035	TSTL	48(D)	: 3606
		0A	13 00038	BEQL	4\$:
	30	A2	DD 0003A	PUSHL	48(D)	: 3608
7E	50	8F	9A 0003D	MOVZBL	#80, -(SP)	:
66		02	FB 00041	CALLS	#2, FREE_VM	:
52	44	A2	9E 00044	MOVAB	68(R2), D	: 3595
53		52	D1 00048	CML	D, R3	:
		CD	1B 0004B	BLEQU	1\$:
OSCC	8F		00	MOVCS	#0, (SP), #0, #972, DIR_BEG	: 3614
		00	2C 0004D			:
	FEA0	C7	00054			:
		04	00057	RET		: 3615

: Routine Size: 88 bytes, Routine Base: CODE + 188A

FASTSCAN
V04-000

Fast file scan
FREE_DIR_DATA - free directory scan context

F 9
15-Sep-1984 23:56:53
14-Sep-1984 11:53:52

VAX-11 Bliss-32 V4.0-742
[BACKUP.SRC]FASTSCAN.B32;1

: 2523 3616 1 END
: 2524 3617 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
COMMON	2124	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, OVR, NOPIC, ALIGN(2)
CODE	6370	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	Symbols		Pages Mapped	Processing Time
	Total	Loaded Percent		
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	196 1	1000	00:02.3

COMMAND QUALIFIERS

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

: Size: 6262 code + 2232 data bytes
: Run Time: 01:53.6
: Elapsed Time: 06:24.4
: Lines/CPU Min: 1911
: Lexemes/CPU-Min: 26165
: Memory Used: 875 pages
: Compilation Complete

