

```

BBBBBBBBBBBBBB      AAAAAAAAAA      CCCCCCCCCCCCCC      KKK      KKK      UUU      UUU      PPPPPPPPPPPP
BBBBBBBBBBBBBB      AAAAAAAAAA      CCCCCCCCCCCCCC      KKK      KKK      UUU      UUU      PPPPPPPPPPPP
BBBBBBBBBBBBBB      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
BBB      BBB      AAA      AAA      CCC      KKK      KKK      UUU      UUU      PPP      PPP
BBBBBBBBBBBBBB      AAA      AAA      CCC      KKKKKKKKK      UUU      UUU      PPPPPPPPPPPP
BBBBBBBBBBBBBB      AAA      AAA      CCC      KKKKKKKKK      UUU      UUU      PPPPPPPPPPPP
BBBBBBBBBBBBBB      AAA      AAA      CCC      KKKKKKKKK      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
BBB      BBB      AAA      AAA      CCC      KKK      KKK      UUU      UUU      PPP
BBBBBBBBBBBBBB      AAA      AAA      CCCCCCCCCCCCCC      KKK      KKK      UUUUUUUUUUUUUUU      PPP
BBBBBBBBBBBBBB      AAA      AAA      CCCCCCCCCCCCCC      KKK      KKK      UUUUUUUUUUUUUUU      PPP
BBBBBBBBBBBBBB      AAA      AAA      CCCCCCCCCCCCCC      KKK      KKK      UUUUUUUUUUUUUUU      PPP

```

```

BBBBBBBB      AAA~AA      DDDDDDDD      BBBB~BBB      LL      000000      CCCCCCCC      KK      KK
BBBBBBBB      AAAAAA      DDDDDDDD      BBBB~BBB      LL      000000      CCCCCCCC      KK      KK
BB      BB      AA      AA      DD      DD      BB      BB      LL      00      00      CC      KK      KK
BB      BB      AA      AA      DD      DD      BB      BB      LL      00      00      CC      KK      KK
BB      BB      AA      AA      DD      DD      BB      BB      LL      00      00      CC      KK      KK
BB      BB      AA      AA      DD      DD      BB      BB      LL      00      00      CC      KK      KK
BBBBBBBB      AA      AA      DD      DD      BBBB~BBB      LL      00      00      CC      KKKKKK
BBBBBBBB      AA      AA      DD      DD      BBBB~BBB      LL      00      00      CC      KKKKKK
BB      BB      AAAAAAAAAA      DD      DD      BB      BB      LL      00      00      CC      KK      KK
BB      BB      AAAAAAAAAA      DD      DD      BB      BB      LL      00      00      CC      KK      KK
BB      BB      AA      AA      DD      DD      BB      BB      LL      00      00      CC      KK      KK
BB      BB      AA      AA      DD      DD      BB      BB      LL      00      00      CC      KK      KK
BBBBBBBB      AA      AA      DDDDDDDD      BBBB~BBB      LLLLLLLLLL      000000      CCCCCCCC      KK      KK
BBBBBBBB      AA      AA      DDDDDDDD      BBBB~BBB      LLLLLLLLLL      000000      CCCCCCCC      KK      KK

```

```

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 BADBLOCK (%TITLE 'Scan bad block information'
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 routines that do the bad block processing.
37 0037 1 The code in this module is derived from and should track module INIBAD
38 0038 1 in the INIT facility.
39 0039 1
40 0040 1 ENVIRONMENT:
41 0041 1 VAX/VMS user mode.
42 0042 1 --
43 0043 1
44 0044 1
45 0045 1 AUTHOR: M. Jack, CREATION DATE: 23-Sep-1980
46 0046 1
47 0047 1 MODIFIED BY:
48 0048 1
49 0049 1 V03-002 ACG0316 Andrew C. Goldstein, 11-Feb-1983 11:49
50 0050 1 Fix handling of software last track bad block data
51 0051 1
52 0052 1 V03-001 ACG0283 Andrew C. Goldstein, 8-Apr-1982 10:33
53 0053 1 Disable bad block processing on MSCP disks
54 0054 1
55 0055 1 V02-003 MLJ0054 Martin L. Jack, 22-Nov-1981 21:40
56 0056 1 Integrate GET_VM and FREE_VM jacket routines.
57 0057 1

```

BADBLOCK  
V04 000

Scan bad block information

N 5  
15-Sep-1984 23:42:03 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 11:53:47 [BACKUP.SRC]BADBLOCK.B32;1

:	58	0058	1	!	V02-002	MLJ0011	Martin L. Jack, 26-Mar-1981	14:20
:	59	0059	1	!			Remove extra indirection in BADBLOCK_LBN and BADBLOCK_CNT definitions	
:	60	0060	1	!				
:	61	0061	1	!	V02-001	MLJ0010	Martin L. Jack, 25-Mar-1981	15:05
:	62	0062	1	!			Add capability to round to cluster boundaries for standalone	
:	63	0063	1	!			volume initialize. Replace OWN storage with LOCAL.	
:	64	0064	1	!				
:	65	0065	1	!*				

```

: 67 0066 1 REQUIRE 'SRC$:COMMON';
: 68 1172 1 LIBRARY 'SYS$LIBRARY:LIB';
: 69 1173 1
: 70 1174 1
: 71 1175 1 LINKAGE
: 72 1176 1 L_PS = CALL: GLOBAL(PS=11);
: 73 1177 1
: 74 1178 1
: 75 1179 1 MACRO
: 76 1180 1 L_DECL= EXTERNAL REGISTER PS = 11: REF VECTOR %;
: 77 1181 1
: 78 1182 1
: 79 1183 1 FORWARD ROUTINE
: 80 1184 1 GET_BADBLOCKS, ! Main level bad block processing
: 81 1185 1 GET_FACTBAD: L_PS, ! Process factory bad block data
: 82 1186 1 GET_SOFTBAD: L_PS NOVALUE, ! Process bad block scan program data
: 83 1187 1 MARR_BAD: L_PS NOVALUE; ! Enter bad block in allocation table
: 84 1188 1
: 85 1189 1
: 86 1190 1 EXTERNAL ROUTINE
: 87 1191 1 CHECKSUM2, ! Compute block checksum
: 88 1192 1 FILE_ERROR: NOVALUE, ! Issue file-related error
: 89 1193 1 GET_VM; ! Allocate virtual memory
: 90 1194 1
: 91 1195 1
: 92 1196 1 EXTERNAL LITERAL
: 93 1197 1 BACKUP$_READBAD,
: 94 1198 1 BACKUP$_DIAGPACK,
: 95 1199 1 BACKUP$_NOBADDATA,
: 96 1200 1 BACKUP$_FACTBAD,
: 97 1201 1 BACKUP$_MAXBAD;

```

```

: 99      1202 1 LITERAL
: 100     1203 1          BADBLOCK_MAX= 128;          ! Length of bad block table
: 101     1204 1
: 102     1205 1
: 103     1206 1 MACRO
: 104     1207 1          BADBLOCK FAB= PS[0] %,          ! FAB for device
: 105     1208 1          BADBLOCK CHAN= PS[1] %,        ! Channel number
: 106     1209 1          DEVICE CHAR= PS[2] %,         ! Device characteristics
: 107     1210 1          CLUSTER= PS[3] %,            ! Cluster factor
: 108     1211 1          CLUST_PRESENT= PS[4] %,       ! Cluster factor present
: 109     1212 1          SERIAL_NUMBER= PS[5] %,       ! Pack serial number
: 110     1213 1          BADBLOCK_TOTAL= PS[6] %,      ! Count of bad areas so far
: 111     1214 1          _BADBLOCK_LBN= PS[7] %,       ! Bad block LBN table
: 112     1215 1          _BADBLOCK_CNT= PS[7+BADBLOCK_MAX] %, ! Bad block count table
: 113     1216 1          BADBLOCK FAB(O,P,S,E)= BBLOCK[._BADBLOCK FAB,O,P,S,E] %,
: 114     1217 1          DEVICE CHAR(O,P,S,E)= BBLOCK[._DEVICE CHAR,O,P,S,E] %,
: 115     1218 1          BADBLOCK_LBN(N)= VECTOR[_BADBLOCK_LBN,N] %,
: 116     1219 1          BADBLOCK_CNT(N)= VECTOR[_BADBLOCK_CNT,N] %;
: 117     1220 1
: 118     1221 1
: 119     1222 1 LITERAL
: 120     1223 1          PS_SIZE= 7 + BADBLOCK_MAX + BADBLOCK_MAX;

```

```

122 1224 1 %SBTTL 'GET_BADBLOCKS - main bad block routine'
123 1225 1 GLOBAL ROUTINE GET_BADBLOCKS(FAB,CHAN,DEVCHAR,CLUST)=
124 1226 1
125 1227 1 |++
126 1228 1
127 1229 1 | FUNCTIONAL DESCRIPTION:
128 1230 1 | This is the main bad block processing routine. It calls the factory
129 1231 1 | data and software data bad block routines as appropriate.
130 1232 1
131 1233 1 | INPUT PARAMETERS:
132 1234 1 | FAB - Pointer to FAB.
133 1235 1 | CHAN - Channel assigned to the device.
134 1236 1 | DEVCHAR - Pointer to device characteristics for the device.
135 1237 1 | CLUST - Cluster factor (optional).
136 1238 1 | If present, bad areas are rounded to cluster
137 1239 1 | boundaries, and the last track is marked bad.
138 1240 1
139 1241 1 | IMPLICIT INPUTS:
140 1242 1 | NONE
141 1243 1
142 1244 1 | OUTPUT PARAMETERS:
143 1245 1 | NONE
144 1246 1
145 1247 1 | IMPLICIT OUTPUTS:
146 1248 1 | NONE
147 1249 1
148 1250 1 | ROUTINE VALUE:
149 1251 1 | Pointer to a bad block descriptor, allocated by LIB$GET_VM.
150 1252 1
151 1253 1 | SIDE EFFECTS:
152 1254 1 | Disk bad block data read
153 1255 1
154 1256 1 |--
155 1257 1
156 1258 2 BEGIN
157 1259 2 LOCAL
158 1260 2 PSAREA: VECTOR[PS SIZE], ! Impure area
159 1261 2 AREA: REF BBLOCK, ! Pointer to gotten area
160 1262 2 DESC: REF BBLOCK; ! Pointer to descriptor
161 1263 2 GLOBAL REGISTER
162 1264 2 PS = 11: REF VECTOR; ! Impure storage base
163 1265 2 BUILTIN
164 1266 2 ACTUALCOUNT;
165 1267 2
166 1268 2
167 1269 2 ! Initialize the impure storage so the inner routines can access it.
168 1270 2
169 1271 2 PS = PSAREA;
170 1272 2 BADBLOCK FAB = .FAB;
171 1273 2 BADBLOCK CHAN = .CHAN;
172 1274 2 DEVICE CHAR = .DEVCHAR;
173 1275 2 CLUST_PRESENT = FALSE;
174 1276 2 CLUSTER = 1;
175 1277 2 IF ACTUALCOUNT() GEQ 4
176 1278 2 THEN
177 1279 2 BEGIN
178 1280 3 CLUST_PRESENT = TRUE;

```

```

179 1281 3 CLUSTER = .CLUST;
180 1282 2 END;
181 1283 2 BADBLOCK_TOTAL = 0;
182 1284 2 SERIAL_NUMBER = 0;
183 1285 2
184 1286 2
185 1287 2 ! Establish whether the volume has factory bad block data or not and
186 1288 2 ! call the appropriate routine.
187 1289 2
188 1290 2 IF .DEVICE_CHAR[DIBSL_MAXBLOCK] GTRU SMALL_DISK
189 1291 2 AND NOT .DEVICE_CHAR[DEVSV_RCT]
190 1292 2 THEN
191 1293 2 BEGIN
192 1294 2 IF NOT GET_FACTBAD() THEN GET_SOFTBAD();
193 1295 2 END
194 1296 2 ELSE
195 1297 2 BEGIN
196 1298 2 IF .CLUST_PRESENT
197 1299 2 AND .DEVICE_CHAR[DIBSL_MAXBLOCK] LSSU
198 1300 2 ((.DEVICE_CHAR[DIBSL_MAXBLOCK] + .CLUSTER - 1) / .CLUSTER) * .CLUSTER
199 1301 2 THEN MARK_BAD(1, .DEVICE_CHAR[DIBSL_MAXBLOCK]);
200 1302 2 END;
201 1303 2
202 1304 2
203 1305 2 ! Get the descriptor from heap storage and initialize.
204 1306 2
205 1307 2 AREA = GET_VM(BAD_S_HEADER + .BADBLOCK_TOTAL*BAD_S_DESC);
206 1308 2 AREA[BAD_NUMDESC] = .BADBLOCK_TOTAL;
207 1309 2 AREA[BAD_SERIAL] = .SERIAL_NUMBER;
208 1310 2 DESC = AREA[BAD_DESC];
209 1311 2 INCR I FROM 0 TO .BADBLOCK_TOTAL-1 DO
210 1312 2 BEGIN
211 1313 2 DESC[BAD_LBN] = .BADBLOCK_LBN[I];
212 1314 2 DESC[BAD_COUNT] = .BADBLOCK_CNT[I];
213 1315 2 DESC = .DESC + BAD_S_DESC;
214 1316 2 END;
215 1317 2
216 1318 2
217 1319 2 ! Return a pointer to the bad block descriptor.
218 1320 2
219 1321 2 .AREA
220 1322 1 END;

```

```

.TITLE BADBLOCK Scan bad block information
.IDENT \V04-000\

.EXTRN CHECKSUM2, FILE_ERROR
.EXTRN GET_VM, BACKUP$_READBAD
.EXTRN BACKUP$_DIAGPACK
.EXTRN BACKUP$_NOBADDATA
.EXTRN BACKUP$_FACTBAD
.EXTRN BACKUP$_MAXBAD

.PSECT CODE,NOWRT,2

.ENTRY GET_BADBLOCKS, Save R2,R3,R4,R11

```

081C 00000

: 1225



	5E	FBE4	CE	9E	00002	MOVAB	-1052(SP), SP		
	5B		6E	9E	00007	MOVAB	PSAREA, PS	1271	
	6B	04	AC	7D	0000A	MOVQ	FAB, (PS)	1272	
08	AB	0C	AC	D0	0000E	MOVL	DEVCHAR, 8(PS)	1274	
0C	AB		01	7D	00013	MOVQ	#1, 12(PS)	1276	
	04		6C	91	00017	CMPB	(AP), #4	1277	
			09	1F	0001A	BLSSU	1\$		
10	AB		01	D0	0001C	MOVL	#1, 16(PS)	1280	
0C	AB	10	AC	D0	00020	MOVL	CLUST, 12(PS)	1281	
		14	AB	7C	00025	CLRQ	20(PS)	1284	
	52	08	AB	D0	00028	MOVL	8(PS), R2	1290	
00001000	8F	70	A2	D1	0002C	CMP	112(R2), #4096		
			13	1B	00034	BLEQU	2\$		
	0F	01	A2	E8	00036	BLBS	1(R2), 2\$	1291	
0000V	CF		00	FB	0003A	CALLS	#0, GET_FACTBAD	1294	
	2B		50	E8	0003F	BLBS	R0, 3\$		
0000V	CF		00	FB	00042	CALLS	#0, GET_SOFTBAD		
			24	11	00047	BRB	3\$	1290	
50	70	20	10	AB	E9	00049	2\$: BLBC	16(PS), 3\$	1298
		A2	0C	AB	C1	0004D	ADDL3	12(PS), 112(R2), R0	1300
			50	D7	00053	DECL	R0		
		50	0C	AB	C6	00055	DIVL2	12(PS), R0	
		50	0C	AB	C4	00059	MULL2	12(PS), R0	
		50	70	A2	D1	0005D	CMP	112(R2), R0	
			70	0A	1E	00061	BGEQU	3\$	
				A2	DD	00063	PUSHL	112(R2)	1301
				01	DD	00066	PUSHL	#1	
0000V	CF		02	FB	00068	CALLS	#2, MARK BAD		
7E	54	18	AB	D0	0006D	3\$: MOVL	24(PS), R4	1307	
	54		03	78	00071	ASHL	#3, R4, -(SP)		
	6E		08	C0	00075	ADDL2	#8, (SP)		
00000000G	00		01	FB	00078	CALLS	#1, GET_VM		
	52		50	D0	0007F	MOVL	R0, AREA		
	62		54	D0	00082	MOVL	R4, (AREA)	1308	
04	A2	14	AB	D0	00085	MOVL	20(PS), 4(AREA)	1309	
	50	08	A2	9E	0008A	MOVAB	8(R2), DESC	1310	
	53	1C	AB	9E	0008E	MOVAB	28(PS), R3	1313	
	5B	021C	CB	9E	00092	MOVAB	540(R11), R11	1314	
	51		01	CE	00097	MNEGL	#1, I		
			08	11	0009A	BRB	5\$		
	80		6341	D0	0009C	4\$: MOVL	(R3)[I], (DESC)+	1313	
	8C		6B41	D0	000A0	MOVL	(R11)[I], (DESC)+	1314	
F4	51		54	F2	000A4	5\$: AOBSS	R4, I, 4\$	1311	
	50		52	D0	000A8	MOVL	AREA, R0	1322	
			04	000AB	RET				

: Routine Size: 172 bytes, Routine Base: CODE + 0000

```

222 1323 1 %SBTTL 'GET_FACTBAD - get factory bad block data'
223 1324 1 ROUTINE GET_FACTBAD: L_P$=
224 1325 1
225 1326 1 !++
226 1327 1
227 1328 1 FUNCTIONAL DESCRIPTION:
228 1329 1 This routine processes the factory bad block data found on the last
229 1330 1 track of the disk.
230 1331 1
231 1332 1 INPUT PARAMETERS:
232 1333 1 NONE
233 1334 1
234 1335 1 IMPLICIT INPUTS:
235 1336 1 Impure area.
236 1337 1
237 1338 1 OUTPUT PARAMETERS:
238 1339 1 NONE
239 1340 1
240 1341 1 IMPLICIT OUTPUTS:
241 1342 1 Impure area.
242 1343 1
243 1344 1 ROUTINE VALUE:
244 1345 1 True if factory data found, false if not found.
245 1346 1
246 1347 1 SIDE EFFECTS:
247 1348 1 Disk blocks read
248 1349 1
249 1350 1 --
250 1351 1
251 1352 2 BEGIN
252 1353 2 MACRO
253 1354 2 PBN_SECTOR = 0, 0, 8, 0%, ! format of physical block number
254 1355 2 PBN_TRACK = 1, 0, 8, 0%, ! sector number
255 1356 2 PBN_CYLINDER = 2, 0, 16, 0%; ! track number
256 1357 2
257 1358 2 LABEL
258 1359 2 SEARCH_TRACK; ! main loop to search last track of disk
259 1360 2
260 1361 2 LOCAL
261 1362 2 LBN, ! LBN to mark bad
262 1363 2 BLOCKFACT, ! blocking factor of disk
263 1364 2 FIRST_TIME, ! first time through flag
264 1365 2 FIRST_BUFFER, ! first buffer flag
265 1366 2 NOGOOD, ! no blocks read without errors
266 1367 2 IOSB : VECTOR[4,WORD], ! I/O status block
267 1368 2 STATUS, ! return status
268 1369 2 P : REF BBLOCK, ! pointer into bad block descriptors
269 1370 2 PHYS_BLOCK : BBLOCK [4], ! current physical block number
270 1371 2 BUFFER : BBLOCK [512], ! I/O buffer
271 1372 2 BUFFER2 : BBLOCK [512]; ! buffer for second copy of data
272 1373 2 L_DECL;
273 1374 2
274 1375 2
275 1376 2 ! Compute the blocking factor.
276 1377 2
277 1378 3 BLOCKFACT = (.DEVICE_CHAR[DIB$B_SECTORS]
278 1379 3 * .DEVICE_CHAR[DIB$B_TRACKS]

```



```

336 1437 7      END
337 1438 6      ELSE
338 1439 7      BEGIN
339 1440 7      IF CH$EQL (512, BUFFER, 512, BUFFER2, 0)
340 1441 7      THEN EXITLOOP;
341 1442 6      END;
342 1443 6      END
343 1444 5      ELSE IF .STATUS NEQ SSS_PARITY
344 1445 5      THEN
345 1446 5      FILE_ERROR(
346 1447 5      BACKUP$_READBAD, ._BADBLOCK_FAB, .STATUS);
347 1448 5
348 1449 5      PHYS_BLOCK[PBN_SECTOR] = .PHYS_BLOCK[PBN_SECTOR]
349 1450 5      + 2*.BLOCKFACT;
350 1451 5      IF .PHYS_BLOCK[PBN_SECTOR] GEQ .DEVICE_CHAR[DIB$B_SECTORS]
351 1452 5      THEN LEAVE SEARCH_TRACK;
352 1453 4      END;
353 1454 4      ! end of block search loop
354 1455 4
355 1456 4      ! We have a good bad block list. Process its entries.
356 1457 4      !
357 1458 4      IF .FIRST TIME
358 1459 4      THEN SERIAL_NUMBER = .BUFFER[BBDS$L_SERIAL];
359 1460 4
360 1461 4      IF .BUFFER[BBDS$W_FLAGS] EQL 65535
361 1462 4      THEN FILE_ERROR(BACKUP$_DIAGPACK, ._BADBLOCK_FAB);
362 1463 4
363 1464 4      P = BUFFER + BBDS$C_DESCRIPTOR;
364 1465 4      DO
365 1466 5      BEGIN
366 1467 5      IF .P[BBDS$V_CYLINDER] EQL 32767
367 1468 5      THEN EXITLOOP;
368 1469 7      LBN = ((.P[BBDS$V_CYLINDER] * .DEVICE_CHAR[DIB$B_TRACKS]
369 1470 6      + .P[BBDS$V_TRACK]) * .DEVICE_CHAR[DIB$B_SECTORS]
370 1471 5      + .P[BBDS$V_SECTOR]) / .BLOCKFACT;
371 1472 5      MARK_BAD (1, .LBN);
372 1473 5      P = .P + BBDS$C_ENTRY;
373 1474 5      END
374 1475 4      UNTIL .P GEQA BUFFER+512;
375 1476 4
376 1477 4
377 1478 4      ! If we are not yet into the user data, position to it and try again.
378 1479 4      !
379 1480 4      FIRST TIME = FALSE;
380 1481 4      IF .PHYS_BLOCK[PBN_SECTOR] GEQ 10*.BLOCKFACT THEN EXITLOOP;
381 1482 4      PHYS_BLOCK[PBN_SECTOR] = 10*.BLOCKFACT;
382 1483 3      END;
383 1484 2      END;
384 1485 2
385 1486 2
386 1487 2      ! If we found no good data at all, complain.
387 1488 2      !
388 1489 2      IF .NOGOOD
389 1490 2      THEN FILE_ERROR(BACKUP$_FACTBAD, ._BADBLOCK_FAB);
390 1491 2
391 1492 2
392 1493 2      RETURN NOT .FIRST_TIME;

```

: 393

1494 1 END;

: end of routine GET\_FACTBAD

			.EXTRN SYS\$QIOW				
			07FC 00000 GET_FACTBAD:				
		5E	FBE0	CE 9E	00002	.WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10 : 1324	
		50	08	AB D0	00007	MOVAB -1056(SP), SP : 1378	
		51	08	A0 9A	00008	MOVL 8(P\$), R0 : 1379	
		52	09	A0 9A	0000F	MOVZBL 8(R0), R1 : 1380	
		51		52 C4	00013	MOVZBL 9(R0), R2 : 1381	
		53	0A	A0 3C	00016	MULL2 R2, R1 : 1382	
		51		53 C4	0001A	MOVZWL 10(R0), R3 : 1383	
55		51	70	A0 C7	0001D	MULL2 R3, R1 : 1384	
		13	10	AB E9	00022	DIVL3 112(R0), R1, BLOCKFACT : 1385	
		51	08	A0 9A	00026	BLBC 16(P\$), 1\$ : 1386	
		51		55 C6	0002A	MOVZBL 8(R0), R1 : 1387	
7E	70	A0		51 C3	0002D	DIVL2 BLOCKFACT, R1 : 1388	
				51 DD	00032	SUBL3 R1, 112(R0), -(SP) : 1389	
	0000V	CF		02 F9	00034	PUSHL R1 : 1390	
	0C	AE		01 D0	00039	CALLS #2, MARK BAD : 1400	
		50	08	AB D0	0003C	MOVL #1, FIRST TIME : 1401	
16	AE	0A	A0	01 A3	00044	MOVL #1, NOGOOD : 1402	
15	AE	09	A0	01 83	0004A	MOVL 8(P\$), R0 : 1403	
				14	AE 94	00050	SUBW3 #1, 10(R0), PHYS_BLOCK+2 : 1404
08	AE			14	AE 94	00050	SUBB3 #1, 9(R0), PHYS_BLOCK+1 : 1405
				55	0A C5	00053	CLRB PHYS_BLOCK : 1406
		58	14	AE 9A	0005C	MULL3 #10, BLOCKFACT, 8(SP) : 1407	
		6E	08	AB 9E	00060	MOVL #1, FIRST BUFFER : 1408	
59		55		01 78	00064	MOVZBL PHYS_BLOCK, R8 : 1409	
		57	00	BE D0	00068	MOVAB 8(P\$), (SP) : 1410	
		50	16	AE 3C	0006C	ASHL #1, BLOCKFACT, R9 : 1411	
		51	09	A7 9A	00070	MOVL #0(SP), R7 : 1412	
		50		51 C4	00074	MOVZWL PHYS_BLOCK+2, R0 : 1413	
		52	15	AE 9A	00077	MOVZBL 9(R7), R1 : 1414	
		50		52 C0	0007B	MULL2 R1, R0 : 1415	
		51	08	A7 9A	0007E	MOVZBL PHYS_BLOCK+1, R2 : 1416	
		50		51 C4	00082	ADDL2 R2, R0 : 1417	
04	AE	50		58 C0	00085	MOVZBL 8(R7), R1 : 1418	
				55 C7	00088	MULL2 R1, R0 : 1419	
				7E 7C	0008D	ADDL2 R8, R0 : 1420	
				7E D4	0008F	DIVL3 BLOCKFACT, R0, LBN : 1421	
				10	AE DD	00091	CLRB -(SP) : 1422
		7E	0200	8F 3C	00094	CLRL -(SP) : 1423	
		07	24	AE E9	00099	PUSHL LBN : 1424	
		50	FD8	CD 9E	0009D	MOVZWL #512, -(SP) : 1425	
				04 11	000A2	BLBC FIRST_BUFFER, 4\$ : 1426	
				50	2C AE	9E 000A4	MOVAB BUFFER, R0 : 1427
				50	DD	000A8	BRB 5\$ : 1428
				7E	7C	000AA	MOVAB BUFFER2, R0 : 1429
				F8	AD	9F 000AC	PUSHL R0 : 1430
				04	AB	DD 000AF	CLRB -(SP) : 1431
				7E	D4	000B4	PUSHAB IOSB : 1432
							PUSHL #33 : 1433
							PUSHL 4(P\$) : 1434
							CLRL -(SP) : 1435

			00000000G	00		0C	FB	000B6	CALLS	#12, SYSSQIOW	
				5A		50	D0	000BD	MOVL	R0, STATUS	
				2A		5A	E9	000C0	BLBC	STATUS, 7\$	1425
				5A	F8	AD	3C	000C3	MOVZWL	IOSB, STATUS	
				23		5A	E9	000C7	BLBC	STATUS, 7\$	1428
						OC	AE	D4	000CA	CLRL	NOGOOD
				OF		10	AE	E9	000CD	BLBC	FIRST_BUFFER, 6\$
			FFFFFFF	8F		F4	AD	D1	000D1	CMPL	BUFFER+508, #-1
							2C	12	000D9	BNEQ	8\$
						10	AE	D4	000DB	CLRL	FIRST_BUFFER
						27	11	000DE	BRB	8\$	1436
	18	AE	FD8	CD	0200	8F	29	000E0	6\$: CMPC3	#512, BUFFER, BUFFER2	1432
						1C	12	000E9	BNEQ	8\$	1432
			000001F4	8F		2E	11	000EB	BRB	10\$	1440
						5A	D1	000ED	7\$: CMPL	STATUS, #500	1441
						11	13	000F4	BEQL	8\$	1444
						5A	DD	000F6	PUSHL	STATUS	1447
						6B	DD	000F8	PUSHL	(P\$)	1446
			00000000G	00	00000000G	8F	DD	000FA	PUSHL	#BACKUP\$_READBAD	1446
			14	AE		03	FB	00100	CALLS	#3, FILE_ERROR	1450
				58	14	59	80	00107	8\$: ADDB2	R9, PHYS_BLOCK	1451
				58	08	AE	9A	0010B	MOVZBL	PHYS_BLOCK, R8	1451
						A7	91	0010F	CMPL	8(R7), R8	
						03	1A	00113	BGTRU	9\$	
						00B9	31	00115	BRW	15\$	
						FF4D	31	00118	9\$: BRW	3\$	
				06		56	E9	0011B	10\$: BLBC	FIRST_TIME, 11\$	1458
			14	AB	FD8	CD	D0	0011E	MOVL	BUFFER, 20(P\$)	1459
			FFFF	8F	FD8	CD	B1	00124	11\$: CMPW	BUFFER+6, #65535	1461
						OF	12	0012B	BNEQ	12\$	1462
						6B	DD	0012D	PUSHL	(P\$)	1462
			00000000G	00	00000000G	8F	DD	0012F	PUSHL	#BACKUP\$_DIAGPACK	
				54	FE00	02	FB	00135	CALLS	#2, FILE_ERROR	1464
			00007FFF	8F		CD	9E	0013C	12\$: MOVAB	BUFFER+8, P	1467
				OF		00	ED	00141	13\$: CMPZV	#0, #15, (P), #32767	1467
						42	13	0014A	BEQL	14\$	
				50	08	AB	D0	0014C	MOVL	8(P\$), R0	1469
				51	64	00	EF	00150	EXTZV	#0, #15, (P), R1	
				52	09	A0	9A	00155	MOVZBL	9(R0), R2	
				51		52	C4	00159	MULL2	R2, R1	
				53	03	A4	00	EF	0015C	EXTZV	#0, #7, 3(P), R3
				51		53	C0	00162	ADDL2	R3, R1	1470
				52	08	A0	9A	00165	MOVZBL	8(R0), R2	
				51		52	C4	00169	MULL2	R2, R1	
				50	02	A4	9A	0016C	MOVZBL	2(P), R0	1471
				51		50	C0	00170	ADDL2	R0, R1	
				51	04	AE	C7	00173	DIVL3	BLOCKFACT, R1, LBN	1472
						01	DD	00178	PUSHL	LBN	
						02	FB	0017D	PUSHL	#1	
			0000V	CF		04	C0	00182	CALLS	#2, MARK_BAD	1473
				54	F8	AD	9E	00185	ADDL2	#4, P	1475
				50		54	D1	00189	MOVAB	BUFFER+512, R0	1475
				50		B3	1F	0018C	CMPL	P, R0	
						56	D4	0018E	14\$: BLSSU	15\$	
						00	ED	00190	CLRL	FIRST_TIME	1480
	08	AE				08	18	00197	1481: CMPZV	#0, #8, PHYS_BLOCK, 8(SP)	1481
									BGEQ	15\$	

BADBLOCK  
V04-000

Scan bad block information  
GET\_FACTBAD - get factory bad block data

L 6  
15-Sep-1984 23:42:03  
14-Sep-1984 11:53:47

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

Page 13  
(5)

14	AE	08	AE	90	00199	MOVB	8(SP), PHYS_BLOCK	:	1482	
			FEB7	31	0019E	BRW	2\$	:	1408	
	0F	0C	AE	E9	001A1	15\$:	BLBC	NOGOOD, 16\$	:	1489
			6B	DD	001A5		PUSHL	(P\$)	:	1490
00000000G	00	00000000G	8F	DD	001A7		PUSHL	#BACKUPS_FACTBAD	:	
	56		02	FB	001AD		CALLS	#2, FILE_ERROR	:	
	50		56	D2	001B4	16\$:	MCOML	FIRST_TIME, R6	:	1493
			56	D0	001B7		MOVL	R6, R0	:	
			04	001BA			RET		:	1494

; Routine Size: 443 bytes, Routine Base: CODE + 00AC

```

395 1495 1 %SBTTL 'GET_SOFTBAD - get software detected bad block info'
396 1496 1 ROUTINE GET_SOFTBAD: L_PS NOVALUE=
397 1497 1
398 1498 1 !++
399 1499 1
400 1500 1 FUNCTIONAL DESCRIPTION:
401 1501 1 This routine processes the data left by the bad block scan program
402 1502 1 somewhere near the end of the disk.
403 1503 1
404 1504 1 INPUT PARAMETERS:
405 1505 1 NONE
406 1506 1
407 1507 1 IMPLICIT INPUTS:
408 1508 1 Impure area.
409 1509 1
410 1510 1 OUTPUT PARAMETERS:
411 1511 1 NONE
412 1512 1
413 1513 1 IMPLICIT OUTPUTS:
414 1514 1 Impure area.
415 1515 1
416 1516 1 ROUTINE VALUE:
417 1517 1 NONE
418 1518 1
419 1519 1 SIDE EFFECTS:
420 1520 1 Disk blocks read
421 1521 1
422 1522 1 !--
423 1523 1
424 1524 2 BEGIN
425 1525 2 LOCAL
426 1526 2 LBN, ! LBN to mark bad
427 1527 2 IO_STATUS : VECTOR[4,WORD], ! I/O status block
428 1528 2 STATUS, ! return status
429 1529 2 P : REF BBLOCK, ! pointer into bad block map
430 1530 2 BUFFER : BBLOCK[512]; ! I/O buffer
431 1531 2 L_DECL;
432 1532 2
433 1533 2
434 1534 2 ! Scan from the end of the volume forward to find the bad block data.
435 1535 2 ! If none is found, output a warning and proceed.
436 1536 2
437 1537 2 LBN = .DEVICE_CHAR[DIB$L_MAXBLOCK];
438 1538 2 IF
439 1539 3 BEGIN
440 1540 3 DECR J FROM 32 TO 1 DO
441 1541 4 BEGIN
442 1542 4 LBN = .LBN - 1;
443 1543 4 STATUS = $QIOW (
444 1544 4 CHAN = .BADBLOCK_CHAN,
445 1545 4 FUNC = IO$ READLBLK,
446 1546 4 IOSB = IO_STATUS[0],
447 1547 4 P1 = BUFFER,
448 1548 4 P2 = 512,
449 1549 4 P3 = .LBN
450 1550 4 );
451 1551 4 IF .STATUS THEN STATUS = .IO_STATUS[0];

```



```

452 1552 4
453 1553 4 IF .STATUS
454 1554 4 THEN
455 1555 5 BEGIN
456 1556 5 IF CHECKSUM2 (BUFFER, $BYTEOFFSET (BBM$W_CHECKSUM))
457 1557 5 AND .BUFFER[BBM$B_COUNTSIZE] EQL 1
458 1558 5 AND .BUFFER[BBM$B_LBNSIZE] EQL 3
459 1559 5 AND .BUFFER[BBM$B_INUSE] LEQ (512 - BBM$C_POINTERS - 2) / 2
460 1560 5 THEN EXITLOOP FALSE;
461 1561 5 END
462 1562 4 ELSE IF .STATUS NEQ $$$_PARITY
463 1563 4 THEN
464 1564 4 FILE_ERROR(
465 1565 4 BACKUP$_READBAD, ._BADBLOCK_FAB, .STATUS);
466 1566 4 END
467 1567 3 END
468 1568 2 THEN
469 1569 3 BEGIN
470 1570 3 FILE_ERROR(BACKUP$_NOBADDATA, ._BADBLOCK_FAB);
471 1571 3 RETURN;
472 1572 3 END;
473 1573 2
474 1574 2
475 1575 2 ! Found a good bad block descriptor. If the cluster factor parameter was
476 1576 2 ! present, enter it in the bad block map. Then, process its contents.
477 1577 2
478 1578 2 IF .CLUST_PRESENT
479 1579 2 THEN
480 1580 2 MARK_BAD(.DEVICE_CHAR[DIB$L_MAXBLOCK] - .LBN, .LBN);
481 1581 2
482 1582 2
483 1583 2 P = BUFFER + BBM$C_POINTERS;
484 1584 2 DECR J FROM .BUFFER[BBM$B_INUSE]/2 TO 1 DO
485 1585 3 BEGIN
486 1586 3 LBN = .P[BBM$W_LOWLBN];
487 1587 3 LBN<16,8> = .P[BBM$B_HIGHLBN];
488 1588 3 MARK_BAD (.P[BBM$B_COUNT]+1, .LBN);
489 1589 3 P = .P + 4;
490 1590 2 END;
491 1591 1 END;

```

! end of routine GET\_SOFTBAD

```

007C 0000 GET_SOFTBAD:
56 0000000G 00 9E 00002 .WORD Save R2,R3,R4,R5,R6
5E FDF8 CE 9E 00009 MOVAB FILE_ERROR, R6
52 08 AB D0 0000E MOVAB -520(SP), SP
54 70 A2 D0 00012 MOVL 8(P$), R2
55 20 D0 00016 MOVL 112(R2), LBN
7E 7C 00019 1$: MOVL #32, J
7E D4 0001B CLRL -(SP)
74 9F 0001D CLRL -(SP)
7E 0200 8F 3C 0001F PUSHAB -(LBN)
14 AE 9F 00024 MOVZWL #512, -(SP)
PUSHAB BUFFER

```

1496  
1537  
1550

			7E	7C	00027	CLRQ	-(SP)		
		F8	AD	9F	00029	PUSHAB	IO STATUS		
			21	DD	0002C	PUSHL	#33		
		04	AB	DD	0002E	PUSHL	4(P\$)		
			7E	D4	00031	CLRL	-(SP)		
00000000G	00		0C	FB	00033	CALLS	#12, SYSSQIOW		
	53		50	DD	0003A	MOVL	R0, STATUS		
	2D		53	E9	0003D	BLBC	STATUS, 2\$	1551	
	53	F8	AD	3C	00040	MOVZWL	IO STATUS, STATUS		
	26		53	E9	00044	BLBC	STATUS, 2\$	1553	
	7E	01FE	8F	3C	00047	MOVZWL	#510, -(SP)	1556	
		04	AE	9F	0004C	PUSHAB	BUFFER		
00000000G	00		02	FB	0004F	CALLS	#2, CHECKSUM2		
	2A		50	E9	00056	BLBC	R0, 3\$		
	01		6E	91	00059	CMPB	BUFFER, #1	1557	
			25	12	0005C	BNEQ	3\$		
	03	01	AE	91	0005E	CMPB	BUFFER+1, #3	1558	
			1F	12	00062	BNEQ	3\$		
	FD	8F	02	AE	91	00064	CMPB	BUFFER+2, #253	1559
			18	1A	00069	BGTRU	3\$		
			25	11	0006B	BRB	4\$	1560	
000001F4	8F		53	D1	0006D	2\$:	CMPL	STATUS, #500	1562
			0D	13	00074		BEQL	3\$	
			53	DD	00076		PUSHL	STATUS	1565
			6B	DD	00078		PUSHL	(P\$)	
		00000000G	8F	DD	0007A		PUSHL	#BACKUP\$_READBAD	1564
	66		03	FB	00080		CALLS	#3, FILE_ERROR	
	93		55	F5	00083	3\$:	SOBGTR	J, 1\$	1540
			6B	DD	00086		PUSHL	(P\$)	1570
		00000000G	8F	DD	00088		PUSHL	#BACKUP\$_NOBADDATA	
	66		02	FB	0008E		CALLS	#2, FILE_ERROR	
				04	00091		RET		1569
	0C	10	AB	E9	00092	4\$:	BLBC	16(P\$), 5\$	1578
			54	DD	00096		PUSHL	LBN	1580
7E	70	A2	54	C3	00098		SUBL3	LBN, 112(R2), -(SP)	
	0000V	CF	02	FB	0009D		CALLS	#2, MARK_BAD	
		52	04	AE	9E	5\$:	MOVAB	BUFFER+4, P	1583
		53	02	AE	9A		MOVZBL	BUFFER+2, R3	1584
		53	02	C6	000AA		DIVL2	#2, R3	
			53	D6	000AD		INCL	J	
			19	11	000AF		BRB	7\$	
		54	02	A2	3C	6\$:	MOVZWL	2(P), LBN	1586
54		08	10	62	F0		INSV	(P), #16, #8, LBN	1587
			54	DD	000BA		PUSHL	LBN	1588
		7E	01	A2	9A		MOVZBL	1(P), -(SP)	
			6E	D6	000C0		INCL	(SP)	
	0000V	CF	02	FB	000C2		CALLS	#2, MARK_BAD	
		52	04	C0	000C7		ADDL2	#4, P	1589
		E4	53	F5	000CA	7\$:	SOBGTR	J, 6\$	1584
			04	000CD			RET		1591

; Routine Size: 206 bytes. Routine Base: CODE + 0267

```

: 493 1592 1 %SBTTL 'MARK_BAD - mark blocks bad'
: 494 1593 1 ROUTINE MARK_BAD(BLOCK_COUNT,START_LBN): L_P$ NOVALUE=
: 495 1594 1
: 496 1595 1 !++
: 497 1596 1
: 498 1597 1 FUNCTIONAL DESCRIPTION:
: 499 1598 1 This routine enters the indicated block(s) into the bad block part
: 500 1599 1 of the allocation table. The table is maintained in reverse order
: 501 1600 1 by LBN, and adjacent or overlapping areas are merged.
: 502 1601 1
: 503 1602 1 INPUT PARAMETERS:
: 504 1603 1 BLOCK_COUNT - Count of blocks to mark bad
: 505 1604 1 START_LBN - Start LBN of blocks
: 506 1605 1
: 507 1606 1 IMPLICIT INPUTS:
: 508 1607 1 Impure area.
: 509 1608 1
: 510 1609 1 OUTPUT PARAMETERS:
: 511 1610 1 NONE
: 512 1611 1
: 513 1612 1 IMPLICIT OUTPUTS:
: 514 1613 1 Impure area.
: 515 1614 1
: 516 1615 1 ROUTINE VALUE:
: 517 1616 1 NONE
: 518 1617 1
: 519 1618 1 SIDE EFFECTS:
: 520 1619 1 NONE
: 521 1620 1
: 522 1621 1 !--
: 523 1622 1
: 524 1623 2 BEGIN
: 525 1624 2 LOCAL
: 526 1625 2 LBN, ! start LBN of new bad cluster
: 527 1626 2 COUNT, ! block count of new bad cluster
: 528 1627 2 J, ! index into bad block allocation table
: 529 1628 2 C; ! merge loop counter
: 530 1629 2 L_DECL;
: 531 1630 2
: 532 1631 2
: 533 1632 2 ! Check for table overflow.
: 534 1633 2
: 535 1634 2 IF .BADBLOCK_TOTAL GEQ BADBLOCK_MAX
: 536 1635 2 THEN
: 537 1636 2 BEGIN
: 538 1637 2 FILE_ERROR(BACKUP$_MAXBAD, ._BADBLOCK_FAB);
: 539 1638 2 RETURN;
: 540 1639 2 END;
: 541 1640 2
: 542 1641 2
: 543 1642 2 ! Round the specified LBN and count to the cluster boundaries.
: 544 1643 2
: 545 1644 2 LBN = .START_LBN / .CLUSTER * .CLUSTER;
: 546 1645 2 COUNT = (.START_LBN + .BLOCK_COUNT + .CLUSTER - 1) / .CLUSTER * .CLUSTER - .LBN;
: 547 1646 2
: 548 1647 2
: 549 1648 2 ! Search the allocation table until an entry is found with a start LBN lower

```

```

: 550 1649 2 ! than the new LBN. Shuffle the table down at this point and insert the
: 551 1650 2 ! new entry.
: 552 1651 2
: 553 1652 2 J = 0;
: 554 1653 2 UNTIL .J GEQ .BADBLOCK_TOTAL DO
: 555 1654 2 BEGIN
: 556 1655 3 IF .BADBLOCK_LBN[.J] LSSU .LBN THEN EXITLOOP;
: 557 1656 3 J = .J + 1;
: 558 1657 2 END;
: 559 1658 2
: 560 1659 2
: 561 1660 2 CH$MOVE ((.BADBLOCK_TOTAL-.J)*4, BADBLOCK_LBN[.J], BADBLOCK_LBN[.J+1]);
: 562 1661 2 CH$MOVE ((.BADBLOCK_TOTAL-.J)*4, BADBLOCK_CNT[.J], BADBLOCK_CNT[.J+1]);
: 563 1662 2 BADBLOCK_TOTAL = .BADBLOCK_TOTAL + 1;
: 564 1663 2 BADBLOCK_CNT[.J] = .COUNT;
: 565 1664 2 BADBLOCK_LBN[.J] = .LBN;
: 566 1665 2
: 567 1666 2
: 568 1667 2 ! Now check for adjacencies and merge if they exist. Start with the previous
: 569 1668 2 ! table entry and compare pairs.
: 570 1669 2
: 571 1670 2 IF .J NEQ 0 THEN J = .J - 1;
: 572 1671 2 C = 0;
: 573 1672 2
: 574 1673 2 UNTIL .J + 1 GEQ .BADBLOCK_TOTAL DO
: 575 1674 3 BEGIN
: 576 1675 3 IF .BADBLOCK_LBN[.J] LEQ .BADBLOCK_LBN[.J+1] + .BADBLOCK_CNT[.J+1]
: 577 1676 3 THEN
: 578 1677 4 BEGIN
: 579 1678 4 BADBLOCK_CNT[.J+1] = MAXU (.BADBLOCK_LBN[.J] + .BADBLOCK_CNT[.J],
: 580 1679 4 .BADBLOCK_LBN[.J+1] + .BADBLOCK_CNT[.J+1])
: 581 1680 4 - .BADBLOCK_LBN[.J+1];
: 582 1681 4 BADBLOCK_TOTAL = .BADBLOCK_TOTAL - 1;
: 583 1682 4 CH$MOVE ((.BADBLOCK_TOTAL-.J)*4, BADBLOCK_LBN[.J+1], BADBLOCK_LBN[.J]);
: 584 1683 4 CH$MOVE ((.BADBLOCK_TOTAL-.J)*4, BADBLOCK_CNT[.J+1], BADBLOCK_CNT[.J]);
: 585 1684 4 BADBLOCK_CNT[.BADBLOCK_TOTAL] = 0;
: 586 1685 4 END
: 587 1686 3 ELSE
: 588 1687 4 BEGIN
: 589 1688 4 J = .J + 1;
: 590 1689 4 C = .C + 1;
: 591 1690 4 IF .C GEQ 2 THEN EXITLOOP;
: 592 1691 3 END;
: 593 1692 2 END;
: 594 1693 1 END;

```

! end of merge loop  
! end of routine MARK\_BAD

07FC 0000 MARK_BAD:				.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10	
	SE	04	C2 00002	SUBL2	#4, SP	: 1593
		18	AB 9F 00005	PUSHAB	24(P\$)	: 1634
00000080	BF	00	BE D1 00008	CMPL	@0(SP), #128	:
			10 19 00010	BLSS	1\$	:
			6B DD 00012	PUSHL	(P\$)	: 1637

	00000000G	00	00000000G	8F	DD	00014	PUSHL	#BACKUP\$ MAXBAD		
				02	FB	0001A	CALLS	#2, FILE_ERROR		
					04	00021	RET		1636	
		51	0C	AB	D0	00022	1\$:	MOVL	12(P\$), R1	1644
50	08	AC		51	C7	00026		DIVL3	R1, START_LBN, R0	
58		50		51	C5	0002B		MULL3	R1, R0, LBN	
50	08	AC	04	AC	C1	0002F		ADDL3	BLOCK_COUNT, START_LBN, R0	1645
		50	FF	A140	9E	00035		MOVAB	-1(R1)[R0], R0	
		50		51	C6	0003A		DIVL2	R1, R0	
		50		51	C4	0003D		MULL2	R1, R0	
59		50		58	C3	00040		SUBL3	LBN, R0, COUNT	
				56	D4	00044		CLRL	J	1652
		5A	1C	AB	9E	00046		MOVAB	28(P\$), R10	1655
	00	BE		56	D1	0004A	2\$:	CMPL	J, a0(SP)	1653
				0A	18	0004E		BGEQ	3\$	
		58		6A46	D1	00050		CMPL	(R10)[J], LBN	1655
				04	1F	00054		BLSSU	3\$	
				56	D6	00056		INCL	J	1656
				F0	11	00058		BRB	2\$	1653
57	00	BE		56	C3	0005A	3\$:	SUBL3	J, a0(SP), R7	1660
		57		04	C4	0005F		MULL2	#4, R7	
			04	AA46	DF	00062		PUSHAL	4(R10)[J]	
				6A46	DF	00066		PUSHAL	(R10)[J]	
9E		9E		57	28	00069		MOV C3	R7, a(SP)+, a(SP)+	
			0220	CB46	DF	0006D		PUSHAL	54(P\$)[J]	1661
			021C	CB46	DF	00072		PUSHAL	540(P\$)[J]	
9E		9E		57	28	00077		MOV C3	R7, a(SP)+, a(SP)+	
			00	BE	D6	0007B		INCL	a0(SP)	1662
	021C	CB46		59	D0	0007E		MOVL	COUNT, 540(P\$)[J]	1663
		6A46		58	D0	00084		MOVL	LBN, (R10)[J]	1664
				56	D5	00088		TSTL	J	1670
				02	13	0008A		BEQL	4\$	
				56	D7	0008C		DECL	J	
			04	AE	D4	0008E	4\$:	CLRL	C	1671
		50	01	A6	9E	00091	5\$:	MOVAB	1(R6), R0	1673
	00	BE		50	D1	00095		CMPL	R0, a0(SP)	
				59	18	00099		BGEQ	8\$	
		59	021C	CB40	DE	0009B		MOVAL	540(P\$)[R0], R9	1675
52		6A40		69	C1	000A1		ADDL3	(R9), (R10)[R0], R2	
		52		6A46	D1	000A6		CMPL	(R10)[J], R2	
				3D	14	000AA		BGTR	7\$	
51		6A46	021C	CB46	C1	000AC		ADDL3	540(P\$)[J], (R10)[J], R1	1678
		52		51	D1	000B4		CMPL	R1, R2	1679
				03	1E	000B7		BGEQU	6\$	
		51		52	D0	000B9		MOVL	R2, R1	
69		51		6A40	C3	000BC	6\$:	SUBL3	(R10)[R0], R1, (R9)	1680
			00	BE	D7	000C1		DECL	a0(SP)	1681
		57	00	BE	D0	000C4		MOVL	a0(SP), R7	1682
58		57		56	C3	000C8		SUBL3	J, R7, R8	
		58		04	C4	000CC		MULL2	#4, R8	
				6A46	DF	000CF		PUSHAL	(R10)[J]	
				6A40	DF	000D2		PUSHAL	(R10)[R0]	
9E		9E		58	28	000D5		MOV C3	R8, a(SP)+, a(SP)+	
			021C	CB46	DF	000D9		PUSHAL	540(P\$)[J]	1683
9E		69		58	28	000DE		MOV C3	R8, (R9), a(SP)+	
			021C	CB47	D4	000E2		CLRL	540(P\$)[R7]	1684
				A8	11	000E7		BRB	5\$	1675

BADBLOCK  
V04-000

Scan bad block information  
MARK\_BAD - mark blocks bad

F 7  
15-Sep-1984 23:42:03  
14-Sep-1984 11:53:47

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

Page 20  
(7)

	04	56	D6	000E9	7s:	INCL	J	
	04	AE	D6	000EB		INCL	C	
02		AE	D1	000EE		CMPL	C	#2
		9D	19	000F2		BLSS	5s	
		04	000F4	8s:		RET		

: 1688  
: 1689  
: 1690  
: 1693

; Routine Size: 245 bytes, Routine Base: CODE + 0335

BADBLOCK  
V04-000

Scan bad block information  
MARK\_BAD - mark blocks bad

G 7  
15-Sep-1984 23:42:03  
14-Sep-1984 11:53:47

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

Page 21  
(8)

: 596  
: 597  
: 1694 1 END  
: 1695 0 ELUDOM

PSECT SUMMARY

Name Bytes Attributes  
CODE 1066 NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	27	0	1000	00:02.2

COMMAND QUALIFIERS

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

: Size: 1066 code + 0 data bytes  
: Run Time: 00:29.2  
: Elapsed Time: 01:37.8  
: Lines/CPU Min: 3485  
: Lexemes/CPU-Min: 31118  
: Memory Used: 259 pages  
: Compilation Complete

0010 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

Grid of 100 terminal windows (10x10) showing various system utilities and data. Key visible titles include:

- BACKUPMSG LIS
- ANALYZE LIS
- BUFFERS LIS
- CREATEDIR LIS
- BADBLOCK LIS
- BACKUPCMD LIS
- COMMAND LIS

Each window displays a mix of text-based data, command prompts, and status information typical of a VAX/VMS environment.