

BBBBBBBBBBBB		AAAAAAAAAA		DDDDDDDDDD
BBBBBBBBBBBB		AAAAAAAAAA		DDDDDDDDDD
BBBBBBBBBBBB		AAAAAAAAAA		DDDDDDDDDD
BBB	BBB	AAA	AAA	DDD
BBB	BBB	AAA	AAA	DDD
BBB	BBB	AAA	AAA	DDD
BBB	BBB	AAA	AAA	DDD
BBB	BBB	AAA	AAA	DDD
BBB	BBB	AAA	AAA	DDD
BBBBBBBBBBBB		AAA	AAA	DDD
BBBBBBBBBBBB		AAA	AAA	DDD
BBBBBBBBBBBB		AAA	AAA	DDD
BBB	BBB	AAAAAAAAAAAA		DDD
BBB	BBB	AAAAAAAAAAAA		DDD
BBB	BBB	AAAAAAAAAAAA		DDD
BBB	BBB	AAA	AAA	DDD
BBB	BBB	AAA	AAA	DDD
BBB	BBB	AAA	AAA	DDD
BBB	BBB	AAA	AAA	DDD
BBBBBBBBBBBB		AAA	AAA	DDDDDDDDDD
BBBBBBBBBBBB		AAA	AAA	DDDDDDDDDD
BBBBBBBBBBBB		AAA	AAA	DDDDDDDDDD

```
BBBBBBBBB      AAAAAA      DDDDDDDD      MM      MM      AAAAAA      IIIIII      NN      NN
BBBBBBBBB      AAAAAA      DDDDDDDD      MM      MM      AAAAAA      IIIIII      NN      NN
BB      BB      AA      AA      DD      DD      MMMM      MMMM      AA      AA      II      II      NN      NN
BB      BB      AA      AA      DD      DD      MMMM      MMMM      AA      AA      II      II      NN      NN
BB      BB      AA      AA      DD      DD      MM      MM      AA      AA      II      II      NNNN      NN
BBBBBBBBB      AA      AA      DD      DD      MM      MM      AA      AA      II      II      NNNN      NN
BBBBBBBBB      AA      AA      DD      DD      MM      MM      AA      AA      II      II      NN      NN
BB      BB      AAAAAAAAAA      DD      DD      MM      MM      AAAAAAAAAA      II      II      NN      NNNN
BB      BB      AAAAAAAAAA      DD      DD      MM      MM      AAAAAAAAAA      II      II      NN      NNNN
BB      BB      AA      AA      DD      DD      MM      MM      AA      AA      II      II      NN      NN
BB      BB      AA      AA      DD      DD      MM      MM      AA      AA      II      II      NN      NN
BBBBBBBBB      AA      AA      DDDDDDDD      MM      MM      AA      AA      IIIIII      NN      NN
BBBBBBBBB      AA      AA      DDDDDDDD      MM      MM      AA      AA      IIIIII      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
LLLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLLL      IIIIII      SSSSSSSS
```



```
1 0001 0 MODULE badmain ( %TITLE 'Analyze/Media Main Module'
2 0002 0 IDENT = 'V04-000',
3 0003 0 MAIN = bad$main) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *****
27 0027 1
28 0028 1
29 0029 1 ++
30 0030 1
31 0031 1 Facility:
32 0032 1
33 0033 1 Analyze/Media
34 0034 1
35 0035 1 Abstract:
36 0036 1
37 0037 1 This module is the main routine or driver for the facility.
38 0038 1
39 0039 1 Environment:
40 0040 1
41 0041 1 VAX/VMS User Mode, Non-Privileged
42 0042 1
43 0043 1 Author:
44 0044 1
45 0045 1 Michael T. Rhodes, CREATION DATE: July, 1982
46 0046 1
47 0047 1 Modified By:
48 0048 1
49 0049 1 V03-005 MTR0009 Michael T. Rhodes 3-Jul-1984
50 0050 1 Don't produce a listing of non-last track SDBSFs
51 0051 1 if they don't exist and we aren't exercising the medium.
52 0052 1
53 0053 1 V03-004 BLS0244 Benn Schreiber 26-Oct-1983
54 0054 1 Exit with warning severity if bad blocks found.
55 0055 1
56 0056 1 V03-003 MTR0004 Michael T. Rhodes 2-Mar-1983
57 0057 1 Move routines 'bad$retry' and 'bad$verify_blocks' to
```

BADMAIN
V04-000

Analyze/Media Main Module

H 7
15-Sep-1984 23:41:46
14-Sep-1984 11:54:24

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[BAD.SRC]BADMAIN.B32;1 Page 2
(1)

```

: 58      0058 1  | module BAD10.B32.
: 59      0059 1  |
: 60      0060 1  | V03-002 MTR0003      Michael T. Rhodes      11-Feb-1982
: 61      0061 1  |      Add support for Stand Alone BAD.
: 62      0062 1  |
: 63      0063 1  | V03-001 MTR0001      Michael T. Rhodes      15-Dec-1982
: 64      0064 1  |      Add the /LOG qualifier, which is processed by the ROUTINE
: 65      0065 1  |      'total_bad_blocks', in addition to code in 'bad$main'.
: 66      0066 1  |
: 67      0067 1  |      Add code to bad$main to replace the "worst case test pattern"
: 68      0068 1  |      with the user supplied test pattern obtained via:
: 69      0069 1  |      /EXERCISE=PATTERN=(12345678 [,%077777777,%D99999999,%XABCDEF])
: 70      0070 1  |
: 71      0071 1  |      Modify the retry logic to mark and step past bad blocks.
: 72      0072 1  |      (NOTE: This code is RECURSIVE.)
: 73      0073 1  |
: 74      0074 1  | --

```



```

76 0075 1 %SBTTL 'Declarations'
77 0076 1
78 0077 1 Include Files:
79 0078 1
80 0079 1 REQUIRE 'lib$:baddef';
81 0199 1 LIBRARY 'SYSS$LIBRARY:LIB';
82 0200 1
83 0201 1
84 0202 1 Table of Contents:
85 0203 1
86 0204 1 FORWARD ROUTINE
87 0205 1     bad$main : NOVALUE,
88 0206 1     finish_medium : NOVALUE,
89 0207 1     read_medium : NOVALUE,
90 0208 1     total_bad_blocks,
91 0209 1     write_last_track : NOVALUE,
92 0210 1     write_medium : NOVALUE,
93 0211 1     write_new_sdbsf : NOVALUE,
94 0212 1     write_non_lsttrk : NOVALUE;
95 0213 1
96 0214 1
97 0215 1 External References:
98 0216 1
99 0217 1 EXTERNAL ROUTINE
100 0218 1     bad$close_files : ADDRESSING_MODE (GENERAL),
101 0219 1     bad$cvr_nlt_long : ADDRESSING_MODE (GENERAL),
102 0220 1     bad$handler : WEAK,
103 0221 1     bad$init : ADDRESSING_MODE (GENERAL),
104 0222 1     bad$init_buffers : ADDRESSING_MODE (GENERAL),
105 0223 1     bad$produce_report : ADDRESSING_MODE (GENERAL),
106 0224 1     bad$retry : ADDRESSING_MODE (GENERAL),
107 0225 1     bad$start_io : ADDRESSING_MODE (GENERAL),
108 0226 1     bad$sync_io : ADDRESSING_MODE (GENERAL),
109 0227 1     bad$verify_blocks : ADDRESSING_MODE (GENERAL),
110 0228 1     CHECKSUM2 : ADDRESSING_MODE (GENERAL);
111 0229 1
112 0230 1 EXTERNAL
113 0231 1     bad$gb_block_fact : BYTE,
114 0232 1     bad$ga_bufadr : VECTOR [2, LONG],
115 0233 1     bad$gl_context : BITVECTOR [32],
116 0234 1     bad$gl_cylinders,
117 0235 1     bad$ga_device : $BBLOCK [dsc$c_s_bln],
118 0236 1     bad$gl_devtype,
119 0237 1     bad$gl_func,
120 0238 1     bad$gq_iosb : VECTOR [4, LONG],
121 0239 1     bad$gl_maxblock,
122 0240 1     bad$ga_mdbsf,
123 0241 1     bad$gl_mdbsf_ptr,
124 0242 1     bad$gl_pagcnt,
125 0243 1     bad$ga_sdbsf,
126 0244 1     bad$gl_sdbsf_ptr,
127 0245 1     bad$gl_sectors,
128 0246 1     bad$gl_sector_siz,
129 0247 1     bad$gl_status : $BBLOCK,
130 0248 1     bad$gl_tracks,
131 0249 1     bad$gl_trnsfr_cnt;
132 0250 1
```

Define BAD's structures etc.
Define VMS structures etc.

Main procedure declaration.
Write the updated SDBSF and produce a report (if r
Read blocks back from medium.
Compute the total number of bad blocks on the devi
Write the SDBSF to the last track.
Write test pattern onto medium.
Perform house cleaning on SDBSF before writing it.
Write the SDBSF to the last good block on the medi

Close all open files.
Convert a non last track SDBSF entry to longwords.
Condition Handler for Stand Alone Environment.
Routines for allocation/initialization of structur
Routine to fill all of the IO buffers with the tes
Generate the report.
Mark bad blocks, continuing the IO operation in pr
Start asynchronous IO operations.
Perform synchronous IO.
Verify that the pattern read, matches the test pat
Perform and generate a checksum on the given buffe

Blocking factor (number of sectors per block).
Data buffer address vector.
Context bits.
Number of cylinders.
Device name descriptor.
Device type identifier.
IO function code.
IO status blocks.
Largest logical block number on device.
Address if the MDDBSF.
Starting block number of the last track.
Number of 512 byte pages per transfer.
Address of the SDBSF.
Address of the next free entry in the SDBSF.
Number of sectors per track.
Size in bytes of the sector.
Global status.
Number of tracks per cylinder.
Number of bytes per transfer.

BADMAIN
V04-000

Analyze/Media Main Module
Declarations

J 7
15-Sep-1984 23:41:46
14-Sep-1984 11:54:24

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[BAD.SRC]BADMAIN.B32;1 Page 4
(2)

```

: 133      0251 1  !
: 134      0252 1  ! Define message codes...
: 135      0253 1  !
: 136      0254 1  EXTERNAL LITERAL
: 137      0255 1      bad$_bbfwt,
: 138      0256 1      bad$_blk0bad,
: 139      0257 1      bad$_facility,
: 140      0258 1      bad$_lsttotbk,
: 141      0259 1      bad$_writeerr;
: 142      0260 1  !
: 143      0261 1  !
: 144      0262 1  ! Private Storage:
: 145      0263 1  !
: 146      0264 1  OWN
: 147      0265 1      blkstrk,
: 148      0266 1      first_pattern : BYTE,
: 149      0267 1      lastfull,
: 150      0268 1      remainder,
: 151      0269 1      test_pattern : VECTOR [3, LONG]
: 152      0270 1      INITIAL (0, -1, %X'DB6DB6DB'),
: 153      0271 1      usable;
: 154      0272 1  !
: 155      0273 1  LITERAL
: 156      0274 1      last_pattern = 2;
: 157      0275 1  !

! Failed to write the SDBSF to media.
! Block 0 is bad, do not use media as a system device
! BAD's facility number.
! Total bad blocks detected.
! Error occurred while trying to write to a device.

! Blocks per track.
! Starting pattern number.
! Last LBN for full buffer transfer.
! Remainder of device to read in bytes.
! Test patterns.

! Total number of usable (writable) blocks.

! Number of test patterns -1.
```



```
159 0276 1 %SBTTL 'bad$main'
160 0277 1 GLOBAL ROUTINE bad$main : NOVALUE =
161 0278 1 ++
162 0279 1
163 0280 1 Functional Description:
164 0281 1
165 0282 1 The driver is responsible for calling routines which perform such
166 0283 1 tasks as allocating and initializing storage and control structures,
167 0284 1 parsing the command line (which also involves manipulation of the
168 0285 1 structures), analyzing the media, and concluding with the release
169 0286 1 of allocated resources (and optionally generating a listing).
170 0287 1
171 0288 1 Implicit Outputs:
172 0289 1
173 0290 1 The media has been analyzed, and if requested a listing will have
174 0291 1 been produced.
175 0292 1
176 0293 1 Completion Codes:
177 0294 1
178 0295 1 BAD-S-NORMAL Normal successfull completion.
179 0296 1
180 0297 1 BAD-W-BLKOBAD Block zero on the device is bad,
181 0298 1 do not use this medium as a system device.
182 0299 1
183 0300 1
184 0301 1 SIDE EFFECTS:
185 0302 1
186 0303 1 Completion codes with severity worse than those listed above will be the
187 0304 1 result of a fatal error that was signalled via a SIGNAL_STOP, or one in
188 0305 1 which we had manually boosted the severity to SEVERE.
189 0306 1
190 0307 1 The success/warnings listed above will have the "inhibit" bit set, to
191 0308 1 prevent noisy messages from being displayed (or displayed twice as the
192 0309 1 case may be).
193 0310 1
194 0311 1 --
195 0312 2 BEGIN
196 0313 2 BUILTIN
197 0314 2 FP;
198 0315 2
199 0316 2 LOCAL
200 0317 2 bad_blocks_found;
201 0318 2
202 0319 2 IF bad$handler NEQ 0
203 0320 2 THEN .FP = bad$handler;
204 0321 2
205 0322 2 WHILE true DO
206 0323 3 BEGIN
207 0324 3
208 0325 3 bad$init ();
209 0326 3
210 0327 3 IF .bad$gl_context [ctx_v_exercise]
211 0328 3 THEN
212 0329 4 BEGIN
213 0330 4 IF .bad$gl_context [ctx_v_ltdevice]
214 0331 4 THEN
215 0332 4 usable = .bad$gl_mdbsf_ptr
```

! Are we executing Stand Alone or under VMS?
! Stand Alone -- Establish Condition Handler.

! Continuously process devices when Stand Alone, how
! under VMS we will perform a normal exit w/status.

! Allocate and initialize storage and control struct

! Are we going to analyze the medium?
! Yes...

! Select the number of usable blocks according to th
! size of the device (as specified by the last track
! device bit in the context flags). Note also, that

```
216 0333 4 ELSE ! on last track devices the last track is protected
217 0334 4 usable = .bad$gl_maxblock; ! regardless of the setting of the "override" bit.
218 0335 4
219 0336 5 lastfull = (.usable / .bad$gl_pagcnt - 1) ! Block number of last full buffer transfer.
220 0337 4 * .bad$gl_pagcnt;
221 0338 5 remainder = (.usable - (.lastfull + .bad$gl_pagcnt))! Bytes remaining to transfer.
222 0339 4 * bad$sk_page_size;
223 0340 4
224 0341 4 IF .bad$gl_context [ctx_v_exercise_full] ! How extensively should the surface be tested?
225 0342 4 THEN first_pattern = 0 ! Use all three test patterns.
226 0343 4 ELSE first_pattern = 2; ! Only use the "worst case" test pattern.
227 0344 4
228 0345 4 INCR pattern FROM .first_pattern TO last_pattern DO ! Note that in all cases the "worst case" test p
229 0346 5 BEGIN ! is left on the medium.
230 0347 5 IF .bad$gl_context [ctx_v_pattern] ! If the user supplied a test pattern,
231 0348 5 AND .pattern EQL last_pattern ! and its time to use it, then initialize
232 0349 5 THEN bad$init_buffers ( ! the buffers with it (upto 128 bits of pattern).
233 0350 5 ELSE bad$init_buffers (.test_pattern [.pattern]); ! Otherwise, initialize them with the default
234 0351 5 write_medium (); ! Write the test pattern onto the medium.
235 0352 5 read_medium (); ! Read and Verify that the patterns match.
236 0353 4 END;
237 0354 4 END; ! End of Analysis...
238 0355 4
239 0356 4 finish_medium (); ! Write updated SDBSF to medium produce a listing
240 0357 4
241 0358 4 bad_blocks_found = total_bad_blocks();
242 0359 4 IF .bad$gl_context [ctx_v_log] ! Should we display the total bad block count?
243 0360 4 THEN
244 0361 4 SIGNAL (bad$_lstattotbk, 3, bad$ga_device, .bad$gl_maxblock, .bad_blocks_found);
245 0362 4
246 0363 4 IF .bad$gl_context [ctx_v_blk0bad]
247 0364 4 THEN
248 0365 4 BEGIN ! If block 0 (zero) is bad, then signal the warning
249 0366 4 bad$gl_status = bad$_blk0bad; ! and exit with warning as our status.
250 0367 4 SIGNAL (bad$_blk0bad, 1, bad$ga_device);
251 0368 4 END
252 0369 4 ELSE ! Otherwise, if we have reached here exit with succe
253 0370 4 IF .bad_blocks_found EQL 0 ! if no bad blocks found, else exit with a warning.
254 0371 4 THEN bad$gl_status = sts$normal ! This is due to the fact that any serious errors wo
255 0372 4 ELSE bad$gl_status = sts$warning; ! have caused us to exit via a SIGNAL_STOP which wou
256 0373 4 ! result in a severity of FATAL in the resultant sta
257 0374 4
258 0375 4 bad$gl_status [sts$v_fac_no] = bad$ facility; ! Stuff the facility number into the status.
259 0376 4 bad$gl_status [sts$v_inhib_msg] = TRUE; ! Do not reprint the message.
260 0377 4
261 0378 4 IF bad$handler EQL 0 ! Are we executing under VMS?
262 0379 4 THEN $EXIT (CODE = .bad$gl_status); ! Yes, perform normal exit.
263 0380 4
264 0381 2 END; ! of WHILE true DO BEGIN ! No, go back for the next command and continue.
265 0382 2
266 0383 1 END; ! of GLOBAL ROUTINE bad$main.
```

```
.TITLE BADMAIN Analyze/Media Main Module
.IDENT \V04-000\
.PSECT $OWNS$,NOEXE,2
```


00000 BLKSTRK:.BLKB 4
00004 FIRST_PATTERN:
 .BLKB 1
00005 .BLKB 3
00008 LASTFULL:
 .BLKB 4
0000C REMAINDER:
 .BLKB 4
DB6DB6DB FFFFFFFF 00000000 00010 TEST_PATTERN:
 .LONG 0, -1, -613566757
0001C USABLE: .BLKB 4

.EXTRN BAD\$CLOSE_FILES
.EXTRN BAD\$CVT_NLT_LONG
.EXTRN BAD\$INIT, BAD\$INIT_BUFFERS
.EXTRN BAD\$PRODUCE_REPORT
.EXTRN BAD\$RETRY, BAD\$START_IO
.EXTRN BAD\$SYNC_IO, BAD\$VERIFY_BLOCKS
.EXTRN CHECKSUM2, BAD\$GB_BLOCK_FACT
.EXTRN BAD\$GA_BUFADR, BAD\$GL_CONTEXT
.EXTRN BAD\$GL_CYLINDERS
.EXTRN BAD\$GA_DEVICE, BAD\$GL_DEVTYPE
.EXTRN BAD\$GL_FUNC, BAD\$GQ_IOSB
.EXTRN BAD\$GL_MAXBLOCK
.EXTRN BAD\$GA_MDBSF, BAD\$GL_MDBSF_PTR
.EXTRN BAD\$GL_PAGCNT, BAD\$GA_SDBSF
.EXTRN BAD\$GL_SDBSF_PTR
.EXTRN BAD\$GL_SECTORS, BAD\$GL_SECTOR_SIZ
.EXTRN BAD\$GL_STATUS, BAD\$GL_TRACKS
.EXTRN BAD\$GL_TRNSFR_CNT
.EXTRN BAD\$BBFWRT, BAD\$BLKOBAD
.EXTRN BAD\$FACILITY, BAD\$LSTTOTBK
.EXTRN BAD\$WRITEERR, SYS\$EXIT
.WEAK BAD\$HANDLER

.PSECT \$CODE\$,NOWRT,2

		07FC	00000	.ENTRY	BAD\$MAIN, Save R2,R3,R4,R5,R6,R7,R8,R9,R10	: 0277
	5A	0000G	CF 9E 00002	MOVAB	BAD\$HANDLER, R10	:
	59	00000000G	8F D0 00007	MOVL	#BAD\$BLKOBAD, R9	:
	58	00000000G	00 9E 0000E	MOVAB	LIB\$SIGNAL, R8	:
	57	00000000G	00 9E 00015	MOVAB	BAD\$INIT_BUFFERS, R7	:
	56	0000G	CF 9E 0001C	MOVAB	BAD\$GL_STATUS, R6	:
	55	0000G	CF 9E 00021	MOVAB	BAD\$GL_CONTEXT, R5	:
	54	0000'	CF 9E 00026	MOVAB	USABLE, R4	:
	50		6A 9E 0002B	MOVAB	BAD\$HANDLER, R0	: 0319
			03 13 0002E	BEQL	1\$	:
	6D		6A 9E 00030	MOVAB	BAD\$HANDLER, (FP)	: 0320
66	00000000G	00	00 FB 00033 1\$:	CALLS	#0, BAD\$INIT	: 0325
	65		02 E1 0003A	BBC	#2, BAD\$GL_CONTEXT, 10\$	: 0327
	07	01	A5 E9 0003E	BLBC	BAD\$GL_CONTEXT+1, 2\$	: 0330
	64	0000G	CF D0 00042	MOVL	BAD\$GL_MDBSF_PTR, USABLE	: 0332
			05 11 00047	BRB	3\$	:
	64	0000G	CF D0 00049 2\$:	MOVL	BAD\$GL_MAXBLOCK, USABLE	: 0334
	50	0000G	CF D0 0004E 3\$:	MOVL	BAD\$GL_PAGCNT, R0	: 0336
51	64		50 C7 00053	DIVL3	R0, USABLE, R1	:

EC	A4	51	51	D7	00057	DECL	R1		
		50	50	C5	00059	MULL3	R0, R1, LASTFULL		0337
		64	EC	A4	C0 0005E	ADDL2	LASTFULL, R0		0338
FO	50	50	50	C3	00062	SUBL3	R0, USABLE, R0		
	A4	50	09	78	00066	ASHL	#9, R0, REMAINDER		0339
	05	65	03	E1	0006B	BBC	#3, BAD\$GL_CONTEXT, 4\$		0341
			E8	A4	94 0006F	CLRB	FIRST_PATTERN		0342
				04	11 00072	BRB	5\$		
	E8	A4	02	90	00074 4\$:	MOVB	#2, FIRST_PATTERN		0343
		52	E8	A4	9A 00078 5\$:	MOVZBL	FIRST_PATTERN, PATTERN		0345
				52	D7 0007C	DECL	PATTERN		
				20	11 0007E	BRB	9\$		
OA	01	A5	02	E1	00080 6\$:	BBC	#2, BAD\$GL_CONTEXT+1, 7\$		0347
		02	52	D1	00085	CMPL	PATTERN, #2		0348
		67	05	12	00088	BNEQ	7\$		
			00	FB	0008A	CALLS	#0, BAD\$INIT_BUFFERS		0349
			07	11	0008D	BRB	8\$		
		67	F4 A4	42	DD 0008F 7\$:	PUSHL	TEST_PATTERN[PATTERN]		0350
				01	FB 00093	CALLS	#1, BAD\$INIT_BUFFERS		
	0000V	CF	00	FB	00096 8\$:	CALLS	#0, WRITE_MEDIUM		0351
	0000V	CF	00	FB	0009B	CALLS	#0, READ_MEDIUM		0352
DC		52	02	F3	000A0 9\$:	AOBLEQ	#2, PATTERN, 6\$		0345
	0000V	CF	00	FB	000A4 10\$:	CALLS	#0, FINISH_MEDIUM		0356
	0000V	CF	00	FB	000A9	CALLS	#0, TOTAL_BAD_BLOCKS		0358
		53	50	D0	000AE	MOVL	R0, BAD_BLOCKS_FOUND		
			65	95	000B1	TSTB	BAD\$GL_CONTEXT		0359
			15	18	000B3	BGEQ	11\$		
			53	DD	000B5	PUSHL	BAD_BLOCKS_FOUND		0361
	0000G		CF	DD	000B7	PUSHL	BAD\$GL_MAXBLOCK		
	0000G		CF	9F	000BB	PUSHAB	BAD\$GA_DEVICE		
			03	DD	000BF	PUSHL	#3		
		68	00000000G	8F	DD 000C1	PUSHL	#BAD\$_LSTTOTBK		
		65		05	FB 000C7	CALLS	#5, LIB\$SIGNAL		
10		66		01	E1 000CA 11\$:	BBC	#1, BAD\$GL_CONTEXT, 12\$		0363
				59	D0 000CE	MOVL	R9, BAD\$GL_STATUS		0366
			0000G	CF	9F 000D1	PUSHAB	BAD\$GA_DEVICE		0367
				01	DD 000D5	PUSHL	#1		
				59	DD 000D7	PUSHL	R9		
		68		03	FB 000D9	CALLS	#3, LIB\$SIGNAL		0363
				0B	11 000DC	BRB	14\$		0370
				53	D5 000DE 12\$:	TSTL	BAD_BLOCKS_FOUND		
		66		05	12 000E0	BNEQ	13\$		
				01	D0 000E2	MOVL	#1, BAD\$GL_STATUS		0371
				02	11 000E5	BRB	14\$		
				66	D4 000E7 13\$:	CLRL	BAD\$GL_STATUS		0372
02	A6		0C	8F	F0 000E9 14\$:	INSV	#BAD\$_FACILITY, #0, #12, BAD\$GL_STATUS+2		0375
			03	10	88 000F3	BISB2	#16, BAD\$GL_STATUS+3		0376
				6A	9E 000F7	MOVAB	BAD\$HANDLER, R0		0378
				09	12 000FA	BNEQ	15\$		
				66	DD 000FC	PUSHL	BAD\$GL_STATUS		0379
	00000000G	00		01	FB 000FE	CALLS	#1, SYS\$EXIT		
			FF2B	31	00105 15\$:	BRW	1\$		0322

; Routine Size: 264 bytes, Routine Base: \$CODE\$ + 0000

; 267 0384 1

: 295 0411 1


```
297 0412 1 %SBTTL 'read_medium -- Read and Verify the blocks against the test pattern'
298 0413 1 ROUTINE read_medium : NOVALUE =
299 0414 1 ++
300 0415 1
301 0416 1 Functional Description:
302 0417 1
303 0418 1 Read and Verify the storage blocks from the medium against the test pattern
304 0419 1 buffer.
305 0420 1
306 0421 1 Side Effects:
307 0422 1
308 0423 1 Any mismatched data pages will be retired to the Software Detected Bad Sector File.
309 0424 1
310 0425 1 Special handling for the TU58 has been added. This simply adds the function
311 0426 1 modifier IOSM_DATACHECK, which increases the read threshold resulting in
312 0427 1 weak spots or drop outs to become visible.
313 0428 1
314 0429 1 --
315 0430 2 BEGIN
316 0431 2
317 0432 2 bad$gl_func = IOS_READLBLK;
318 0433 2
319 0434 2 IF .bad$gl_devtype EQL dt$_tu58 ! If the device is a TU-58, read
320 0435 2 THEN ! data back with increased threshold
321 0436 2 bad$gl_func = .bad$gl_func OR IOSM_DATACHECK; ! (forces weak spots to show up).
322 0437 2
323 0438 2 bad$start_io (.lastfull); ! Read device asynchronously.
324 0439 2
325 0440 2 IF NOT bad$sync_io (.bad$ga_bufadr [0], .remainder, .lastfull + .bad$gl_pagcnt)
326 0441 2 THEN
327 0442 2 bad$retry (.lastfull + .bad$gl_pagcnt, .remainder, bad$gq_iosb [0], .bad$ga_bufadr [0]);
328 0443 2
329 0444 2 bad$verify_blocks (.bad$ga_bufadr [0], .remainder, .lastfull + .bad$gl_pagcnt, bad$gq_iosb [0]);
330 0445 2
331 0446 1 END; ! of ROUTINE read_medium
```

```
001C 00000 READ_MEDIUM:
54 0000G CF 9E 00002 .WORD Save R2,R3,R4 0413
53 0000G CF 9E 00007 MOVAB BAD$GA_BUFADR, R4
52 0000' CF 9E 0000C MOVAB BAD$GL_PAGCNT, R3
0000G CF 21 D0 00011 MOVAB LASTFUL, R2
OE 0000G CF D1 00016 MOVL #33, BAD$GL_FUNC 0432
06 12 0001B CMPL BAD$GL_DEVTYPE, #14 0434
0000G CF 40 8F 88 0001D BNEQ 1$
62 DD 00023 1$: BISB2 #64, BAD$GL_FUNC+1 0436
00000000G 00 01 FB 00025 PUSHL LASTFULL 0438
7E 62 63 C1 0002C CALLS #1, BAD$START_IO
04 A2 DD 00030 ADDL3 BAD$GL_PAGCNT, LASTFULL, -(SP) 0440
64 DD 00033 PUSHL REMAINDER
00000000G 00 03 FB 00035 PUSHL BAD$GA_BUFADR
14 50 E8 0003C CALLS #3, BAD$SYNC_IO
64 DD 0003F BLBS R0, 2$
PUSHL BAD$GA_BUFADR 0442
```


			0000G	CF	9F	00041	
			04	A2	DD	00045	
7E		62		63	C1	00048	
	00000000G	00		04	FB	0004C	
			0000G	CF	9F	00053	2\$:
7E		62		63	C1	00057	
			04	A2	DD	0005B	
				64	DD	0005E	
	00000000G	00		04	FB	00060	
					04	00067	

```

PUSHAB    BAD$GQ_IOSB
PUSHL     REMAINDER
ADDL3     BAD$GL_PAGCNT, LASTFULL, -(SP)
CALLS     #4, BAD$RETRY
PUSHAB    BAD$GQ_IOSB
ADDL3     BAD$GL_PAGCNT, LASTFULL, -(SP)
PUSHL     REMAINDER
PUSHL     BAD$GA_BUFADR
CALLS     #4, BAD$VERIFY_BLOCKS
RET

```

0444

0446

; Routine Size: 104 bytes, Routine Base: \$CODE\$ + 014D

: 332 0447 1

```
0448 1 %SBTTL 'finish_medium -- Perform completion activities'
0449 1 ROUTINE finish_medium : NOVALUE =
0450 1 ++
0451 1
0452 1 Functional Description:
0453 1
0454 1 Write the Software Detected Bad Sector File to the medium if it was
0455 1 modified, produce a listing if one was requested and close the output
0456 1 file(s).
0457 1
0458 1 --
0459 2 BEGIN
0460 2 IF .bad$gl_context [ctx_v_exercise] OR .bad$gl_context [ctx_v_badblocks]
0461 2 THEN
0462 2 write_new_sdbsf ();
0463 2
0464 2 IF .bad$gl_context [ctx_v_show_after]
0465 2 THEN
0466 2 IF .bad$gl_context [ctx_v_ltdevice] OR .bad$gl_context [ctx_v_nlt_sdbsf]
0467 2 THEN bad$produce_report ();
0468 2
0469 2 IF .bad$gl_context [ctx_v_output]
0470 2 THEN
0471 2 bad$close_files ();
0472 2
0473 1 END; ! of ROUTINE finish_medium
```

0004 00000 FINISH_MEDIUM:									
		52	0000G	CF	9E	00002		.WORD	Save R2
03		62		02	E0	00007		MOVAB	BAD\$GL_CONTEXT, R2
		05		62	E9	0000B		BBS	#2, BAD\$GL_CONTEXT, 1\$
	0000V	CF		00	FB	0000E	1\$:	BLBC	BAD\$GL_CONTEXT, 2\$
10	01	A2		05	E1	00013	2\$:	CALLS	#0, WRITE NEW SDBSF
		05	01	A2	E8	00018		BBC	#5, BAD\$GL_CONTEXT+1, 4\$
			01	A2	95	0001C		BLBS	BAD\$GL_CONTEXT+1, 3\$
				07	18	0001F		TSTB	BAD\$GL_CONTEXT+1
	00000000G	00		00	FB	00021	3\$:	BGEQ	4\$
07	01	A2		01	E1	00028	4\$:	CALLS	#0, BAD\$PRODUCE REPORT
	00000000G	00		00	FB	0002D		BBC	#1, BAD\$GL_CONTEXT+1, 5\$
				04	00034	5\$:		CALLS	#0, BAD\$CLOSE_FILES
								RET	

; Routine Size: 53 bytes, Routine Base: \$CODE\$ + 01B5

; 360 0474 1


```
0475 1 %SBTTL 'total_bad_blocks -- Compute the total number of bad blocks'
0476 1 ROUTINE total_bad_blocks =
0477 1 ++
0478 1
0479 1 Functional Description:
0480 1
0481 1 Compute the total number of bad blocks which have been detected by
0482 1 either the Manufacturer or BAD and return it to the caller.
0483 1
0484 1 Implicit Inputs:
0485 1
0486 1 bad$gl_context Context flags to determine [non] last track device.
0487 1
0488 1 bad$ga_mdbsf Manufacturers Detected Bad Sector File buffer.
0489 1
0490 1 bad$ga_sdbsf Software Detected Bad Sector File buffer.
0491 1
0492 1 Routine Value:
0493 1
0494 1 The total number of bad blocks recorded in both the MDBSF and SDBSF,
0495 1 this includes any blocks which were entered manually by the user via
0496 1 the /BAD_BLOCKS qualifier.
0497 1
0498 1 --
0499 2 BEGIN
0500 2 LOCAL
0501 2 count,
0502 2 lbn,
0503 2 total : INITIAL (0);
0504 2
0505 2 IF .bad$gl_context [ctx_v_ltdevice]
0506 2 THEN
0507 2 BEGIN
0508 2 INCR entry FROM .bad$ga_mdbsf + ltk_k_headersiz TO .bad$ga_mdbsf + bad$k_page_size - 4 BY 4
0509 2 DO IF ..entry NEQ -1
0510 2 THEN total = .total + 1
0511 2 ELSE EXITLOOP;
0512 2 total = .total + ((.bad$gl_sdbsf_ptr - .bad$ga_sdbsf - ltk_k_headersiz) / 4);
0513 2 END
0514 2 ELSE
0515 2 BEGIN
0516 2 INCR entry FROM .bad$ga_sdbsf + nlt_k_headersiz TO .bad$gl_sdbsf_ptr - 4 BY 4 DO
0517 2 BEGIN
0518 2 bad$cvl_nlt_long (..entry, count, lbn);
0519 2 total = .total + .count + 1;
0520 2 END;
0521 2 END;
0522 2
0523 2 RETURN .total;
0524 2
0525 1 END; ! of ROUTINE total_bad_blocks
```

001C 00000 TOTAL_BAD_BLOCKS:

BADMAIN
V04-000

Analyze/Media Main Module

total_bad_blocks -- Compute the total number of

15-Sep-1984 23:41:46
14-Sep-1984 11:54:24

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[BAD.SRC]BADMAIN.B32;1

Page 14
(7)

		5E		08	C2	00002		.WORD	Save R2,R3,R4		0476
				52	D4	00005		SUBL2	#8, SP		
		36	0000G	CF	E9	00007		CLRL	TOTAL		0499
51	0000G	CF	000001FC	8F	C1	0000C		BLBC	BAD\$GL_CONTEXT+1, 4\$		0505
50	0000G	CF		04	C1	00016		ADDL3	#508, BAD\$GA_MDBSF, R1		0508
				0B	11	0001C		ADDL3	#4, BAD\$GA_MDBSF, ENTRY		
	FFFFFFF	8F		60	D1	0001E	1\$:	BRB	2\$		
				08	13	00025		CML	(ENTRY), #-1		0509
				52	D6	00027		BEQL	3\$		
FFEF	50			51	F1	00029	2\$:	INCL	TOTAL		0510
	50	0000G		CF	C3	0002F	3\$:	ACBL	R1, #4, ENTRY, 1\$		0509
				08	C2	00037		SUBL3	BAD\$GA_SDBSF, BAD\$GL_SDBSF_PTR, R0		0512
				50				SUBL2	#8, R0		
				50	C6	0003A		DIVL2	#4, R0		
				52	50	0003D		ADDL2	R0, TOTAL		
				2A	11	00040		BRB	7\$		0505
	54	0000G		04	C3	00042	4\$:	SUBL3	#4, BAD\$GL_SDBSF_PTR, R4		0516
				CF	D0	00048		MOVL	BAD\$GA_SDBSF, ENTRY		
				17	11	0004D		BRB	6\$		
				5E	DD	0004F	5\$:	PUSHL	SP		0518
				08	AE	9F		PUSHAB	COUNT		
				63	DD	00054		PUSHL	(ENTRY)		
				03	FB	00056		CALLS	#3, BAD\$CVT_NLT_LONG		
	50	00000000G		52	AE	C1	0005D	ADDL3	COUNT, TOTAL, R0		0519
				52	01	A0	9E	MOVAB	1(R0), TOTAL		
FFE3	53			54	F1	00066	6\$:	ACBL	R4, #4, ENTRY, 5\$		0516
				52	D0	0006C	7\$:	MOVL	TOTAL, R0		0523
				04	0006F			RET			0525

; Routine Size: 112 bytes, Routine Base: \$CODE\$ + 01EA

; 413 0526 1


```

415 0527 1 %SBTTL 'write_new_sdbsf -- Write a new SDBSF to the medium'
416 0528 1 ROUTINE write_new_sdbsf : NOVALUE =
417 0529 1 ++
418 0530 1
419 0531 1 Functional Description:
420 0532 1
421 0533 1 Perform house cleaning on the SDBSF, by filling the remaining unused bad
422 0534 1 block descriptors with the selected fill character. If the device is a
423 0535 1 non last track format device, the bad block file is checksum'd and the
424 0536 1 checksum is left in the last word of the file. If the device is a last
425 0537 1 track class device (ie. has more than bad$k_nltmaxblk), then the DATACHECK
426 0538 1 feature of the QIO is used to assure that the block was written correctly.
427 0539 1
428 0540 1 --
429 0541 2 BEGIN
430 0542 2 LOCAL
431 0543 2 fill;
432 0544 2
433 0545 2 bad$gl_func = IOS$WRITEBLK;
434 0546 2 fill = (IF .bad$gl_context [ctx_v_ltdevice]
435 0547 2 THEN =1
436 0548 2 ELSE 0);
437 0549 2
438 0550 2 IF .bad$gl_context [ctx_v_ltdevice]
439 0551 2 THEN
440 0552 2 bad$gl_func = .bad$gl_func OR IOS$DATACHECK;
441 0553 2
442 0554 2 INCR entry FROM .bad$gl_sdbsf_ptr TO .bad$ga_sdbsf + bad$k_page_size - 4 BY 4 DO
443 0555 2 .entry = .fill;
444 0556 2
445 0557 2 IF .bad$gl_context [ctx_v_ltdevice]
446 0558 2 THEN
447 0559 2 write_last_track ()
448 0560 2 ELSE
449 0561 2 BEGIN
450 0562 2 CHECKSUM2 (.bad$ga_sdbsf, bad$k_page_size - 2);
451 0563 2 write_non_lsttrk ();
452 0564 2 END;
453 0565 2
454 0566 1 END; ! of ROUTINE write_new_sdbsf
```

000C 00000 WRITE_NEW SDBSF:											
									0528		
									0545		
53	0000G	CF	0000G	CF	20	D0	00002	WORD	Save R2,R3	0546	
				01	00	EF	00007	MOVL	#32, BAD\$GL_FUNC		
				05	53	E9	0000E	EXTZV	#0, #1, BAD\$GL_CONTEXT+1, R3		
				52	01	CE	00011	BLBC	R3, 1\$		
					02	11	00014	MNEGL	#1, FILL	0547	
					52	D4	00016	BRB	2\$		
					53	E9	00018	CLRL	FILL	0546	
				06	8F	88	0001B	BLBC	R3, 3\$	0550	
				CF	8F	C1	00021	BISB2	#64, BAD\$GL_FUNC+1	0552	
				51	CF	000001FC	ADDL3	#508, BAD\$GA_SDBSF, R1	0554		
				50	0000G	CF	D0	0002B	MOVL	BAD\$GL_SDBSF_PTR, ENTRY	0555

BADMAIN
V04-000

Analyze/Media Main Module

write_new_sdbsf -- Write a new SDBSF to the med

1 8
15-Sep-1984 23:41:46
14-Sep-1984 11:54:24

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[BAD.SRC]BADMAIN.B32;1

Page 16
(8)

	80		03	11	00030	BRB	5\$	:	
	51		52	D0	00032	MOVL	FILL, (ENTRY)+	:	
			50	D1	00035	CMPL	ENTRY, R1	:	
			FB	15	00038	BLEQ	4\$	:	
	06		53	E9	0003A	BLBC	R3, 6\$	:	0557
0000V	CF		00	FB	0003D	CALLS	#0, WRITE_LAST_TRACK	:	0559
				04	00042	RET		:	
	7E	01FE	8F	3C	00043	MOVZWL	#510, -(SP)	:	0562
		0000G	CF	DD	00048	PUSHL	BAD\$GA_SDBSF	:	
00000000G	00		02	FB	0004C	CALLS	#2, CHECKSUM2	:	
0000V	CF		00	FB	00053	CALLS	#0, WRITE_NON_LSTTRK	:	0563
				04	00058	RET		:	0566

; Routine Size: 89 bytes, Routine Base: \$CODE\$ + 025A

; 455 0567 1


```

: 457 0568 1 %SBTTL 'write_last_track -- Write the new SDBSF onto the last track'
: 458 0569 1 ROUTINE write_last_track : NOVALUE =
: 459 0570 1 ++
: 460 0571 1
: 461 0572 1 Functional Description:
: 462 0573 1
: 463 0574 1 Write the bad block file to the remainder of the last track, following
: 464 0575 1 the MDBSF.
: 465 0576 1
: 466 0577 1 Side Effects:
: 467 0578 1
: 468 0579 1 If we could not write the updated SDBSF to the medium, then signal
: 469 0580 1 a fatal error and terminate the analysis (via SIGNAL_STOP).
: 470 0581 1
: 471 0582 1 --
: 472 0583 2 BEGIN
: 473 0584 2 INCR block FROM 10 TO .bad$gl_sectors / .bad$gb_block_fact - 1 DO
: 474 0585 2 IF NOT (bad$gl_status = bad$sync_io (.bad$ga_sdbsf,
: 475 0586 2 bad$sk_page_size,
: 476 0587 2 .bad$gl_mdbsf_ptr + .block))
: 477 0588 2 THEN
: 478 0589 2 SIGNAL_STOP (bad$_bbfwr, 1, bad$ga_device, .bad$gl_status);
: 479 0590 2
: 480 0591 1 END; ! of ROUTINE write_last_track
```

000C 00000 WRITE_LAST_TRACK:

		53	0000G	CF	9A	00002		WORD	Save R2,R3		0569
		53	CF	53	C7	00007		MOVZBL	BAD\$GB_BLOCK_FACT, R3		0584
53	0000G	52		09	D0	0000D		DIVL3	R3, BAD\$GL_SECTORS, R3		
				34	11	00010		MOVL	#9, BLOCK		
								BRB	2\$		
			0000GDF	42	9F	00012	1\$:	PUSHAB	@BAD\$GL_MDBSF_PTR[BLOCK]		0587
		7E	0200	8F	3C	00017		MOVZWL	#512, -(SP)		0585
			0000G	CF	DD	0001C		PUSHL	BAD\$GA_SDBSF		
	00000000G	00		03	FB	00020		CALLS	#3, BAD\$SYNC_IO		
	0000G	CF		50	D0	00027		MOVL	R0, BAD\$GL_STATUS		
		17		50	E8	0002C		BLBS	R0, 2\$		
			0000G	CF	DD	0002F		PUSHL	BAD\$GL_STATUS		0589
			0000G	CF	9F	00033		PUSHAB	BAD\$GA_DEVICE		
				01	DD	00037		PUSHL	#1		
	00000000G	00	00000000G	8F	DD	00039		PUSHL	#BAD\$ BBFWRT		
				04	FB	0003F		CALLS	#4, LIB\$STOP		
C8		52		53	F2	00046	2\$:	AOBLSS	R3, BLOCK, 1\$		0585
				04	0004A			RET			0591

; Routine Size: 75 bytes, Routine Base: \$CODE\$ + 02B3

; 481 0592 1

```

483 0593 1 %SBTTL 'write_non_lsttrk -- Write the new SDBSF on the last good block'
484 0594 1 ROUTINE write_non_lsttrk : NOVALUE =
485 0595 1 ++
486 0596 1
487 0597 1 Functional Description:
488 0598 1
489 0599 1 Write the bad block file to the last good block on the device (that is
490 0600 1 somewhere within the last 256 block on the device).
491 0601 1
492 0602 1 Side Effects:
493 0603 1
494 0604 1 If we could not write the updated SDBSF to the medium, then signal
495 0605 1 a fatal error and terminate the analysis (via SIGNAL_STOP).
496 0606 1
497 0607 1 --
498 0608 2 BEGIN
499 0609 2 LOCAL
500 0610 2 block,
501 0611 2 write_ok;
502 0612 2
503 0613 2 write_ok = FALSE;
504 0614 2
505 0615 2 IF NOT .bad$gl_context [ctx_v_ltdevice]
506 0616 2 THEN
507 0617 2 block = .bad$gl_maxblock
508 0618 2 ELSE
509 0619 2 block = .bad$gl_mdbsf_ptr;
510 0620 2
511 0621 2 INCR offset FROM 1 TO 255 DO
512 0622 3 BEGIN
513 0623 4 IF (bad$gl_status = bad$sync_io (.bad$ga_sdbsf,
514 0624 4 bad$gl_page_size,
515 0625 4 .block - .offset))
516 0626 3 THEN
517 0627 4 BEGIN
518 0628 4 write_ok = TRUE;
519 0629 4 EXITLOOP;
520 0630 4 END;
521 0631 2 END;
522 0632 2
523 0633 2 IF NOT .write_ok
524 0634 2 THEN
525 0635 2 SIGNAL_STOP (bad$_bbfwr, 1, bad$ga_device, .bad$gl_status);
526 0636 2
527 0637 1 END; ! of ROUTINE write_non_lsttrk

```

```

! If this is a true non last track device start
! at the last block.
! If the device is last track converted by
! /OVERRIDE, then protect the last track.
! Write the SDBSF until we get success or run
! out of rope!

```

001C 00000 WRITE_NON_LSTTRK:

				WORD	Save R2,R3,R4	: 0594
		53	D4 00002	CLRL	WRITE_OK	: 0613
07	0000G	CF	E8 00004	BLBS	BAD\$GL_CONTEXT+1, 1\$	: 0615
54	0000G	CF	D0 00009	MOVL	BAD\$GL_MAXBLOCK, BLOCK	: 0617
		05	11 0000E	BRB	2\$	: 0619
54	0000G	CF	D0 00010 1\$:	MOVL	BAD\$GL_MDBSF_PTR, BLOCK	

BADMAIN
V04-000

Analyze/Media Main Module

write_non_lsttrk -- Write the new SDBSF on the

L 8
15-Sep-1984 23:41:46
14-Sep-1984 11:54:24

VAX-11 Bliss-32 V4.0-742

DISK\$VMSMASTER:[BAD.SRC]BADMAIN.B32;1 Page 19
(10)

```

      52      01 D0 00015 2$:      MOVL      #1, OFFSET      : 0625
      54      52 C3 00018 3$:      SUBL3      OFFSET, BLOCK, -(SP)
      7E      8F 3C 0001C      MOVZWL      #512, -(SP)      : 0623
      0200      CF DD 00021      PUSHL      BAD$GA_SDBSF
      0000G      03 FB 00025      CALLS      #3, BAD$SYNC_IO
      0000G      50 D0 0002C      MOVL      R0, BAD$GL_STATUS
      05      50 E9 00031      BLBC      R0, 4$
      53      01 D0 00034      MOVL      #1, WRITE_OK      : 0628
      08      11 00037      BRB      5$      : 0627
      D7      52 000000FF      8F F3 00039 4$:      AOBLEQ      #255, OFFSET, 3$
      17      53 E8 00041 5$:      BLBS      WRITE OK, 6$      : 0621
      0000G      CF DD 00044      PUSHL      BAD$GE_STATUS      : 0633
      0000G      CF 9F 00048      PUSHAB     BAD$GA_DEVICE
      01 DD 0004C      PUSHL      #1
      00000000G 8F DD 0004E      PUSHL      #BAD$ BBFWRT
      00      04 FB 00054      CALLS      #4, LIB$STOP      : 0635
      04 0005B 6$:      RET      : 0637
```

; Routine Size: 92 bytes, Routine Base: \$CODE\$ + 02FE

```

: 528      0638 1
: 529      0639 1 END
: 530      0640 0 ELUDOM
```

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	32	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODE\$	858	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	12	0	1000	00:01.4

COMMAND QUALIFIERS

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

BADMAIN
V04-000

Analyze/Media Main Module
write_non_lsttrk -- Write the new SDBSF on the

M 8
15-Sep-1984 23:41:46

VAX-11 Bliss-32 V4.0-742

Page 20

: Size: 858 code + 32 data bytes
: Run Time: 00:10.5
: Elapsed Time: 00:48.3
: Lines/CPU Min: 3650
: Lexemes/CPU-Min: 13579
: Memory Used: 112 pages
: Compilation Complete

0018 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

