

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

```

0001 0 MODULE SCANFILE (
0002 0      IDENT = 'V04-001'
0003 0 ) =
0004 1 BEGIN
0005 1
0006 1
0007 1 *****
0008 1 *
0009 1 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0010 1 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0011 1 *  ALL RIGHTS RESERVED.
0012 1 *
0013 1 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0014 1 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0015 1 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0016 1 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0017 1 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0018 1 *  TRANSFERRED.
0019 1 *
0020 1 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0021 1 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0022 1 *  CORPORATION.
0023 1 *
0024 1 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 1 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 1 *
0027 1 *
0028 1 *****
0029 1
0030 1 ++
0031 1 FACILITY:
0032 1     DYNAMIC BAD BLOCK UTILITY
0033 1
0034 1 ABSTRACT:
0035 1     This process examines files suspected of containing bad
0036 1     disk blocks. Those disk blocks verified to be bad are added
0037 1     to the bad block file. The others are returned to the volume
0038 1     for reuse.
0039 1
0040 1 ENVIRONMENT:
0041 1     VAX/VMS OPERATING SYSTEM, VERSION 1.0
0042 1
0043 1 AUTHOR:THOMAS G. DOPIRAK     , CREATION DATE:5/16/78
0044 1
0045 1 MODIFIED BY:
0046 1
0047 1     V04-001 HH0059     Hai Huang     01-Oct-1984
0048 1     Enhance BADBLOCK.EXE to handle a larger class of I/O errors.
0049 1     Rework completion routines. General code cleanup.
0050 1
0051 1     V0002  ACG0059     Andrew C. Goldstein,  21-Aug-1979  20:45
0052 1     Fix repeated write/read test so it really repeats
0053 1
0054 1 --

```

```

56 0055 1 |
57 0056 1 | Table of contents:
58 0057 1 |
59 0058 1 | FORWARD ROUTINE
60 0059 1 |     SCAN : NOVALUE, | main program of file processing logic
61 0060 1 |     GROUP_BLOCKTEST, | routine tests 'groups' of blocks
62 0061 1 |     GROUP_RETURN, | examines a group of blocks when error detected
63 0062 1 |     CHECK_BADSTATUS, | determines whether a status indicates a bad block
64 0063 1 |     NORMAL_COMPLETE : NOVALUE, | called after normal processing of a file
65 0064 1 |     ERROR_COMPLETE : NOVALUE, | called after abnormal processing of a file
66 0065 1 |     TRUNCATE, | truncates a file containing no bad blocks
67 0066 1 |     TRUNCATE_BAD, | truncates a file containing a bad block
68 0067 1 |     BLOCKTEST | tests an individual block
69 0068 1 |     POSITION_TO_EOF, | accesses a file and discovers its size
70 0069 1 |     DO_QIOW, | does QIOW'S and checks status
71 0070 1 |     REMOVE_PBB : NOVALUE; | remove PBB entries from BADLOG.SYS
72 0071 1 |
73 0072 1 |
74 0073 1 | Include files:
75 0074 1 |
76 0075 1 | LIBRARY 'SYSS$LIBRARY:LIB.L32';
77 0076 1 |
78 0077 1 |
79 0078 1 | MACROS:
80 0079 1 |
81 0080 1 | MACRO
82 0081 1 |     DIRECTORY_ID=FIB[FIBSW_DID]%, | start of directory ID
83 M 0082 1 |     CSTRING[] = (UPLIT_BYTE(%CHARCOUNT(%STRING(%REMAINING))),
84 0083 1 |     %STRING(%REMAINING)) % ;
85 0084 1 |
86 0085 1 |
87 0086 1 | Various definitions:
88 0087 1 |
89 0088 1 | LITERAL
90 0089 1 |     BLOCK_TEST_SIZE=15, | number of blocks in a group
91 0090 1 |     GROUP_SIZE=BLOCK_TEST_SIZE*512, | number of bytes in a group
92 0091 1 |     TRIALS_TO_SUC=3; | number of times a block must be successfully
93 0092 1 | | be written before its declared no bad
94 0093 1 |
95 0094 1 | OWN storage:
96 0095 1 |
97 0096 1 | OWN
98 0097 1 |
99 0098 1 |
100 0099 1 | Blocks of test data.
101 0100 1 |
102 0101 1 | DISK_TEXT : VECTOR[GROUP_SIZE,BYTE],
103 0102 1 | BUFFER : VECTO [GROUP_SIZE,BYTE],
104 0103 1 | GROUP_TEST_DATA : VECTOR[GROUP_SIZE,BYTE],
105 0104 1 |
106 0105 1 | READ_FAIL, | logical indicating if group failed on read
107 0106 1 | TRUNC_BLOCK, | first VBN actually removed in truncate
108 0107 1 | BAD_COUNT : INITIAL(0), | count of bad blocks found
109 0108 1 | STARTING_BLOCK, | first VBN in group
110 0109 1 | LAST_BLOCK, | last block in a group
111 0110 1 | IOSB : VECTOR[4,WORD] ,INITIAL(0,0), | IOSB for disk operations
112 0111 1 | FIB : BLOCK[FIBSC_LENGTH,BYTE]; | file identification block

```

```
113 0112 1  
114 0113 1  
115 0114 1 : Equated symbols:  
116 0115 1  
117 0116 1 BIND  
118 0117 1  
119 0118 1  
120 0119 1 : Symbols for types of block test results  
121 0120 1  
122 0121 1 NORMAL_STS=0, : test completed normally  
123 0122 1 ERROR_STS=1, : non-recoverable error  
124 0123 1 BAD_STS=2, : bad block indicated  
125 0124 1 TRUE=1,  
126 0125 1 FALSE=0,  
127 0126 1 FIB_DESC=UPLIT(FIB$C_LENGTH,FIB),  
128 0127 1 BUFT = DISK_TEXT, : bind buffer names  
129 0128 1 BUF2 = BUFFER;  
130 0129 1  
131 0130 1 :  
132 0131 1 : External references:  
133 0132 1  
134 0133 1 EXTERNAL  
135 0134 1 CHANNEL : WORD,  
136 0135 1 MBX_CHANNEL : WORD, : channel to F11BXQP mailbox  
137 0136 1 ACP_MAIL : BLOCKC, BYTE], : buffer from F11BXQP  
138 0137 1 OLD_UCB;  
139 0138 1  
140 0139 1 EXTERNAL ROUTINE  
141 0140 1 SET_UCB;  
142 0141 1
```



```

201 0199
202 0200
203 0201
204 0202
205 0203
206 0204
207 0205
208 0206
209 0207
210 0208
211 0209
212 0210
213 0211
214 0212
215 0213
216 0214
217 0215
218 0216
219 0217
220 0218
221 0219
222 0220
223 0221
224 0222
225 0223
226 0224
227 0225
228 0226
229 0227
230 0228
231 0229
232 0230
233 0231
234 0232
235 0233
236 0234
237 0235
238 0236
239 0237
240 0238
241 0239
242 0240
243 0241
244 0242
245 0243
246 0244
247 0245

```

```

STARTING_BLOCK=.LAST_BLOCK-BLOCK_TEST_SIZE+1;

:
: Test group of blocks.
: Action depends upon whether any 'bad' blocks found.
CASE CHECK_BADSTATUS.GROUP_BLOCKTEST()
FROM NORMAL_STS TO BAD_STS OF

SET

[NORMAL_STS] :      IF .STARTING_BLOCK EQLU 1      ! successful test
                    THEN
                    BEGIN
                    NORMAL_COMPLETE();
                    RETURN;
                    END
                    ELSE
                    LAST_BLOCK=.STARTING_BLOCK-1;

[ERROR_STS] :      BEGIN                                ! error, but not badblock error
                    ERROR_COMPLETE();
                    RETURN
                    END;

[BAD_STS] :        BEGIN                                ! bad block found, scan inividual blocks
                    IF NOT GROUP_RETURN()
                    THEN
                    BEGIN
                    ERROR_COMPLETE();
                    RETURN
                    END;

                    IF (.STARTING_BLOCK EQLU 1)
                    OR (.TRUNC_BLOCK LEQ 1)
                    THEN
                    BEGIN
                    NORMAL_COMPLETE();
                    RETURN
                    END
                    ELSE
                    LAST_BLOCK=.TRUNC_BLOCK-1
                    END;

TES:
END:                                     ! end of while true loop

END;
END;

```

```

.TITLE SCANFILE
.IDENT \V04-001\
.PSECT SPLITS,NOWRT,NOEXE,2

```

```

00000040 00000 P.AAA: .LONG 64
00000000 00004 .ADDRESS FIB

```

⋮

.PSECT \$OWNS,NOEXE,2

```

00000 DISK_TEXT:
01E00 BUFFER: .BLKB 7680
03C00 GROUP_TEST_DATA:
      .BLKB 7680
05A00 READ_FAIL:
      .BLKB 4
05A04 TRUNC_BLOCK:
      .BLKB 4
00000000 05A08 BAD_COUNT:
      .LONG 0
05A0C STARTING_BLOCK:
      .BLKB 4
05A10 LAST_BLOCK:
      .BLKB 4
00000000 00000000 05A14 IOSB: .LONG 0, 0
05A1C FIB: .BLKB 64

```

```

NORMAL_STS= 0
ERROR_STS= 1
BAD_STS= 2
TRUE= 1
FALSE= 0
FIB_DESC= P.AAA
BUF1= DISK TEXT
BUF2= BUFFER

```

```

.EXTRN CHANNEL, MBX_CHANNEL
.EXTRN ACP_MAIL, OLD_UCB
.EXTRN SET_UCB

```

.PSECT \$CODES, NOWRT, 2

0040	8F	00	56	0000'	007C	00000	.ENTRY	SCAN, Save R2,R3,R4,R5,R6	: 0142
			6E		CF	9E 00002	MOVAB	STARTING_BLOCK, R6	: 0177
				10	00	2C 00007	MOVCS	#0, (SP); #0, #64, FIB	: 0182
		0000V	CF		A6	0000E			: 0195
			59		00	FB 00010	CALLS	#0, POSITION_TO_EOF	: 0197
			0F	04	50	E9 00015	BLBC	R0, 11\$: 0199
					A6	D1 00018 1\$:	CMPL	LAST_BLOCK, #15	: 0205
			66		05	1E 0001C	BGEQU	2\$: 0210
					01	D0 0001E	MOVL	#1, STARTING_BLOCK	: 0217
		66			05	11 00021	BRB	3\$: 0225
			04		0E	C3 00023 2\$:	SUBL3	#14, LAST_BLOCK, STARTING_BLOCK	: 0225
		0000V	A6		00	FB 00028 3\$:	CALLS	#0, GROUP_BLOCKTEST	: 0225
			CF		50	DD 0002D	PUSHL	R0	: 0225
					01	FB 0002F	CALLS	#1, CHECK_BADSTATUS	: 0225
		0000V	CF		50	CF 00034	CASEL	R0, #0, #2	: 0225
		0012	00	0006	00	CF 00038 4\$:	.WORD	5\$-4\$,-	: 0225
								7\$-4\$,-	: 0225
								6\$-4\$: 0225
			01	66	D1	0003E 5\$:	CMPL	STARTING_BLOCK, #1	: 0225
				20	13	00041	BEQL	9\$: 0225
	04	A6	66	01	C3	00043	SUBL3	#1, STARTING_BLOCK, LAST_BLOCK	: 0225
				CE	11	00048	BRB	1\$: 0225
		0000V	CF	00	FB	0004A 6\$:	CALLS	#0, GROUP_RETURN	: 0225

SCANFILE
V04-001

D 2
8-Jan-1985 17:22:51 VAX-11 Bliss-32 V4.0-742
7-Nov-1984 11:22:13 [BADBLK.BUGSRC]SCANFILE.B32;2

Page 7
(3)

SCA
V04

0000V	C6	50	E8	0004F	BLBS	R0, 8\$:	
	CF	00	FB	00052 7\$:	CALLS	#0, ERROR_COMPLETE	:	0228
				04 C0057	RET		:	0227
	01	66	D1	00058 8\$:	CMPL	STARTING_BLOCK, #1	:	0232
		06	13	0005B	BEQL	9\$:	
	01	F8	A6	D1 0005D	CMPL	TRUNC_BLOCK, #1	:	0233
		06	14	00061	BGTR	10\$:	
0000V	CF	00	FB	00063 9\$:	CALLS	#0, NORMAL_COMPLETE	:	0236
				04 00068	RET		:	0235
04	A6	F8	A6	01 C3 00069 10\$:	SUBL3	#1, TRUNC_BLOCK, LAST_BLOCK	:	0240
				A7 11 0006F	BRB	1\$:	0189
				04 00071 11\$:	RET		:	0245

: Routine Size: 114 bytes, Routine Base: \$ ODES + 0000

: 248 0246 1

.....

```

250 0247 1 ROUTINE POSITION_TO_EOF =
251 0248 1
252 0249 1 ++
253 0250 1 FUNCTIONAL DESCRIPTION:
254 0251 1
255 0252 1 Routine initializes the FIB, accesses the file whose
256 0253 1 FID is the acp_mail, and determines the files length
257 0254 1 in blocks.
258 0255 1
259 0256 1 CALLING SEQUENCE:
260 0257 1 CALL OPSITION_TO_EOF
261 0258 1
262 0259 1 FORMAL PARAMETERS:
263 0260 1 NONE
264 0261 1
265 0262 1 IMPLICIT INPUTS:
266 0263 1 ACP_MAIL[BBSSW_FID]: File ID of suspect file
267 0264 1
268 0265 1 IMPLICIT OUTPUTS:
269 0266 1 LAST_BLOCK: Total number of blocks in file
270 0267 1 FIB: Assorted fields set by IOS_ACCESS
271 0268 1
272 0269 1 ROUTINE VALUE:
273 0270 1 If IOS_ACCESS fails then that code is returned
274 0271 1
275 0272 1 SIDE EFFECTS:
276 0273 1 NONE
277 0274 1
278 0275 1 --
279 0276 1
280 0277 2 BEGIN
281 0278 2
282 0279 2
283 0280 2 OWN
284 0281 2 STAT_BLOCK:VECTOR[5,WORD], : space for file statistics block
285 0282 2 : returned by IOS_ACCESS
286 0283 2 ATTRIBUTES:VECTOR[3]
287 0284 2 INITIAL(ATRSC_STATBLK*16+10,STAT_BLOCK,0);
288 0285 2
289 0286 2
290 0287 2 : Set file access attributes.
291 0288 2
292 0289 2 FIB[FIBSV_WRITE]=1;
293 0290 2 FIB[FIBSV_TRUNC]=1;
294 0291 2
295 0292 2
296 0293 2 : Push File ID into fib.
297 0294 2
298 0295 2 CHSMOVE(6,ACP_MAIL[BBSSW_FID],FIB[FIBSW_FID]);
299 0296 2
300 0297 2
301 0298 2 : Open the specified file and get its size in blocks.
302 0299 2
303 0300 2 DO_QIOW(IOS_ACCESS+IOSM_ACCESS,FIB_DESC,0,0,0,ATTRIBUTES);
304 0301 2
305 0302 2
306 0303 2 : Move the word swapped virtual block number.

```

```

: 307      0304 2 1
: 308      0305 2 LAST_BLOCK<0,16>=.STAT_BLOCK[3];
: 309      0306 2 LAST_BLOCK<16,16>=.STAT_BLOCK[2];
: 310      0307 2
: 311      0308 2 RETURN TRUE;
: 312      0309 1 END;

```

```

.PSECT $OWNS,NOEXE,2
05A5C STAT_BLOCK:
.BLKB 10
05A66 .BLKB 2
0009000A 05A68 ATTRIBUTES:
.LONG 589834
00000000' 05A6C .ADDRESS STAT_BLOCK
00000000 05A70 .LONG 0

```

```

.PSECT $CODE$,NOWRT,2
007C 0000 POSITION TO EOF:

```

		56	0000'	CF	9E	00002	WORD	Save R2,R3,R4,R5,R6	:	0247
	0D	A6		01	88	00007	MOVAB	LAST_BLOCK, R6	:	
	23	A6		01	88	0000B	BISB2	#1, FIB+1	:	0289
10	A6	0000G		06	28	0000F	BISB2	#1, FIB+23	:	0290
			58	A6	9F	00016	MOV3	#6, ACP MAIL+12, FIB+4	:	0295
				7E	7C	00019	PUSHAB	ATTRIBUTES	:	0300
				7E	D4	0001B	CLRQ	-(SP)	:	
			0000'	CF	9F	0001D	CLRL	-(SP)	:	
		7E		72	8F	9A	PUSHAB	FIB_DESC	:	
	0000V	CF		06	FB	00025	MOVZBL	#114, -(SP)	:	
		66	52	A6	B0	0002A	CALLS	#6, DO QIOW	:	0305
	02	A6	50	A6	B0	0002E	MOVW	STAT_BLOCK+6, LAST_BLOCK	:	0306
		50		01	D0	00033	MOVW	STAT_BLOCK+4, LAST_BLOCK+2	:	0308
				04	00	00036	MOVL	#1, R0	:	0309
							RET		:	

: Routine Size: 55 bytes. Routine Base: \$CODE\$ + 0072

: 313 0310 1

```

315 0311 1 ROUTINE TRUNCATE (VBN) =
316 0312 1
317 0313 1 +-
318 0314 1 FUNCTIONAL DESCRIPTION:
319 0315 1
320 0316 1 Routine truncates of the end of the current file
321 0317 1 starting at the indicated block number. Because of
322 0318 1 clustering not all blocks requested may be truncated.
323 0319 1 Last block truncated is placed into trunc_block.
324 0320 1
325 0321 1 CALLING SEQUENCE:
326 0322 1 CALL TRUNCATE (ARG1)
327 0323 1
328 0324 1 FORMAL PARAMETERS:
329 0325 1 ARG1: Virtual block at which to start truncate
330 0326 1
331 0327 1 IMPLICIT INPUTS:
332 0328 1 NONE
333 0329 1
334 0330 1 IMPLICIT OUTPUTS:
335 0331 1 NONE
336 0332 1
337 0333 1 ROUTINE VALUE:
338 0334 1 Status of IOS_MODIFY operation is returned
339 0335 1
340 0336 1 SIDE EFFECTS:
341 0337 1 NONE
342 0338 1
343 0339 1 --
344 0340 1
345 0341 2 BEGIN
346 0342 2 LOCAL
347 0343 2 STATUS;
348 0344 2
349 0345 2 :
350 0346 2 : Set block to truncate at.
351 0347 2
352 0348 2 FIB[FIB$L_EXVBN]=.VBN;
353 0349 2
354 0350 2 :
355 0351 2 : Truncate a piece off of file.
356 0352 2
357 0353 2 STATUS=DO_QIOW(IOS_MODIFY,FIB_DESC,0,0,0,0);
358 0354 2
359 0355 2 :
360 0356 2 : Clear size field.
361 0357 2
362 0358 2 FIB[FIB$L_EXSZ]=0;
363 0359 2
364 0360 2 :
365 0361 2 : Check for rounding from clustering.
366 0362 2
367 0363 2 IF .VBN NEQ .FIB[FIB$L_EXVBN]
368 0364 2 THEN
369 0365 2 TRUNC_BLOCK=.FIB[FIB$L_EXVBN]
370 0366 2 ELSE
371 0367 2 TRUNC_BLOCK=.VBN;

```



```

: 378 0373 1 ROUTINE TRUNCATE_BAD (VBN) =
: 379 0374 1
: 380 0375 1 ++
: 381 0376 1 FUNCTIONAL DESCRIPTION:
: 382 0377 1
: 383 0378 1 TRUNCATE_BAD performs 2 truncation operations.
: 384 0379 1 All blocks after (higher VBN's) are returned to
: 385 0380 1 the system via a call to truncate. The current
: 386 0381 1 VBN known as 'bad' is truncated off the current file
: 387 0382 1 and onto the bad block file. Due to clustering, more blocks
: 388 0383 1 than requested may be added to the bad block file and
: 389 0384 1 trunc_block is set to the last block added.
: 390 0385 1
: 391 0386 1 CALLING SEQUENCE:
: 392 0387 1 TRUNCATE_BAD (ARG1)
: 393 0388 1
: 394 0389 1 FORMAL PARAMETERS:
: 395 0390 1 ARG1: Virtual Block Number of block to mark bad
: 396 0391 1
: 397 0392 1 IMPLICIT INPUTS:
: 398 0393 1 NONE
: 399 0394 1
: 400 0395 1 IMPLICIT OUTPUTS:
: 401 0396 1 TRUNC_BLOCK: last block (lowest VBN) added to bad block file
: 402 0397 1
: 403 0398 1 ROUTINE VALUE:
: 404 0399 1 If either truncate operation fails then that status is returned
: 405 0400 1
: 406 0401 1 SIDE EFFECTS:
: 407 0402 1 NONE
: 408 0403 1
: 409 0404 1 --
: 410 0405 1
: 411 0406 2 BEGIN
: 412 0407 2 LOCAL
: 413 0408 2 STATUS;
: 414 0409 2
: 415 0410 2 BAD_COUNT=.BAD_COUNT+1;
: 416 0411 2
: 417 0412 2
: 418 0413 2 Truncate off good portions of file.
: 419 0414 2
: 420 0415 2 STATUS=TRUNCATE(.VBN+1);
: 421 0416 2
: 422 0417 2 IF (.STATUS NEQ SSS_NORMAL)
: 423 0418 2 AND (.STATUS NEQ SSS_ENDOFFILE)
: 424 0419 2 THEN
: 425 0420 2 RETURN .STATUS;
: 426 0421 2
: 427 0422 2
: 428 0423 2 Set block to truncate at.
: 429 0424 2
: 430 0425 2 FIB[FIB$L_EXVBN]-.VBN;
: 431 0426 2
: 432 0427 2
: 433 0428 2 Note return is to bad block file.
: 434 0429 2

```

:
:
:
:
:
:

: Ro

```

435 0430 FIB[FIBSV_MARKBAD]=1;
436 0431
437 0432
438 0433
439 0434
440 0435 STATUS=DO_QIOW(IOS_MODIFY,FIB_DESC,0,0,0,0);
441 0436
442 0437
443 0438
444 0439
445 0440 FIB[FIBSL_EXSZ]=0;
446 0441
447 0442
448 0443
449 0444
450 0445 IF .VBN NEQ .FIB[FIBSL_EXVBN]
451 0446 THEN
452 0447 TRUNC_BLOCK=.FIB[FIBSL_EXVBN]
453 0448 ELSE
454 0449 TRUNC_BLOCK=.VBN;
455 0450
456 0451
457 0452
458 0453
459 0454 FIB[FIBSV_MARKBAD]=0;
460 0455
461 0456 RETURN .STATUS
462 0457
463 0458 1 END;

```

				0004	0000	TRUNCATE_BAD:			
						WORD	Save R2		0373
		52	0000'	CF	9E	00002	MOVAB	FIB+28, R2	
			DO	A2	D6	00007	INCL	BAD_COUNT	0410
7E	04	AC		01	C1	0000A	ADDL3	#1, VBN, -(SP)	0415
	BF	AF		01	FB	0000F	CALLS	#1, TRUNCATE	
		01		50	D1	00013	CMPL	STATUS, #1	0417
				09	13	00016	BEQL	1\$	
	00000870	8F		50	D1	00018	CMPL	STATUS, #2160	0418
				2F	12	0001F	BNEQ	4\$	
		62	04	AC	D0	00021	1\$:	MOVL	VBN, FIB+28
		FB	A2	04	88	00025	BISB2	#4, FIB+23	0425
				7E	7C	00029	CLRQ	-(SP)	0430
				7E	7C	0002B	CLRQ	-(SP)	0435
			0000'	CF	9F	0002D	PUSHAB	FIB_DESC	
				36	DD	00031	PUSHL	#54	
	0000V	CF		06	FB	00033	CALLS	#6, DO_QIOW	
			FC	A2	D4	00038	CLRL	FIB+24	0440
		62	04	AC	D1	0003B	CMPL	VBN, FIB+28	0445
				06	13	0003F	BEQL	2\$	
	CC	A2		62	D0	00041	MOVL	FIB+28, TRUNC_BLOCK	0447
				05	11	00045	BRB	3\$	
	CC	A2	04	AC	D0	00047	2\$:	MOVL	VBN, TRUNC_BLOCK
									0449

SCANFILE
V04-001

K 2
8-Jan-1985 17:22:51 VAX-11 Bliss-32 V4.0-742
7-Nov-1984 11:22:13 [BADBLK.BUGSRC]SCANFILE.B32;2

Page 14
(6)

FB A2

04 8A 0004C 3S: BICB2 #4, FIB+23
04 00050 4S: RET

: 0454
: 0458

; Routine Size: 81 bytes, Routine Base: \$CODES + 00D7

; 464 0459 1

SCAN
V04-0

; Ro

; 8


```

466 0460 1 ROUTINE BLOCKTEST(VBN) =
467 0461 1
468 0462 1
469 0463 1 **
470 0464 1 FUNCTIONAL DESCRIPTION:
471 0465 1 This routine tests a single virtual block
472 0466 1 for 'badness'. The routine reads the block a number
473 0467 1 of times, checking for a data sensitive condition, and then
474 0468 1 writes and reads back the worst case pattern. Upon any
475 0469 1 abnormal condition the routine exits with that status.
476 0470 1
477 0471 1 CALLING SEQUENCE:
478 0472 1 CALL BLOCKTEST (ARG1)
479 0473 1
480 0474 1 FORMAL PARAMETERS:
481 0475 1 ARG1: Virtual block to be tested
482 0476 1
483 0477 1 IMPLICIT INPUTS:
484 0478 1 READ_FAIL: a logical variable, when true indicates that
485 0479 1 group_blocktest encountered an error while reading the
486 0480 1 user data on the current group. This directs blocktest
487 0481 1 to read the individual blocks before overwriting them
488 0482 1
489 0483 1 IMPLICIT OUTPUTS:
490 0484 1 NONE
491 0485 1
492 0486 1 ROUTINE VALUE:
493 0487 1 If any QIOW fails then its status is returned
494 0488 1
495 0489 1 SIDE EFFECTS:
496 0490 1 NONE
497 0491 1
498 0492 1 --
499 0493 1
500 0494 2 BEGIN
501 0495 2
502 0496 2 LOCAL
503 0497 2 STATUS:
504 0498 2
505 0499 2
506 0500 2 If group test failed in data dependent manner, do a series of reads
507 0501 2 and writes of the original data.
508 0502 2
509 0503 2 Note that for DSA disks, writing the block could clear up the
510 0504 2 SSS_FORCEDERROR encountered during the read.
511 0505 2
512 0506 2 A double buffering technique is used for the read/write buffers. A boolean
513 0507 2 flag is used to indicate which buffer is to be written from and the other
514 0508 2 is to be read into.
515 0509 2
516 0510 2 For reference: following is the section of code that previously performed
517 0511 2 the data-sensitive test (reading the block a number of times).
518 0512 2
519 0513 2 IF .READ_FAIL
520 0514 2 THEN
521 0515 2 INCR TEST_INDEX FROM 1 TO TRIALS TO SUC DO
522 0516 2 IF NOT (STATUS=DO_QIOW(IOS_READVBLK,IOSM_INHRETRY,DISK_TEXT,512,.VBN,0,0))

```

```

523 0517 2 THEN
524 0518 2 RETURN .STATUS;
525 0519 2
526 0520 2
527 0521 2 IF .READ_FAIL
528 0522 2 THEN
529 0523 2 BEGIN
530 0524 2
531 0525 2 LOCAL FLAG; ! boolean flag
532 0526 2 FLAG;
533 0527 2
534 0528 2 FLAG = TRUE; ! init flag
535 0529 2
536 0530 2 DO_QIOW(IOS_READVBLK+IOSM_INHRETRY, BUF1, 512, .VBN, 0, 0);
537 0531 2
538 0532 3 INCR TEST_INDEX FROM 1 TO TRIALS_TO_SUC DO
539 0533 4 BEGIN
540 0534 4
541 0535 4 STATUS = DO_QIOW(IOS_WRITEVBLK+IOSM_INHRETRY,
542 0536 4 (IF .FLAG THEN BUF1
543 0537 4 ELSE BUF2),
544 0538 4 512, .VBN, 0, 0);
545 0539 4 IF NOT .STATUS
546 0540 4 THEN
547 0541 4 RETURN .STATUS;
548 0542 4
549 0543 4 STATUS = DO_QIOW(IOS_READVBLK+IOSM_INHRETRY,
550 0544 4 (IF .FLAG THEN BUF2
551 0545 4 ELSE BUF1),
552 0546 4 512, .VBN, 0, 0);
553 0547 4 IF NOT .STATUS
554 0548 4 THEN
555 0549 4 RETURN .STATUS;
556 0550 4
557 0551 4 IF CH$NEQ (512, BUF1, 512, BUF2)
558 0552 4 THEN
559 0553 4 RETURN $$$_PARITY;
560 0554 4
561 0555 4 FLAG = NOT .FLAG;
562 0556 4
563 0557 3 END; ! end of INCR loop
564 0558 2 END; ! end of REAL_FAIL condition
565 0559 2
566 0560 2 ! Block must pass read/write test multiple before being marked good.
567 0561 2
568 0562 2 INCR TEST_INDEX FROM 1 TO TRIALS_TO_SUC DO
569 0563 3 BEGIN
570 0564 3
571 0565 3 ! Write to the indicated disk block.
572 0566 3
573 0567 3
574 0568 3 IF NOT (STATUS=DO_QIOW(IOS_WRITEVBLK+IOSM_INHRETRY, GROUP_TEST_DATA, 512, .VBN, 0, 0))
575 0569 3 THEN
576 0570 3 RETURN .STATUS;
577 0571 3
578 0572 3 ! Try and read it back.
579 0573 3

```


1E00	C7	4A	55	E9	00076	BLBC	STATUS, 8\$: 0547
		67	8F	29	00079	CMPC3	#512, BUF1, BUF2	: 0551
			4E	12	00081	BNEQ	10\$: 0555
	A1	54	54	D2	00083	MCOML	FLAG, FLAG	: 0532
		56	03	F3	00086	AOBLEQ	#3, TEST_INDEX, 1\$: 0568
		54	01	D0	0008A	MOVL	#1, TEST_INDEX	: 0575
			7E	7C	0008D	CLRQ	-(SP)	: 0577
			04	AC	DD	PUSHL	VBN	: 0582
		7E	8F	3C	00092	MOVZWL	#512, -(SP)	: 0584
			3C00	C7	9F	PUSHAB	GROUP TEST DATA	: 0587
		7E	8F	3C	0009B	MOVZWL	#32818, -(SP)	: 0589
		68	06	FB	000A0	CALLS	#6, DO Q W	
		55	50	D0	000A3	MOVL	R0, STATUS	
		1A	55	E9	000A6	BLBC	STATUS, 8\$	
			7E	7C	000A9	CLRQ	-(SP)	
			04	AC	DD	PUSHL	VBN	
		7E	8F	3C	000AE	MOVZWL	#512, -(SP)	
			57	DD	000B3	PUSHL	R7	
		7E	8F	3C	000B5	MOVZWL	#32817, -(SP)	
		68	06	FB	000BA	CALLS	#6, DO QIOW	
		55	50	D0	000BD	MOVL	R0, STATUS	
		04	55	EB	000C0	BLBS	STATUS, 9\$	
		50	55	D0	000C3	MOVL	STATUS, R0	
			04	C0	000C6	RET		
	67	3C00	C7	8F	29	CMPC3	#512, GROUP_TEST_DATA, DISK_TEXT	
			50	06	13	BEQL	11\$	
			50	8F	3C	MOVZWL	#500, R0	
				04	000D6	RET		
	B2	54	03	F3	000D7	AOBLEQ	#3, TEST_INDEX, 7\$	
		50	01	D0	000DB	MOVL	#1, R0	
			04	00	000DE	RET		

; Routine Size: 223 bytes, Routine Base: \$CODE\$ + 0128

```

: 597 0590 1 ROUTINE GROUP_BLOCKTEST =
: 598 0591 1
: 599 0592 1 **
: 600 0593 1 FUNCTIONAL DESCRIPTION:
: 601 0594 1
: 602 0595 1 Routine tests groups of virtually contiguous blocks for
: 603 0596 1 'badness'. Should any of the I/O operations fail the status
: 604 0597 1 is immediately returned. Groups are read several times for
: 605 0598 1 error. A worst case is written to the group and then read
: 606 0599 1 back. The read data is compared with that written.
: 607 0600 1
: 608 0601 1 CALLING SEQUENCE:
: 609 0602 1 CALL GROUP_BLOCKTEST
: 610 0603 1
: 611 0604 1 FORMAL PARAMETERS:
: 612 0605 1 NONE
: 613 0606 1
: 614 0607 1 IMPLICIT INPUTS:
: 615 0608 1 STARTING_BLOCK: First virtual block in group
: 616 0609 1 LAST_BLOCK: Last virtual block in group
: 617 0610 1
: 618 0611 1 IMPLICIT OUTPUTS:
: 619 0612 1 NONE
: 620 0613 1
: 621 0614 1 ROUTINE VALUE:
: 622 0615 1 NONE
: 623 0616 1
: 624 0617 1 SIDE EFFECTS:
: 625 0618 1 NONE
: 626 0619 1
: 627 0620 1 --
: 628 0621 1
: 629 0622 2 BEGIN
: 630 0623 2
: 631 0624 2 LOCAL
: 632 0625 2 CURRENT_SIZE,
: 633 0626 2 STATUS;
: 634 0627 2
: 635 0628 2
: 636 0629 2 For short files or for the start of a file, group size may be shorter
: 637 0630 2 than the default.
: 638 0631 2
: 639 0632 2 IF .STARTING_BLOCK EQL 1
: 640 0633 2 THEN
: 641 0634 2 CURRENT_SIZE=.LAST_BLOCK*512
: 642 0635 2 ELSE
: 643 0636 2 CURRENT_SIZE=GROUP_SIZE;
: 644 0637 2
: 645 0638 2
: 646 0639 2 Default that failures will not be data sensitive.
: 647 0640 2
: 648 0641 2 READ_FAIL=FALSE;
: 649 0642 2
: 650 0643 2
: 651 0644 2 Group failure may be data sensitive. Read/write the original data several
: 652 0645 2 times before passing to write/read testing.
: 653 0646 2

```

```

: 654 0647 2 : Note that for DSA disks, writing the block could clear up the
: 655 0648 2 : SSS_FORCEDERROR encountered during the read.
: 656 0649 2 :
: 657 0650 2 : A double buffering technique is used for the read/write buffers. A boolean
: 658 0651 2 : flag is used to indicate which buffer is to be written from and the other
: 659 0652 2 : is to be read into.
: 660 0653 2 :
: 661 0654 2 : For reference: following is the section of code that previously performed
: 662 0655 2 : the data-sensitive test (reading the block a number of times).
: 663 0656 2 :
: 664 0657 2 : INCR TEST_INDEX FROM 1 TO TRIALS_TO_SUC DO
: 665 0658 2 : IF NOT (STATUS=DO_QIOW(IOS_READVBLK+IOSM_INHRETRY,DISK_TEXT,.CURRENT_SIZE,.STARTING_BLOCK,0,0))
: 666 0659 2 : THEN
: 667 0660 2 : BEGIN
: 668 0661 2 : READ FAIL=TRUE;
: 669 0662 2 : RETURN .STATUS
: 670 0663 2 : END;
: 671 0664 2 :
: 672 0665 2 :
: 673 0666 3 : BEGIN ! data-sensitive test block
: 674 0667 3 : LOCAL ! boolean flag
: 675 0668 3 : FLAG;
: 676 0669 3 : FLAG = TRUE; ! init flag
: 677 0670 3 :
: 678 0671 3 : DO_QIOW(IOS_READVBLK+IOSM_INHRETRY,BUF1,.CURRENT_SIZE,.STARTING_BLOCK,0,0);
: 679 0672 3 :
: 680 0673 3 : INCR TEST_INDEX FROM 1 TO TRIALS_TO_SUC DO
: 681 0674 3 : BEGIN
: 682 0675 4 : STATUS = DO_QIOW(IOS_WRITEVBLK+IOSM_INHRETRY,
: 683 0676 4 : (IF .FLAG THEN BUF1
: 684 0677 4 : ELSE BUF2),
: 685 0678 4 : .CURRENT_SIZE,
: 686 0679 5 : .STARTING_BLOCK,0,0);
: 687 0680 4 :
: 688 0681 4 : IF NOT .STATUS
: 689 0682 4 : THEN
: 690 0683 4 : BEGIN
: 691 0684 4 : READ FAIL = TRUE;
: 692 0685 5 : RETURN .STATUS;
: 693 0686 5 : END;
: 694 0687 5 :
: 695 0688 4 : STATUS = DO_QIOW(IOS_READVBLK+IOSM_INHRETRY,
: 696 0689 4 : (IF .FLAG THEN BUF2
: 697 0690 4 : ELSE BUF1),
: 698 0691 5 : .CURRENT_SIZE,
: 699 0692 4 : .STARTING_BLOCK,0,0);
: 700 0693 4 :
: 701 0694 4 : IF NOT .STATUS
: 702 0695 4 : THEN
: 703 0696 4 : BEGIN
: 704 0697 5 : READ FAIL = TRUE;
: 705 0698 5 : RETURN .STATUS;
: 706 0699 5 : END;
: 707 0700 4 :
: 708 0701 4 : IF CH$NEQ (.CURRENT_SIZE, BUF1, .CURRENT_SIZE, BUF2)
: 709 0702 4 : THEN
: 710 0703 4 :

```

```

: 711      0704  S      BEGIN
: 712      0705  S      READ FAIL = TRUE;
: 713      0706  S      RETURN SSS_PARITY;
: 714      0707  S      END;
: 715      0708  S
: 716      0709  S      FLAG = NOT .FLAG;
: 717      0710  S
: 718      0711  S      END;
: 719      0712  S      END;
: 720      0713  S      ! end of INCR loop
: 721      0714  S      ! end of data-sensitive block
: 722      0715  S      !
: 723      0716  S      ! Group must pass write/read test multiple times before being considered good.
: 724      0717  S      INCR TEST_INDEX FROM 1 TO TRIALS_TO_SUC DO
: 725      0718  S      BEGIN
: 726      0719  S      !
: 727      0720  S      ! Write to the indicated disk block.
: 728      0721  S
: 729      0722  S      IF NOT (STATUS=DO_QIOW(IOS_WRITEVBLK+IOSM_INHRETRY, GROUP_TEST_DATA, .CURRENT_SIZE, .STARTING_BLOCK, 0, 0))
: 730      0723  S      THEN
: 731      0724  S      RETURN .STATUS;
: 732      0725  S
: 733      0726  S      !
: 734      0727  S      ! Try and read it back.
: 735      0728  S
: 736      0729  S      IF NOT (STATUS=DO_QIOW(IOS_READVBLK+IOSM_INHRETRY, DISK_TEXT, .CURRENT_SIZE, .STARTING_BLOCK, 0, 0))
: 737      0730  S      THEN
: 738      0731  S      RETURN .STATUS;
: 739      0732  S
: 740      0733  S      !
: 741      0734  S      ! Make sure its the same.
: 742      0735  S
: 743      0736  S      IF CH$NEQ(.CURRENT_SIZE, GROUP_TEST_DATA, .CURRENT_SIZE, DISK_TEXT)
: 744      0737  S      THEN
: 745      0738  S      RETURN SSS_PARITY
: 746      0739  S      END;
: 747      0740  S
: 748      0741  S      RETURN TRUE;
: 749      0742  S
: 750      0743  S      END;

```

03FC 0000 GROUP_BLOCKTEST:

					.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9	: 0590
	59	0000V	CF	9E	00002	MOVAB	DO_QIOW, R9
	58	0000'	CF	9E	00007	MOVAB	STARTING_BLOCK, R8
	0'		68	D1	0000C	CMPL	STARTING_BLOCK, #1
			07	12	0000F	BNEQ	1\$
54	04	A8	09	78	00011	ASHL	#9, LAST_BLOCK, CURRENT_SIZE
			05	11	00016	BRB	2\$
	54	1E0.	8F	3C	00018	MOVZWL	#7680, CURRENT_SIZE
		F4	A8	D4	0001D	CLRL	READ_FAIL
	55		01	D0	00020	MOVL	#1, FLAG
			7E	7C	00023	CLRQ	-(SP)
							: 0632
							: 0634
							: 0636
							: 0641
							: 0671
							: 0673

			64	DD	00025	PUSHL	STARTING_BLOCK		
			54	DD	00027	PUSHL	CURRENT_SIZE		
		ASF4	C8	9F	00029	PUSHAB	BUF1		
		8031	8F	3C	0002D	MOVZWL	#32817, -(SP)		
7E			06	FB	00032	CALLS	#6, DO_QIOW		
69			01	DD	00035	MOVL	#1, TEST_INDEX		0675
57			7E	7C	00038	3\$: CLRQ	-(SP)		0678
			68	DD	0003A	PUSHL	STARTING_BLOCK		0682
			54	DD	0003C	PUSHL	CURRENT_SIZE		0681
07			55	E9	0003E	BLBC	FLAG, 4\$		0679
50		ASF4	C8	9E	00041	MOVAB	BUF1, R0		
			05	11	00046	BRB	5\$		
50		C3F4	C8	9E	00048	4\$: MOVAB	BUF2, R0		
			5C	DD	0004D	5\$: PUSHL	R0		
7E		8030	8F	3C	0004F	MOVZWL	#32816, -(SP)		0678
69			06	FB	00054	CALLS	#6, DO_QIOW		
56			50	DD	00057	MOVL	R0, STATUS		
25			56	E9	0005A	BLBC	STATUS, 8\$		0683
			7E	7C	0005D	CLRQ	-(SP)		0690
			68	DD	0005F	PUSHL	STARTING_BLOCK		0694
			54	DD	00061	PUSHL	CURRENT_SIZE		0693
07			55	E9	00063	BLBC	FLAG, 6\$		
50		C3F4	C8	9E	00066	MOVAB	BUF2, R0		0691
			05	11	0006B	BRB	7\$		
50		ASF4	C8	9E	0006D	6\$: MOVAB	BUF1, R0		
			50	DD	00072	7\$: PUSHL	R0		
7E		8031	8F	3C	00074	MOVZWL	#32817, -(SP)		0690
69			06	FB	00079	CALLS	#6, DO_QIOW		
56			50	DD	0007C	MOVL	R0, STATUS		
06			56	E8	0007F	BLBS	STATUS, 9\$		0695
F4		A8	01	DD	00082	8\$: MOVL	#1, READ_FAIL		0698
			4A	11	00086	BRB	12\$		0699
C3F4	C8	ASF4	C8	29	00088	9\$: CMPC3	CURRENT_SIZE, BUF1, BUF2		0702
			06	13	00090	BEQL	10\$		
		F4	A8	01	DD	00092	MOVL	#1, READ_FAIL	0705
			48	11	00096	BRB	11\$		0706
			55	D2	00098	10\$: MCOML	FLAG, FLAG		0709
		99	03	F3	0009B	AOBLEQ	#3, TEST_INDEX, 3\$		0675
			01	DD	0009F	MOVL	#1, TEST_INDEX		0717
			7E	7C	000A2	11\$: CLRQ	-(SP)		0722
			68	DD	000A4	PUSHL	STARTING_BLOCK		
			54	DD	000A6	PUSHL	CURRENT_SIZE		
		E1F4	C8	9F	000A8	PUSHAB	GROUP TEST DATA		
7E		8030	8F	3C	000AC	MOVZWL	#32816, -(SP)		
69			06	FB	000B1	CALLS	#6, DO_QIOW		
56			50	DD	000B4	MOVL	R0, STATUS		
18			56	E9	000B7	BLBC	STATUS, 12\$		
			7E	7C	000BA	CLRQ	-(SP)		0729
			68	DD	000BC	PUSHL	STARTING_BLOCK		
			54	DD	000BE	PUSHL	CURRENT_SIZE		
		ASF4	C8	9F	000C0	PUSHAB	DISK TEXT		
7E		8031	8F	3C	000C4	MOVZWL	#32817, -(SP)		
69			06	FB	000C9	CALLS	#6, DO_QIOW		
56			50	DD	000CC	MOVL	R0, STATUS		
04			56	E8	000CF	BLBS	STATUS, 13\$		
50			56	DD	000D2	12\$: MOVL	STATUS, R0		0731
			04	000D5	RET				


```

: 752 0744 1 ROUTINE DO_QIOW (FUNCTIONAL ,P5) =
: 753 0745 1
: 754 0746 1 ++
: 755 0747 1 FUNCTIONAL DESCRIPTION:
: 756 0748 1
: 757 0749 1 Common routine for performing $QIOW system service.
: 758 0750 1
: 759 0751 1 CALLING SEQUENCE:
: 760 0752 1 CALL DO_QIOW (ARG1,ARG2,ARG3,ARG4,ARG5,ARG6)
: 761 0753 1
: 762 0754 1 FORMAL PARAMETERS:
: 763 0755 1 ARG1: QIOW function code
: 764 0756 1 ARG2: The address of the P1 parameter
: 765 0757 1 ARG3: The address of the P2 parameter
: 766 0758 1 ARG4: The address of the P3 parameter
: 767 0759 1 ARG5: The address of the P4 parameter
: 768 0760 1 ARG6: The address of the P5 parameter
: 769 0761 1
: 770 0762 1 IMPLICIT INPUTS:
: 771 0763 1 CHANNEL: The channel number to the FILES ACP
: 772 0764 1 IOSB: The I/O status block
: 773 0765 1
: 774 0766 1 IMPLICIT OUTPUTS:
: 775 0767 1 NONE
: 776 0768 1
: 777 0769 1 ROUTINE VALUE:
: 778 0770 1 The system service code for the $QIOW
: 779 0771 1
: 780 0772 1 SIDE EFFECTS:
: 781 0773 1 NONE
: 782 0774 1
: 783 0775 1 --
: 784 0776 1
: 785 0777 2 BEGIN
: 786 0778 2 LOCAL
: 787 0779 2 STATUS;
: 788 0780 2
: 789 0781 2
: 790 0782 2 Do QIOW and check io service return.
: 791 0783 2
: 792 P 0784 2 IF NOT (STATUS = $QIOW( CHAN = .CHANNEL,
: 793 P 0785 2 IOSB = IOSB,
: 794 P 0786 2 FUNC = .FUNCTION,
: 795 P 0787 2 P1 = .P1,
: 796 P 0788 2 P2 = .P2,
: 797 P 0789 2 P3 = .P3,
: 798 P 0790 2 P4 = .P4,
: 799 0791 2 P5 = .P5))
: 800 0792 2 THEN
: 801 0793 2 RETURN .STATUS;
: 802 0794 2
: 803 0795 2
: 804 0796 2 Check I/O completion return.
: 805 0797 2
: 806 0798 2 IF NOT .IOSB[0]
: 807 0799 2 THEN
: 808 0800 2 RETURN .IOSB[0]

```

```

: 809      0801 2 ELSE
: 810      0802 2     RETURN SSS_NORMAL;
: 811      0803 2
: 812      0804 1 END;

```

```

                                .EXTRN  SYSSQIOW
                                DO_GIOW: .WORD  Save R2
                                MOVAB   IOSB, R2
                                CLRL    -(SP)
                                MOVQ    P4, -(SP)
                                MOVQ    P2, -(SP)
                                PUSHL   P1
                                CLRQ   -(SP)
                                PUSHL   R2
                                PUSHL   FUNCTION
                                MOVZWL  CHANNEL, -(SP)
                                CLRL    -(SP)
                                CALLS   #12, SYSSQIOW
                                BLBC    STATUS, 2$
                                BLBS   IOSB, 1$
                                MOVZWL  IOSB, R0
                                RET
                                50      01  D0 00033 1$: MOVL  #1, R0
                                04      04  00036 2$: RET

```

: Routine Size: 55 bytes, Routine Base: \$CODE\$ + 02F5

: 0744
: 0791
: 0798
: 0802
: 0804


```

0000 00000
50 0000' CF 9E 00002
51 01E0 CO 9E 00007
51 50 D1 0000C 1$:
09 13 0000F
80 EB6DB6DB 8F D0 00011
F2 11 00018
04 0001A 2$:

```

```

.ENTRY DATA INIT, Save nothing
MOVAB GROUP TEST DATA, POINTER
MOVAB 480(R0), END_POINTER
CMLP POINTER, END_POINTER
BEQL 2$
MOVL #-345131301, (POINTER)+
BRB 1$
RET

```

```

: 0805
: 0842
: 0843
: 0848
:
: 0850
: 0851
: 0856

```

: Routine Size: 27 bytes, Routine Base: \$CODE\$ + 032C

: 866 0857 i

```

868 0858 1 ROUTINE CHECK_BADSTATUS (STATUS) =
869 0859 1
870 0860 1 ++
871 0861 1 FUNCTIONAL DESCRIPTION:
872 0862 1
873 0863 1 Routine classifies the system service codes that it receives
874 0864 1 as input into 3 categories:
875 0865 1
876 0866 1 NORMAL_STS : SSS NORMAL
877 0867 1 BAD_STS : device error indicating a bad block
878 0868 1 ERROR_STS : unrecoverable device error
879 0869 1
880 0870 1 CALLING SEQUENCE:
881 0871 1 CALL CHECK_BADSTATUS (ARG1)
882 0872 1
883 0873 1 FORMAL PARAMETERS:
884 0874 1 ARG1: A system service code
885 0875 1
886 0876 1 IMPLICIT INPUTS:
887 0877 1 NONE
888 0878 1
889 0879 1 IMPLICIT OUTPUTS:
890 0880 1 NONE
891 0881 1
892 0882 1 ROUTINE VALUE:
893 0883 1 Returns as a value one of the 3 above mentioned codes:
894 0884 1 NORMAL_STS, ERROR_STS or BAD_STS
895 0885 1
896 0886 1 COMPLETION CODES:
897 0887 1 NONE
898 0888 1
899 0889 1 SIDE EFFECTS:
900 0890 1 NONE
901 0891 1
902 0892 1 --
903 0893 1
904 0894 2 BEGIN
905 0895 2
906 0896 2
907 0897 2 Possible I/O codes are divided into three cases:
908 0898 2 good blocks, bad blocks, and severe device errors.
909 0899 2
910 0900 2 SELECTONE .STATUS OF
911 0901 2 SET
912 0902 2
913 0903 2 [SSS_NORMAL] : RETURN NORMAL_STS;
914 0904 2
915 0905 2 [SSS_PARITY,
916 0906 2 SSS_DATACHECK,
917 0907 2 SSS_FORMAT,
918 0908 2 SSS_FORCEDERROR,
919 0909 2 SSS_CTRLERR,
920 0910 2 SSS_DRVERR] : RETURN BAD_STS;
921 0911 2
922 0912 2 [OTHERWISE] : RETURN ERROR_STS
923 0913 2
924 0914 2 TES;

```

: 925 0915 1 END;

```

                                0000 00000 CHECK_BADSTATUS:
                                .WORD  Save nothing
                                50      04  AC  D0 00002      MOVL  STATUS, R0      : 0858
                                01      50  D1 00006      CMPL  R0, #1         : 0900
                                03      12 00009      BNEQ  1$           : 0903
                                50      04 00008      CLRL  R0
                                04      04 0000D      RET
00000054 8F      50  D1 0000E 1$:  CMPL  R0, #84         : 0905
0000005C 8F      2D  13 00015      BEQL  2$
0000008C 8F      50  D1 00017      CMPL  R0, #92
0000008C 8F      24  13 0001E      BEQL  2$
0000008C 8F      50  D1 00020      CMPL  R0, #140
0000008C 8F      1B  13 00027      BEQL  2$
0000008C 8F      50  D1 00029      CMPL  R0, #188
000001F4 8F      12  13 00030      BEQL  2$
00002144 8F      50  D1 00032      CMPL  R0, #500
00002144 8F      09  13 00039      BEQL  2$
                                50      50  D1 0003B      CMPL  R0, #8516
                                04      12 00042      BNEQ  3$
                                50      02  D0 00044 2$:  MOVL  #2, R0         : 0910
                                04      04 00047      RET
                                50      01  D0 00048 3$:  MOVL  #1, R0         : 0912
                                04      04 00048      RET                               : 0915

```

: Routine Size: 76 bytes, Routine Base: \$CODE\$ + 0347

```

: 927 0916 1 ROUTINE NORMAL_COMPLETE : NOVALUE =
: 928 0917 1
: 929 0918 1 **
: 930 0919 1 FUNCTIONAL DESCRIPTION:
: 931 0920 1
: 932 0921 1     Called after entire file has been scanned for bad blocks.
: 933 0922 1     Any of the file remaining is good and should be returned
: 934 0923 1     to the volume. File is deleted and deaccessed.
: 935 0924 1
: 936 0925 1 CALLING SEQUENCE:
: 937 0926 1     CALL NORMAL_COMPLETE
: 938 0927 1
: 939 0928 1 FORMAL PARAMETERS:
: 940 0929 1     NONE
: 941 0930 1
: 942 0931 1 IMPLICIT INPUTS:
: 943 0932 1     FIB: File identification of current file.
: 944 0933 1
: 945 0934 1 IMPLICIT OUTPUTS:
: 946 0935 1     NONE
: 947 0936 1
: 948 0937 1 ROUTINE VALUE:
: 949 0938 1     NONE
: 950 0939 1
: 951 0940 1 COMPLETION CODES:
: 952 0941 1     NONE
: 953 0942 1
: 954 0943 1 SIDE EFFECTS:
: 955 0944 1     NONE
: 956 0945 1
: 957 0946 1 --
: 958 0947 1
: 959 0948 2 BEGIN
: 960 0949 2 LOCAL
: 961 0950 2     STATUS;
: 962 0951 2
: 963 0952 2
: 964 0953 2     Truncate any of the file that remains. If the truncation is successful,
: 965 0954 2     we delete the file and then deaccess it. Otherwise, we simply deaccess
: 966 0955 2     the file.
: 967 0956 2
: 968 0957 2
: 969 0958 2 STATUS=TRUNCATE(1);
: 970 0959 2
: 971 0960 2 IF (.STATUS EQL SSS_NORMAL)
: 972 0961 2 OR (.STATUS EQL SSS_ENDOFFILE)
: 973 0962 2 THEN
: 974 0963 2     |
: 975 0964 2     | Delete the file.
: 976 0965 2     |
: 977 0966 2     | DO_QIOW( IOS_DELETE+IOSM_DELETE,FIB_DESC,0,0,0,0);
: 978 0967 2     |
: 979 0968 2     |
: 980 0969 2     | Deaccess the file.
: 981 0970 2     |
: 982 0971 2     | DO_QIOW( IOS_DEACCESS,FIB_DESC,0,0,0,0);
: 983 0972 2

```



```
: 984      0973 2 |  
: 985      0974 2 | Remove Pending Bad Block entries for this file from the Pending Bad Block  
: 986      0975 2 | file ([0,0]BADLOG.SYS).  
: 987      0976 2 |  
: 988      0977 2 REMOVE_PBB (FIB[FIB$W_FID]);  
: 989      0978 2 |  
: 990      0979 2 RETURN  
: 991      0980 1 END;
```

```
                                0000 00000 NORMAL_COMPLETE:  
                                .WORD Save nothing  
                                PUSHL #1  
                                CALLS #1, TRUNCATE  
                                CMPL STATUS, #1  
                                BEQL 1$  
                                CMPL STATUS, #2160  
                                BNEQ 2$  
                                CLRQ -(SP)  
                                CLRQ -(SP)  
                                PUSHAB FIB_DESC  
                                MOVZWL #309, -(SP)  
                                CALLS #6, DO_QIOW  
                                CLRQ -(SP)  
                                CLRQ -(SP)  
                                PUSHAB FIB_DESC  
                                PUSHL #52  
                                CALLS #6, DO_QIOW  
                                PUSHAB FIB+4  
                                CALLS #1, REMOVE_PBB  
                                RET  
                                : 0916  
                                : 0958  
                                : 0960  
                                : 0961  
                                : 0966  
                                : 0971  
                                : 0977  
                                : 0980
```

: Routine Size: 66 bytes, Routine Base: \$CODE\$ + 0393

: 992 0981 1

```

: 994      0982 1 ROUTINE ERROR_COMPLETE : NOVALUE =
: 995      0983 1
: 996      0984 1 :++
: 997      0985 1 : FUNCTIONAL DESCRIPTION:
: 998      0986 1
: 999      0987 1 :     This routine is called when a fatal device error or system service
1000     0988 1 :     error is encountered during processing. The current file is truncated
1001     0989 1 :     to zero blocks and deaccessed.
1002     0990 1
1003     0991 1 : CALLING SEQUENCE:
1004     0992 1 :     CALL ERROR_COMPLETE
1005     0993 1
1006     0994 1 : FORMAL PARAMETERS:
1007     0995 1 :     NONE
1008     0996 1
1009     0997 1 : IMPLICIT INPUTS:
1010     0998 1 :     FIB: file identification of current file.
1011     0999 1
1012     1000 1 : IMPLICIT OUTPUTS:
1013     1001 1 :     NONE
1014     1002 1
1015     1003 1 : ROUTINE VALUE:
1016     1004 1 :     NONE
1017     1005 1
1018     1006 1 : SIDE EFFECTS:
1019     1007 1 :     NONE
1020     1008 1
1021     1009 1 : --
1022     1010 1
1023     1011 2 BEGIN
1024     1012 2
1025     1013 2 :
1026     1014 2 : Truncate the file to zero blocks.
1027     1015 2
1028     1016 2 TRUNCATE (1);
1029     1017 2
1030     1018 2 :
1031     1019 2 : Deaccess the file.
1032     1020 2
1033     1021 2 DO_QIOW(10$,DEACCESS,FIB_DESC,0,0,0,0);
1034     1022 2
1035     1023 2 RETURN
: 1036     1024 1 END;

```

```

                                0000 00000 ERROR_COMPLETE:
                                .WORD   Save nothing
                                PUSHL   #1
                                CALLS   #1, TRUNCATE
                                CLRQ    -(SP)
                                CLRQ    -(SP)
                                PUSHAB  FIB_DESC
                                PUSHL   #52
                                CALLS   #6, DO_QIOW
FCCB   CF           01 DD 00002
                                01 FB 00004
                                7E 7C 00009
                                7E 7C 0000B
                                0000' CF 9F 0000D
                                FF08 CF 34 DD 00011
                                06 FB 00013

```

```

: 0982
: 1016
: 1021
:
:
:
:

```

SCANFILE
V04-00i

D 4
8-Jan-1985 17:22:51
7-Nov-1984 11:22:13

VAX-11 Bliss-32 V4.0-742
[BADBLK.BUGSRC]SCANFILE.B32;2

Page 33
(13)
; 1024

04 00018 RET

; Routine Size: 25 bytes. Routine Base: \$CODE\$ + 03D5

_\$2

Sym

DCL

```

1038 1025 1 ROUTINE GROUP_RETURN =
1039 1026 1 ++
1040 1027 1 FUNCTIONAL DESCRIPTION:
1041 1028 1
1042 1029 1     Called when a bad block error is encountered by group
1043 1030 1     block testing. The individual blocks in a group are
1044 1031 1     tested for "badness" and truncated off the current file
1045 1032 1     and into the bad block file when found.
1046 1033 1
1047 1034 1 CALLING SEQUENCE:
1048 1035 1     CALL GROUP_RETURN
1049 1036 1
1050 1037 1 FORMAL PARAMETERS:
1051 1038 1     NONE
1052 1039 1
1053 1040 1 IMPLICIT INPUTS:
1054 1041 1     STARTING_BLOCK: First block in group
1055 1042 1     LAST_BLOCK: Last block in group
1056 1043 1
1057 1044 1 IMPLICIT OUTPUTS:
1058 1045 1     NONE
1059 1046 1
1060 1047 1 ROUTINE VALUE:
1061 1048 1     NONE
1062 1049 1
1063 1050 1 SIDE EFFECTS:
1064 1051 1     NONE
1065 1052 1
1066 1053 1 --
1067 1054 1
1068 1055 2 BEGIN
1069 1056 2 LOCAL
1070 1057 2     VBN;
1071 1058 2
1072 1059 2
1073 1060 2     Individually consider all blocks in the group.
1074 1061 2     Return each to the badblock file or free space.
1075 1062 2
1076 1063 2     VBN=.LAST_BLOCK;
1077 1064 2
1078 1065 2 WHILE TRUE DO
1079 1066 2     BEGIN
1080 1067 2
1081 1068 2     CASE CHECK_BADSTATUS(BLOCKTEST(.VBN))
1082 1069 2     FROM NORMAL_STS TO BAD_STS OF
1083 1070 2     SET
1084 1071 2
1085 1072 2         [NORMAL_STS] : TRUNC_BLOCK=.VBN;
1086 1073 2
1087 1074 2         [ERROR_STS] : RETURN FALSE;
1088 1075 2
1089 1076 2         [BAD_STS] : TRUNCATE_BAD(.VBN);
1090 1077 2
1091 1078 2     TES;
1092 1079 2
1093 1080 2     VBN=.TRUNC_BLOCK-1;
1094 1081 2     IF .VBN LSS .STARTING_BLOCK

```

```

: 1095      1082 3 THEN
: 1096      1083 3 RETURN TRUE;
: 1097      1084 3
: 1098      1085 3 END
: 1099      1086 3
: 1100      1087 1 END;

```

! end of while true loop

```

                                000C 0000 GROUP_RETURN:
                                .WORD      Save R2,R3
                                MOVAB     TRUNC_BLOCK, R3
                                MOVL      LAST_BLOCK, VBN
                                PUSHL     VBN
                                CALLS     #1, BLOCKTEST
                                PUSHL     R0
                                CALLS     #1, CHECK_BADSTATUS
                                CASEL     R0, #0, #2
                                .WORD     3$-2$,-
                                .WORD     4$-2$,-
                                .WORD     5$-2$,-
                                MOVL      VBN, TRUNC_BLOCK
                                BRB       6$
                                CLRL      R0
                                RET
                                PUSHL     VBN
                                CALLS     #1, TRUNCATE_BAD
                                SUBL3     #1, TRUNC_BLOCK, VBN
                                CMPL     VBN, STARTING_BLOCK
                                BGEQ      1$
                                MOVL     #1, R0
                                RET
                                : 1025
                                : 1063
                                : 1068
                                :
                                :
                                :
                                :
                                :
                                : 1072
                                : 1074
                                : 1076
                                : 1080
                                : 1081
                                : 1083
                                : 1087

```

: Routine Size: 64 bytes, Routine Base: \$CODE\$ + 03EE

: 1101 1088 1

```

: 1103      1089 1 ROUTINE REMOVE_PBB (FID) : NOVALUE =
: 1104      1090 1
: 1105      1091 1  **
: 1106      1092 1  FUNCTIONAL DESCRIPTION:
: 1107      1093 1
: 1108      1094 1      This routine removes all pending bad block entries for a given
: 1109      1095 1      File ID from the [0,0]BADLOG.SYS file.
: 1110      1096 1
: 1111      1097 1  CALLING SEQUENCE:
: 1112      1098 1      CALL REMOVE_PBB (ARG1)
: 1113      1099 1
: 1114      1100 1  FORMAL PARAMETERS:
: 1115      1101 1      ARG1:  File ID of the desired file
: 1116      1102 1
: 1117      1103 1  IMPLICIT INPUTS:
: 1118      1104 1      NONE
: 1119      1105 1
: 1120      1106 1  IMPLICIT OUTPUTS:
: 1121      1107 1      NONE
: 1122      1108 1
: 1123      1109 1  ROUTINE VALUE:
: 1124      1110 1      NONE
: 1125      1111 1
: 1126      1112 1  SIDE EFFECTS:
: 1127      1113 1      Volume suspected bad block list altered
: 1128      1114 1  --
: 1129      1115 1
: 1130      1116 2 BEGIN
: 1131      1117 2
: 1132      1118 2 MAP
: 1133      1119 2     FID : REF BLOCK [, BYTE];
: 1134      1120 2
: 1135      1121 2 OWN
: 1136      1122 2     BADLOG_FIB : BLOCK [FIBSC_LENGTH, BYTE],      ! FIB for [0,0]BADLOG.SYS
: 1137      1123 2     BADLOG_BUF  : BLOCK [512, BYTE];           ! I/O buffer
: 1138      1124 2
: 1139      1125 2 BIND
: 1140      1126 2     BADLOG_FIB_DESC = UPLIT (FIBSC_LENGTH,BADLOG_FIB);
: 1141      1127 2
: 1142      1128 2 LITERAL
: 1143      1129 2     BADLOG_FID = 9;                               ! define file ID for BADLOG.SYS
: 1144      1130 2
: 1145      1131 2 LABEL
: 1146      1132 2     SEARCH_LOOP;
: 1147      1133 2
: 1148      1134 2 LOCAL
: 1149      1135 2     PBB : REF BLOCK [, BYTE],
: 1150      1136 2     WRITE_FLAG,
: 1151      1137 2     STATUS,
: 1152      1138 2     VBN,
: 1153      1139 2     J;
: 1154      1140 2
: 1155      1141 2
: 1156      1142 2     Initialize FIB for BADLOG.SYS
: 1157      1143 2
: 1158      1144 2 CH$FILL (0, FIBSC_LENGTH, BADLOG_FIB);
: 1159      1145 2 BADLOG_FIB [FIB$W_FID_NUM] = BADLOG_FID;           ! initialize file number

```


.PSECT \$PLITS,NOWRT,NOEXE,2

00000040 00008 P.AAB: .LONG 64
00000000 0000C .ADDRESS BADLOG_FIB

PSECT \$OWNS,NOEXE,2

05A74 BADLOG_FIB:
.BLKB 64
05AB4 BADLOG_BUF:
.BLKB 512

BADLOG_FIB_DESC= P.AAB

.PSECT \$CODES,NOWRT,2

OFFC 00000 REMOVE_PBB:

0040	BF	00	5B	0000'	CF	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	1089	
			6E		00	2C	00007	MOVAB	BADLOG_BUF, R11	1144	
								MOVCS	#0, (SP), #0, #64, BADLOG_FIB		
			C4	00090009	AB	8F	D0	00010	MOVL	#589833, BADLOG_FIB+4	1145
			C1		AB	01	88	00018	BISB2	#1, BADLOG_FIB+T	1148
						7E	7C	0001C	CLRQ	-(SP)	1150
						7E	7C	0001E	CLRQ	-(SP)	
				0000'	CF	9F	00020	PUSHAB	BADLOG_FIB_DESC		
			FE9A	72	8F	9A	00024	MOVZBL	#114, -(SP)		
					06	FB	00028	CALLS	#6, DO Q10W		
					50	D0	0002D	MOVL	R0, STATUS		
					5A	E9	00030	BLBC	STATUS, 5\$	1153	
					57	D4	00033	CLRL	VBN	1157	
					59	D4	00035	CLRL	WRITE_FLAG	1163	
					57	D6	00037	INCL	VBN	1164	
					7E	7C	00039	CLRQ	-(SP)	1165	
					57	DD	00038	PUSHL	VBN	1168	
				0200	8F	3C	0003D	MOVZWL	#512, -(SP)	1165	
					5B	DD	00042	PUSHL	R11		
					31	DD	00044	PUSHL	#49		
			FE7C		06	FB	00046	CALLS	#6, DO Q10W		
					50	D0	0004B	MOVL	R0, STATUS		
					5A	E9	0004E	BLBC	STATUS, 4\$	1170	
					58	9E	00051	MOVAB	BADLOG_BUF, PBB	1174	
					56	D4	00054	CLRL	J	1176	
		68	04		06	29	00056	CMPC3	#6, @FID, (PBB)	1178	
					09	12	0005B	BNEQ	3\$		
		10	00		00	2C	0005D	MOVCS	#0, (SP), #0, #16, (PBB)	1181	
					68		00062				
					59	01	00063	MOVL	#1, WRITE_FLAG	1182	
					58	10	00066	ADDL2	#16, PBB	1184	
			E9		1F	F3	00069	AOBLEQ	#31, J, 2\$	1176	
					59	E9	0006D	BLBC	WRITE_FLAG, 1\$	1187	
					7E	7C	00070	CLRQ	-(SP)	1189	
					57	DD	00072	PUSHL	VBN	1192	

defi

defi

	7E	0200	8F	3C	00074	MOVZWL	#512, -(SP)	:	1189
			5B	DD	00079	PUSHL	R11	:	
			30	DD	0007B	PUSHL	#48	:	
FE45	CF		06	FB	0007D	CALLS	#6, DO_QIOW	:	
			B1	11	00082	BRB	1\$:	1161
			7E	7C	00084	4\$: CLRQ	-(SP)	:	1197
			7E	7C	00086	CLRQ	-(SP)	:	
		0000	CF	9F	00088	PUSHAB	BADLOG_FIB_DESC	:	
			34	DD	0008C	PUSHL	#52	:	
FE34	CF		06	FB	0008E	CALLS	#6, DO_QIOW	:	
			04	00093	5\$: RET			:	1202

: Routine Size: 148 bytes, Routine Base: \$CODE\$ + 042E

: 1217 1203 1
: 1218 1204 1 END ! end of module
: 1219 1205 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	23732	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$SPLITS	16	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODE\$	1218	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Symbols -----		Pages Mapped	Processing Time
	Total	Loaded Percent		
_\$255\$DUA18:[SYSLIB]LIB.L32;1	18619	35 0	1000	00:01.9

COMMAND QUALIFIERS

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

: Size: 1218 code + 23748 data bytes
: Run Time: 00:22.4
: Elapsed Time: 01:15.8
: Lines/CPU Min: 3226
: Lexemes/CPU-Min: 9692

DCLI
defir
defir
defir
defir
defir
defir
defir
defir

SCANFILE
V04-001

K 4
8-Jan-1985 17:22:51

VAX-11 Bliss-32 v4.0-742

Page 40

: Memory Used: 113 pages
: Compilation Complete

DCLII

defir

defir

defir

defir

defir

defir

defir

defir

defir

defir

