

```

RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSSSSSS
RRR      RRR      MMMMMM      MMMMMM      SSS
RRR      RRR      MMMMMM      MMMMMM      SSS
RRR      RRR      MMMMMM      MMMMMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSS
RRRRRRRRRRRRRRR      MMM      MMM      SSSSSSSSSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      MMM      SSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSSS
RRR      RRR      MMM      MMM      SSSSSSSSSSSSSS

```

— 52

Syn

NTS

NTS

NTS

1997

NTS

NTS

NTS

NTS

NTS

NTS

NTS
NTSNTS
NTSNTS
NTSNTS
NTS

NTS

NTS

NTS
NTSNTS
NTSNTS
NTS

NYT

10

NTS

NTS
NTSNT
NT

NI
PI

10

1

1

10

100

[illegible]

G 4
16-Sep-1984 01:58:54
14-Sep-1984 13:01:37

VAX-11 Bliss-32 V4.0-742
[RMS.SRC]RM3PUTERR.B32;1

Page 1
(1)

**F

```
0001 0
0002 0 MODULE RM3PUTERR (LANGUAGE (BLISS32) ,
0003 0 IDENT = 'V04-000'
0004 0 ) =
0005 1 BEGIN
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
0032 1 FACILITY: RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
0033 1
0034 1 ABSTRACT:
0035 1 $PUT and $UPDATE specific error cleanup routines
0036 1
0037 1
0038 1 ENVIRONMENT:
0039 1
0040 1 VAX/VMS OPERATING SYSTEM
0041 1
0042 1 --
0043 1
0044 1
0045 1 AUTHOR: Todd M. Katz CREATION DATE: 17-Jul-82
0046 1
0047 1
0048 1 Modified by:
0049 1
0050 1 V03-014 MCN0003 Maria del C. Nasr 04-Apr-1983
0051 1 Change linkage of RM$NULLKEY to RL$JSB.
0052 1
0053 1 V03-013 MCN0002 Maria del C. Nasr 15-Mar-1983
0054 1 More linkages reorganization
0055 1
0056 1 V03-012 MCN0001 Maria del C. Nasr 28-Feb-1983
0057 1 Reorganize linkages
```

```
58 0058 1 |
59 0059 1 | V03-011 TMK0004 Todd M. Katz 15-Feb-1983
60 0060 1 | If the deletion of the RRV fails in RM$PUTUPD_ERROR, do not
61 0061 1 | delete the primary data record completely, and then create a
62 0062 1 | pointerless RRV at the end of the bucket. This is what is
63 0063 1 | done currently. Just delete the primary data record by calling
64 0064 1 | RM$DELETE_UDR so that it is deleted according to the normal
65 0065 1 | rules for primary data record deletion. This does leave the
66 0066 1 | possibility of having a RRV point to nothing (if the RRV
67 0067 1 | deletion fails and the primary data record is completely
68 0068 1 | deleted), but such a occurrence would also exist if the
69 0069 1 | pointerless RRV were deleted as part of a CONVERT/RECLAIM.
70 0070 1 |
71 0071 1 | V03-010 TMK0003 Todd M. Katz 05-Jan-1983
72 0072 1 | The routine RM$PUTUPD_ERROR was saving, zeroing, and then
73 0073 1 | restoring the current NRP key of reference while all newly
74 0074 1 | added SIDs were being deleted. This is no longer necessary.
75 0075 1 |
76 0076 1 | V03-009 TMK0002 Todd M. Katz 19-Sep-1982
77 0077 1 | Add support for prologue 3 SIDs. This involves setting AP to
78 0078 1 | 3 instead of to 1 each time RM$RECORD_KEY is called to extract
79 0079 1 | the key of the SID which is to be located and deleted.
80 0080 1 |
81 0081 1 | V03-008 KBT0229 Keith B. Thompson 24-Aug-1982
82 0082 1 | Reorganize psects
83 0083 1 |
84 0084 1 | V03-007 TMK0001 Todd M. Katz 17-Jul-1982
85 0085 1 | Completely revised the routines in this module because of
86 0086 1 | changes in the routines they interface to.
87 0087 1 |
88 0088 1 |
89 0089 1 | *****
90 0090 1 |
91 0091 1 | LIBRARY 'RMSLIB:RMS';
92 0092 1 |
93 0093 1 | REQUIRE 'RMSSRC:RMSIDXDEF';
94 0158 1 |
95 0159 1 | ! Define default PSECTS for code
96 0160 1 |
97 0161 1 | PSECT
98 0162 1 | CODE = RM$RMS3(PSECT_ATTR),
99 0163 1 | PLIT = RM$RMS3(PSECT_ATTR);
100 0164 1 |
101 0165 1 | ! Linkages
102 0166 1 |
103 0167 1 | LINKAGE
104 0168 1 | L_ERROR_LINK1,
105 0169 1 | L_ERROR_LINK2,
106 0170 1 | L_JSB,
107 0171 1 | L_LINK_7_10_11,
108 0172 1 | L_RABREG_4587,
109 0173 1 | L_RABREG_67,
110 0174 1 | L_RABREG_7,
111 0175 1 | L_PRESERVE1;
112 0176 1 |
113 0177 1 | ! External Routines
114 0178 1 |
```

:	115	0179	1	EXTERNAL ROUTINE	:	RL\$RABREG_4567,
:	116	0180	1	RMSDELETE_RRV	:	RL\$RABREG_7,
:	117	0181	1	RMSDELETE_SIDR	:	RL\$RABREG_4567,
:	118	0182	1	RMSDELETE_UDR	:	RL\$RABREG_67,
:	119	0183	1	RMSFIND_BY_RRV	:	RL\$LINK_7-10-11,
:	120	0184	1	RMSGET_NEXT_KEY	:	RL\$RABREG_7,
:	121	0185	1	RMSKEY_DESC	:	RL\$JSB,
:	122	0186	1	RMSNOREAD_LONG	:	RL\$JSB,
:	123	0187	1	RMSNULLKEY	:	RL\$PRESERVE1,
:	124	0188	1	RMSRECORD_KEY	:	RL\$PRESERVE1;
:	125	0189	1	RMSRLSBKT	:	

```
127 0190 1 %SBTTL 'RM$PUTUPD_ERROR'
128 0191 1 GLOBAL ROUTINE RM$PUTUPD_ERROR : RL$ERROR_LINK2 NOVALUE =
129 0192 1
130 0193 1 ++
131 0194 1
132 0195 1 FUNCTIONAL DESCRIPTION:
133 0196 1
134 0197 1 This routine's responsibility is to delete SDR entires and the user
135 0198 1 data record on $PUT/$UPDATE errors.
136 0199 1
137 0200 1 If this routine is called with the index descriptor for the primary key
138 0201 1 of reference then all SDR entries are deleted, otherwise, the deletion
139 0202 1 of SDR entries begins with 1 less than the current index descriptor.
140 0203 1
141 0204 1 The user data record and any RRV associated with it are deleted only if
142 0205 1 the error occurred on a $PUT. The user data record will not be deleted
143 0206 1 if the error occurred on any type of a $UPDATE (a regular $UPDATE or a
144 0207 1 $PUT converted into a $UPDATE).
145 0208 1
146 0209 1 CALLING SEQUENCE:
147 0210 1
148 0211 1 RM$PUTUPD_ERROR()
149 0212 1
150 0213 1 INPUT PARAMETERS:
151 0214 1 NONE
152 0215 1
153 0216 1 IMPLICIT INPUTS:
154 0217 1
155 0218 1 IDX_DFN - address of index decriptor
156 0219 1 -IDX$B_DESC_NO - descriptor number (index into update buffer)
157 0220 1 -IDX$B_KEYREF - key of reference
158 0221 1 -IDX$W_MINRECSZ - minimum record size necessary to contain key
159 0222 1
160 0223 1 IFAB - address of IFAB
161 0224 1 -IFB$W_KBUFSZ - size of a keybuffer
162 0225 1 -IFB$B_NUM_KEYS - number of keys in the file
163 0226 1 -IFB$B_PLG_VER - prologue version of the file
164 0227 1
165 0228 1 IRAB - address of IRAB
166 0229 1 -IRB$B_CACHEFLGS - flags for bucket retrieval routines
167 0230 1 -IRB$L_CURBDB - address of current buffer descriptor block
168 0231 1 -IRB$L_KEYBUF - address of contiguous keybuffers
169 0232 1 -IRB$B_MODE - access mode of caller
170 0233 1 -IRB$L_NXTBDB - address of a BDB (used to hold RRV bucket BDB)
171 0234 1 -IRB$W_PUTUP_ID - ID of user data record
172 0235 1 -IRB$L_PUTUP_VBN - VBN of user data record
173 0236 1 -IRB$W_UDR_ID - ID of current primary data record
174 0237 1 -IRB$L_UDR_VBN - VBN of current primary data record
175 0238 1 -IRB$V_UPDATE - if set, current operation is an $UPDATE
176 0239 1 -IRB$L_UPDBUF - address of internal update buffer
177 0240 1
178 0241 1 RAB - address of RAB
179 0242 1 -RAB$L_RBF - record buffer containing user data record
180 0243 1 -RAB$W_RSZ - size of user data record
181 0244 1
182 0245 1 OUTPUT PARAMETERS:
183 0246 1 NONE
```

RMSPUTUPD_ERROR

```
184 0247 1 |
185 0248 1 | IMPLICIT OUTPUTS:
186 0249 1 |
187 0250 1 |     IRBSB_CACHEFLGS      - the bit CSH$V_LOCK will be set
188 0251 1 |
189 0252 1 | ROUTINE VALUE:
190 0253 1 |     NONE
191 0254 1 |
192 0255 1 | SIDE EFFECTS:
193 0256 1 |
194 0257 1 |     AP is trashed.
195 0258 1 |     All new SIDR entries inserted during the current operation before the
196 0259 1 |     error occurred are deleted as is the user data record and any RRV
197 0260 1 |     pointing to it if a new data record was inserted during the course
198 0261 1 |     of the operation before the error occurred (ie - the operation was
199 0262 1 |     a $PUT).
200 0263 1 |
201 0264 1 | --
202 0265 1 |
203 0266 2 | BEGIN
204 0267 2 |
205 0268 2 | EXTERNAL REGISTER
206 0269 2 |     R_IDX_DFN_STR,
207 0270 2 |     COMMON_RAB_STR;
208 0271 2 |
209 0272 2 | GLOBAL REGISTER
210 0273 2 |     R_REC_ADDR_STR;
211 0274 2 |
212 0275 2 | ! If the file allows secondary keys then delete any of those that had been
213 0276 2 | ! newly inserted before the error occurred.
214 0277 2 |
215 0278 2 | IF .IFAB[IFBSB_NUM_KEYS] GTRU 0
216 0279 2 | THEN
217 0280 2 |     BEGIN
218 0281 2 |
219 0282 2 |     LABEL
220 0283 2 |     ENTRY;
221 0284 2 |
222 0285 2 |     LOCAL
223 0286 2 |         KREF          : BYTE,
224 0287 2 |         SAVE_UDR_ID   : WORD,
225 0288 2 |         SAVE_UDR_VBN  : LONG;
226 0289 2 |
227 0290 2 |     ! The routine which is called to deleted each SIDR entry,
228 0291 2 |     ! RMSDELETE_SIDR, operates only on the current primary data record.
229 0292 2 |     ! However, this routine maybe called to delete SIDR entries of a record
230 0293 2 |     ! other than the current primary data record. Therefore, in order to
231 0294 2 |     ! make use of RMSDELETE_SIDR, RMS must fool it into believing that it
232 0295 2 |     ! is operating on the current primary data record. This is done by
233 0296 2 |     ! saving the RFA address of the current primary data record, if there
234 0297 2 |     ! is one, and replacing it with the RFA address of the user data record
235 0298 2 |     ! whose SIDR entries are to be deleted.
236 0299 2 |
237 0300 2 |     SAVE_UDR_VBN = .IRAB[IRBSL_UDR_VBN];
238 0301 2 |     SAVE_UDR_ID  = .IRAB[IRBSW_UDR_ID];
239 0302 2 |
240 0303 2 |     IRAB[IRBSL_UDR_VBN] = .IRAB[IRBSL_PUTUP_VBN];
```

```
241 0304 3 IRAB[IRB$W_UDR_ID] = .IRAB[IRB$W_PUTUP_ID];
242 0305
243 0306 ! The user data record whose $PUT/$UPDATE resulted in the error maybe
244 0307 ! found in the user's record buffer. The keys of the SDR entries to
245 0308 ! be deleted maybe extracted from it.
246 0309
247 0310 REC_ADDR = .RAB[RAB$L_RBF];
248 0311
249 0312 ! If this routine was called with the index descriptor for the primary
250 0313 ! key then all SDR entries for the user data record are deleted;
251 0314 ! otherwise, all SDR entries up until the entry that was being
252 0315 ! inserted when the error occurred are deleted.
253 0316
254 0317 KREF = .IDX_DFN[IDX$B_KEYREF];
255 0318 RM$KEY_DESC(0);
256 0319
257 0320 WHILE (RM$GET_NEXT_KEY()
258 0321 AND
259 0322 (.IDX_DFN[IDX$B_KEYREF] NEQU .KREF))
260 0323 DO
261 0324
262 0325 ! Delete each SDR entry that had been inserted before the error
263 0326 ! occurred.
264 0327
265 0328 4 ENTRY: BEGIN
266 0329 4
267 0330 4 BUILTIN
268 0331 4 AP;
269 0332 4
270 0333 4 ! Under the following circumstances, the SDR entry for the current
271 0334 4 ! index descriptor being processed will not be deleted:
272 0335 4
273 0336 4 1. The operation being performed when the error occurred was an
274 0337 4 $UPDATE and no new SDR was inserted for this key of reference.
275 0338 4 2. RMS does not have read access to the user's record buffer.
276 0339 4 3. No new SDR was inserted for this key of reference because the
277 0340 4 user's data record was too short to contain such a key.
278 0341 4
279 0342 5 IF (.IRAB[IRB$V_UPDATE]
280 0343 5 AND
281 0344 6 (NOT .BBLOCK[.IRAB[IRB$L_UPDBUF] + .IDX_DFN[IDX$B_DESC_NO],
282 0345 5 UPD$V_INS_NEW]))
283 0346 4 OR
284 0347 4 RM$NOREAD_LONG (.RAB[RAB$W_RSZ], .REC_ADDR, .IRAB[IRB$B_MODE])
285 0348 4 OR
286 0349 5 (.RAB[RAB$W_RSZ] LSSU .IDX_DFN[IDX$W_MINRECSZ])
287 0350 4 THEN
288 0351 4 LEAVE ENTRY;
289 0352 4
290 0353 4 ! Extract into keybuffer 2, the secondary key for the key of
291 0354 4 ! reference being processed from the user's record buffer. Check
292 0355 4 ! whether the key is null and only delete the SDR entry for this
293 0356 4 ! key of reference if it is not.
294 0357 4
295 0358 4 AP = 3;
296 0359 4
297 0360 5 BEGIN
```

```
298 0361 5 GLOBAL REGISTER
299 0362 5 R_BDB;
300 0363 5
301 0364 5 RMSRECORD_KEY (KEYBUF_ADDR(2));
302 0365 4 END;
303 0366 4
304 0367 4 AP = 1;
305 0368 4 IF RMSNULLKEY (KEYBUF_ADDR(2))
306 0369 4 THEN
307 0370 4 RMSDELETE_SIDR();
308 0371 4
309 0372 4 END;
310 0373 4
311 0374 4 ! Restore the RFA of the current primary data record (if there is one)
312 0375 4 ! to its corresponding location in the IRAB as part of the next record
313 0376 4 ! positioning context as RMS has finished deleting SIDR entries.
314 0377 4
315 0378 4 IRAB[IRBSL_UDR_VBN] = .SAVE_UDR_VBN;
316 0379 4 IRAB[IRBSW_UDR_ID] = .SAVE_UDR_ID;
317 0380 4 END;
318 0381 4
319 0382 4 ! If the error occurred during a $PUT, then a user data record was
320 0383 4 ! inserted before any SIDRs and must be deleted. If the error occurred
321 0384 4 ! during an $UPDATE then just the deletion of any new SIDR entries required
322 0385 4 ! by the $UPDATE is sufficient to restore the record to the state it
323 0386 4 ! occupied prior to the $UPDATE. None of the SIDR entries for the user data
324 0387 4 ! record existing in the file before the $UPDATE are deleted until all the
325 0388 4 ! new SIDR entries are inserted so there are no SIDR entries to re-insert,
326 0389 4 ! and of course, the user data record itself can never be deleted because
327 0390 4 ! it existed in the file prior to the $UPDATE.
328 0391 4
329 0392 4 NOTE that it is possible that RMS will also have to delete an RRV for this
330 0393 4 ! new user data record even though RRVs are never created during the
331 0394 4 ! insertion of a new primary data record. This is because RMS will release
332 0395 4 ! the primary data bucket containing the new record during index updates
333 0396 4 ! and SIDR entry insertions, and the action of some other stream may cause
334 0397 4 ! the bucket containing this new primary data bucket to split and an RRV
335 0398 4 ! created for it.
336 0399 4
337 0400 4 IF NOT .IRAB[IRBSV_UPDATE]
338 0401 4 THEN
339 0402 4 BEGIN
340 0403 4
341 0404 4 RMSKEY_DESC(0);
342 0405 4
343 0406 4 ! Attempt to position to the user data record, and delete it if able to
344 0407 4 ! successfully position to it. Perform the FIND_BY_RRV in a way such
345 0408 4 ! that if an RRV was created for the new primary data record between
346 0409 4 ! the time this stream released the primary data bucket and reclaims it
347 0410 4 ! below, the RRV bucket will be locked during the positioning to the
348 0411 4 ! user data record, and the address of the BDB for it placed in
349 0412 4 ! IRBSL_NXTBDB.
350 0413 4
351 0414 4 IRAB[IRBSB_CACHEFLGS] = CSHSM_LOCK;
352 0415 4 IRAB[IRBSL_NXTBDB] = 0;
353 0416 4
354 0417 4 IF RMSFIND_BY_RRV (.IRAB[IRBSL_PUTUP_VBN], .IRAB[IRBSW_PUTUP_ID], 1)
```

```

355      0418 3      THEN
356      0419 4      BEGIN
357      0420 4
358      0421 4      GLOBAL REGISTER
359      0422 4      R_BDB_STR,
360      0423 4      R_BKT_ADDR;
361      0424 4
362      0425 4      ! If the new user data record is found not to be in its original
363      0426 4      ! bucket, then the RRV for it must be deleted.
364      0427 4
365      0428 4      IF (BDB = .IRAB[IRB$L_NXTBDB]) NEQU 0
366      0429 4      THEN
367      0430 5      BEGIN
368      0431 5      IRAB[IRB$L_NXTBDB] = 0;
369      0432 5      RMSDELETE_RRV();
370      0433 4      END;
371      0434 4
372      0435 4      ! Delete the new user data record.
373      0436 4
374      0437 4      BDB = .IRAB[IRB$L_CURBDB];
375      0438 4      IRAB[IRB$L_CURBDB] = 0;
376      0439 4
377      0440 4      RMSDELETE_UDR();
378      0441 4
379      0442 4      ! Mark the primary data bucket that contained the new user data
380      0443 4      ! record dirty, and release it.
381      0444 4
382      0445 4      BDB[BDB$V_DRT] = 1;
383      0446 4      RMSRLSBKT(0);
384      0447 4
385      0448 3      END;
386      0449 3
387      0450 2      END;
388      0451 2
389      0452 1      END;
```

```

.TITLE RM3PUTERR
.IDENT \V04-000\

.EXTRN RMSDELETE_RRV, RMSDELETE_SIDR
.EXTRN RMSDELETE_UDR, RMSFIND_BY_RRV
.EXTRN RMSGET_NEXT_KEY
.EXTRN RMSKEY_DESC, RMSNOREAD_LONG
.EXTRN RMSNUL[KEY, RMSRECORD_KEY
.EXTRN RMSRLSBKT

.PSECT RMSRMS3,NOWRT, GBL, PIC,2
```

```

007D 8F BB 00000 RMSPUTUPD_ERROR:
00B2 CA 95 00004      PUSH  #^M<R0,R2,R3,R4,R5,R6>      : 0191
      03 12 00008      TSTB  178(IRAB)                  : 0278
      0091 31 0000A      BNEQ  1$
      55 00B0 C9 D0 0000D 1$:      BRW  5$
      52 00BC C9 B0 00012      MOVL  176(IRAB), SAVE_UDR_VBN      : 0300
00B0 C9 78 A9 D0 00017      MOVW  188(IRAB), SAVE_UDR_ID        : 0301
      78 A9 D0 00017      MOVL  120(IRAB), 176(IRAB)         : 0303
```

00BC	C9	0080	C9	B0	0001D	MOVW	128(IRAB), 188(IRAB)	0304
	56	28	A8	D0	00024	MOVL	40(RAB), REC_ADDR	0310
	53	21	A7	90	00028	MOVB	33(IDX_DFN), KREF	0317
			7E	D4	0002C	CLRL	-(SP)	0318
			0000G	30	0002E	BSBW	RM\$KEY_DESC	
	5E		04	C0	00031	ADDL2	#4, SP	
			0000G	30	00034	BSBW	RM\$GET_NEXT_KEY	0320
	5A		50	E9	00037	BLBC	R0, 4\$	
	53	21	A7	91	0003A	CMPB	33(IDX_DFN), KREF	0322
			54	13	0003E	BEQL	4\$	
0B	06		03	E1	00040	BBC	#3, 6(IRAB), 3\$	0342
	50	10	A7	9A	00045	MOVZBL	16(IDX_DFN), R0	0345
	50	64	A9	C0	00049	ADDL2	100(IRAB), R0	
	E4		60	E9	0004D	BLBC	(R0), 2\$	
	7E	0A	A9	9A	00050	MOVZBL	10(IRAB), -(SP)	0347
			56	DD	00054	PUSHL	REC_ADDR	
	7E	22	A8	3C	00056	MOVZWL	34(RAB), -(SP)	
			0000G	30	0005A	BSBW	RM\$NOREAD_LONG	
	5E		0C	C0	0005D	ADDL2	#12, SP	
	D1		50	E8	00060	BLBS	R0, 2\$	
	22	22	A8	B1	00063	CMPW	34(RAB), 34(IDX_DFN)	0349
			CA	1F	00068	BLSSU	2\$	
	5C		03	D0	0006A	MOVL	#3, AP	0358
	50	00B4	CA	3C	0006D	MOVZWL	180(IFAB), R0	0364
		60	B940	9F	00072	PUSHAB	@96(IRAB)[R0]	
			0000G	30	00076	BSBW	RM\$RECORD_KEY	
	5C		01	D0	00079	MOVL	#1, AP	0367
	50	00B4	CA	3C	0007C	MOVZWL	180(IFAB), R0	0368
	6E	60	B940	9E	00081	MOVAB	@96(IRAB)[R0], (SP)	
			0000G	30	00086	BSBW	RM\$NULLKEY	
	5E		04	C0	00089	ADDL2	#4, SP	
	A5		50	E9	0008C	BLBC	R0, 2\$	
			0000G	30	0008F	BSBW	RM\$DELETE_SIDR	0370
			A0	11	00092	BRB	2\$	0320
	00B0		55	D0	00094	MOVL	SAVE_UDR_VBN, 176(IRAB)	0378
	00BC		52	B0	00099	MOVW	SAVE_UDR_ID, 188(IRAB)	0379
42	06		03	E0	0009E	BBS	#3, 8(IRAB), 7\$	0400
			7E	D4	000A3	CLRL	-(SP)	0404
			0000G	30	000A5	BSBW	RM\$KEY_DESC	
	40		01	90	000A8	MOVB	#1, 64(IRAB)	0414
		3C	A9	D4	000AC	CLRL	60(IRAB)	0415
	6E		01	D0	000AF	MOVL	#1, (SP)	0417
	7E	0080	C9	3C	000B2	MOVZWL	128(IRAB), -(SP)	
		78	A9	DD	000B7	PUSHL	120(IRAB)	
			0000G	30	000BA	BSBW	RM\$FIND_BY_RRV	
	5E		0C	C0	000BD	ADDL2	#12, SP	
	22		50	E9	000C0	BLBC	R0, 7\$	
	54	3C	A9	D0	000C3	MOVL	60(IRAB), BDB	0428
			06	13	000C7	BEQL	6\$	
		3C	A9	D4	000C9	CLRL	60(IRAB)	0431
			0000G	30	000CC	BSBW	RM\$DELETE_RRV	0432
	54	20	A9	D0	000CF	MOVL	32(IRAB), BDB	0437
		20	A9	D4	000D3	CLRL	32(IRAB)	0438
			0000G	30	000D6	BSBW	RM\$DELETE_UDR	0440
	0A		02	88	000D9	BISB2	#2, 10(BDB)	0445
			7E	D4	000DD	CLRL	-(SP)	0446
			0000G	30	000DF	BSBW	RM\$RLSBKT	

RM3PUTERR
V04-000

RM\$PUTUPD_ERROR

C 5
16-Sep-1984 01:58:54
14-Sep-1984 13:01:37

VAX-11 Bliss-32 V4.0-742
[RMS.SRC]RM3PUTERR.B32;1

Page 10
(2)

5E

007D

04
8F

CO 000E2
BA 000E5
OS 000E9

7\$:

ADDL2
POPR
RSB

#4, SP
#^M<R0,R2,R3,R4,R5,R6>

: 0452
:
:

; Routine Size: 234 bytes, Routine Base: RM\$RMS3 + 0000

RMSCLEAN_BDB

```
391 0453 1 %SBTTL 'RMSCLEAN_BDB'
392 0454 1 GLOBAL ROUTINE RMSCLEAN_BDB : RL$ERROR_LINK1 NOVALUE =
393 0455 1
394 0456 1 ++
395 0457 1
396 0458 1 FUNCTIONAL DESCRIPTION:
397 0459 1
398 0460 1 This routine's responsibility is to release any buckets that
399 0461 1 are currently accessed.
400 0462 1
401 0463 1
402 0464 1 CALLING SEQUENCE:
403 0465 1
404 0466 1 BSBW RMSCLEAN_BDB()
405 0467 1
406 0468 1 INPUT PARAMETERS:
407 0469 1 NONE
408 0470 1
409 0471 1 IMPLICIT INPUTS:
410 0472 1
411 0473 1 IRAB - address of IRAB
412 0474 1 IRB$L_CURBDB
413 0475 1 IRB$L_LOCKBDB
414 0476 1 IRB$L_NXTBDB
415 0477 1
416 0478 1 OUTPUT PARAMETERS:
417 0479 1 NONE
418 0480 1
419 0481 1 IMPLICIT OUTPUTS:
420 0482 1 NONE
421 0483 1
422 0484 1 ROUTINE VALUE:
423 0485 1 NONE
424 0486 1
425 0487 1 SIDE EFFECTS:
426 0488 1
427 0489 1 If there is a bucket associated with IRB$L_NXTBDB, it is released.
428 0490 1 if there is a bucket associated with IRB$L_LOCK BDB, it is released.
429 0491 1 if there is a bucket associated with IRB$L_CURBDB, it is released.
430 0492 1
431 0493 1 --
432 0494 1
433 0495 2 BEGIN
434 0496 2
435 0497 2 EXTERNAL REGISTER
436 0498 2 COMMON_RAB_STR;
437 0499 2
438 0500 2 GLOBAL REGISTER
439 0501 2 R_REC_ADDR,
440 0502 2 R_IDX_DFN,
441 0503 2 R_BDB_STR;
442 0504 2
443 0505 2 ! If there is an accessed bucket associated with IRB$L_NXTBDB,
444 0506 2 ! then release it.
445 0507 2
446 0508 2 IF (BDB = .IRAB[IRB$L_NXTBDB]) NEQ 0
447 0509 2 THEN
```

```

: 448      0510      3      BEGIN
: 449      0511      3      IRAB[IRB$L_NXTBDB] = 0;
: 450      0512      3      RMS$RLSBKT(0);
: 451      0513      3      END;
: 452      0514      3
: 453      0515      3      ! If there is an accessed bucket associated with IRB$L_LOCK_BDB,
: 454      0516      3      ! then release it.
: 455      0517      3
: 456      0518      3      IF (BDB = .IRAB[IRB$L_LOCK_BDB]) NEQ 0
: 457      0519      3      THEN
: 458      0520      3      BEGIN
: 459      0521      3      IRAB[IRB$L_LOCK_BDB] = 0;
: 460      0522      3      RMS$RLSBKT(0);
: 461      0523      3      END;
: 462      0524      3
: 463      0525      3      ! If there is an accessed bucket associated with IRB$L_CURBDB,
: 464      0526      3      ! then release it.
: 465      0527      3
: 466      0528      3      IF (BDB = .IRAB[IRB$L_CURBDB]) NEQ 0
: 467      0529      3      THEN
: 468      0530      3      BEGIN
: 469      0531      3      IRAB[IRB$L_CURBDB] = 0;
: 470      0532      3      RMS$RLSBKT(0);
: 471      0533      3      END;
: 472      0534      3
: 473      0535      1      END;
```

	00D1	8F	BB	00000	RMS\$CLEAN_BDB::		
					PUSHR	#*M<R0,R4,R6,R7>	
54	3C	A9	D0	00004	MOVL	60(IRAB), BDB	0454
		0B	13	00008	BEQL	1\$	0508
	3C	A9	D4	0000A	CLRL	60(IRAB)	0511
		7E	D4	0000D	CLRL	-(SP)	0512
		0000G	30	0000F	BSBW	RMS\$RLSBKT	
5E		04	C0	00012	ADDL2	#4, SP	
54	0084	C9	D0	00015	1\$: MOVL	132(IRAB), BDB	0518
		0C	13	0001A	BEQL	2\$	
	0084	C9	D4	0001C	CLRL	132(IRAB)	0521
		7E	D4	00020	CLRL	-(SP)	0522
		0000G	30	00022	BSBW	RMS\$RLSBKT	
5E		04	C0	00025	ADDL2	#4, SP	
54	20	A9	D0	00028	2\$: MOVL	32(IRAB), BDB	0528
		0B	13	0002C	BEQL	3\$	
	20	A9	D4	0002E	CLRL	32(IRAB)	0531
		7E	D4	00031	CLRL	-(SP)	0532
		0000G	30	00033	BSBW	RMS\$RLSBKT	
5E		04	C0	00036	ADDL2	#4, SP	
	00D1	8F	BA	00039	3\$: POPR	#*M<R0,R4,R6,R7>	0535
			05	0003D	RSB		

; Routine Size: 62 bytes, Routine Base: RMS\$RMS3 + 00EA

RM3PUTERR
V04-000

RM\$CLEAN_BDB

F 5
16-Sep-1984 01:58:54
14-Sep-1984 13:01:37

VAX-11 Bliss-32 V4.0-742
[RMS.SRC]RM3PUTERR.B32;1

Page 13
(3)

```
: 474      0536 1
: 475      0537 1 END
: 476      0538 1
: 477      0539 0 ELUDOM
```

PSECT SUMMARY

```
:
:      Name                Bytes                Attributes
:
:  RM$RMS3                296 NOVEC,NOWRT, RD , EXE,NOSHR, GBL, REL, CON, PIC,ALIGN(2)
```

Library Statistics

```
:
:      File                ----- Symbols ----- Pages Processing
:                        Total   Loaded   Percent   Mapped   Time
:
:  _$255$DUA28:[RMS.OBJ]RMS.L32;1      3109       51       1       154       00:00.4
```

COMMAND QUALIFIERS

```
:
:  BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS$:RM3PUTERR/OBJ=OBJ$:RM3PUTERR MSRC$:RM3PUTERR/UPDATE=(ENH$:RM3PUTERR)
```

```
: Size:                296 code + 0 data bytes
: Run Time:            00:09.4
: Elapsed Time:        00:19.5
: Lines/CPU Min:       3444
: Lexemes/CPU-Min:     15559
: Memory Used:         103 pages
: Compilation Complete
```

0327 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RM3PROBE
LIS

RM3IOXSP
LIS

RM3PUTERR
LIS

RM3PUTUPD
LIS

RM3SPLUDR
LIS

RM3RRU
LIS

RM3ROOT
LIS

RM3PUT
LIS