```
RRRRRRRRRRR      MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSS
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
RRRRRRRRRRR      MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
```

```
RRRRRRRR    MM      MM    333333    BBBBBBBB    KK      KK  TTTTTTTTTT    IIIIII      000000
RRRRRRRR    MM      MM    333333    BBBBBBBB    KK      KK  TTTTTTTTTT    IIIIII      000000
RR      RR  MMMM  MMMM  33      33  BB      BB  KK      KK      TT          II      00      00
RR      RR  MMMM  MMMM  33      33  BB      BB  KK      KK      TT          II      00      00
RR      RR  MM  MM  MM        33    BB      BB  KK    KK        TT          II      00      00
RR      RR  MM  MM  MM        33    BB      BB  KK    KK        TT          II      00      00
RRRRRRRR    MM      MM      33      BBBBBBBB    KKKKKK          TT          II      00      00
RRRRRRRR    MM      MM      33      BBBBBBBB    KKKKKK          TT          II      00      00
RR   RR     MM      MM        33    BB      BB  KK    KK        TT          II      00      00
RR   RR     MM      MM        33    BB      BB  KK    KK        TT          II      00      00
RR    RR    MM      MM  33      33  BB      BB  KK      KK      TT          II      00      00      ....
RR    RR    MM      MM  33      33  BB      BB  KK      KK      TT          II      00      00      ....
RR     RR   MM      MM    333333    BBBBBBBB    KK      KK      TT      IIIIII      000000        ....
RR     RR   MM      MM    333333    BBBBBBBB    KK      KK      TT      IIIIII      000000        ....
```

```
LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
;     1          0001  0
;     2          0002  0 MODULE RM3BKTIO (LANGUAGE (BLISS32) ,
;     3          0003  0                  IDENT = 'V04-000'
;     4          0004  0                  ) =
;     5          0005  1 BEGIN
;     6          0006  1 !
;     7          0007  1 !**************************************************************************
;     8          0008  1 !*                                                                        *
;     9          0009  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;    10          0010  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;    11          0011  1 !*  ALL RIGHTS RESERVED.                                                  *
;    12          0012  1 !*                                                                        *
;    13          0013  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    14          0014  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    15          0015  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    16          0016  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    17          0017  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    18          0018  1 !*  TRANSFERRED.                                                           *
;    19          0019  1 !*                                                                        *
;    20          0020  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    21          0021  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    22          0022  1 !*  CORPORATION.                                                           *
;    23          0023  1 !*                                                                        *
;    24          0024  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    25          0025  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    26          0026  1 !*                                                                        *
;    27          0027  1 !*                                                                        *
;    28          0028  1 !**************************************************************************
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 !
;    32          0032  1 ! FACILITY:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
;    33          0033  1 !
;    34          0034  1 ! ABSTRACT:
;    35          0035  1 !                This module performs IO for IDX file buckets checking and
;    36          0036  1 !                updating the reliablity data in the bucket overhead area.
;    37          0037  1 !
;    38          0038  1 !
;    39          0039  1 ! ENVIRONMENT:
;    40          0040  1 !
;    41          0041  1 !                VAX/VMS OPERATING SYSTEM
;    42          0042  1 !
;    43          0043  1 !--
;    44          0044  1
;    45          0045  1 !
;    46          0046  1 ! AUTHOR:       E. H. Marison           28-Mar-1978
;    47          0047  1 !
;    48          0048  1 ! MODIFIED BY:
;    49          0049  1 !
;    50          0050  1 !       V03-004 MCN0002         Maria del C. Nasr       22-Mar-1983
;    51          0051  1 !               More linkages reorganization
;    52          0052  1 !
;    53          0053  1 !       V03-003 MCN0001         Maria del C. Nasr       24-Feb-1983
;    54          0054  1 !               Reorganize linkages.
;    55          0055  1 !
;    56          0056  1 !       V03-002 TMK0001         Todd M. Katz            01-Nov-1982
;    57          0057  1 !               Make a modification to the bucket format checking done in
```

```
;    58          0058  1 ;                RM$GETBKT. This routine scans the bucket in order to make sure
;    59          0059  1 ;                that the records in the bucket actually end where the freespace
;    60          0060  1 ;                offset pointer says they do. Actually one of several scans was
;    61          0061  1 ;                done based upon the level of the bucket, prologue version of
;    62          0062  1 ;                the file, and the state of key compression. It is now possible
;    63          0063  1 ;                to do the same scan for every possible combination because I
;    64          0064  1 ;                have re-written the routine RM$REC_OVHD to take into account
;    65          0065  1 ;                every concievable combination of prologue version, key
;    66          0066  1 ;                compression state, and bucket level. This change was necessary
;    67          0067  1 ;                because the format checking of prologue 3 SIDR buckets was
;    68          0068  1 ;                incorrect, and would occassionally result in an IRC error when
;    69          0069  1 ;                the bucket actually had the correct structure.
;    70          0070  1 ;
;    71          0071  1 ;        V03-001 KBT0154         Keith B. Thompson       21-Aug-1982
;    72          0072  1 ;                Reorganize psects
;    73          0073  1 ;
;    74          0074  1 ;        V02-017 CDS0006         C Saether               8-Feb-1982
;    75          0075  1 ;                Only do bucket pre-scan when locking bucket.
;    76          0076  1 ;
;    77          0077  1 ;        V02-016 CDS0005         C Saether               28-Jan-1982
;    78          0078  1 ;                Make check for BLB instead of BDB in RLSBKT so that
;    79          0079  1 ;                the cache value is stored in gbpb's.
;    80          0080  1 ;
;    81          0081  1 ;        V02-015 CDS0004         C Saether               29-Dec-1981
;    82          0082  1 ;                RLSBKT also needs to check whether bucket was locked
;    83          0083  1 ;                before incrementing check characters.
;    84          0084  1 ;
;    85          0085  1 ;        V02-014 CDS0003         C Saether               9-Dec-1981
;    86          0086  1 ;                Comment out references to the CSH$M_READAHEAD flag.
;    87          0087  1 ;
;    88          0088  1 ;        V02-013 PSK0005         Paulina S. Knibbe       04-Oct-1981
;    89          0089  1 ;                Fix 012 below.
;    90          0090  1 ;
;    91          0091  1 ;        V02-012 PSK0004         Paulina S. Knibbe       01-Oct-1981
;    92          0092  1 ;                Make sure prologue one/two tests are not applied
;    93          0093  1 ;                to prologue three buckets.
;    94          0094  1 ;
;    95          0095  1 ;        V02-011 CDS0002         C Saether               10-Sep-1981
;    96          0096  1 ;                Correction to 010.
;    97          0097  1 ;
;    98          0098  1 ;        V02-010 CDS0001         C Saether               30-Aug-1981
;    99          0099  1 ;                Have rm$rlsbkt handle bdb value of 0 - this occurs
;   100          0100  1 ;                when trying to release a lock bdb and the file
;   101          0101  1 ;                is not shared.  Also store cache value.
;   102          0102  1 ;
;   103          0103  1 ;        V02-009 PSK0003         Paulina S. Knibbe       06-May-1981
;   104          0104  1 ;                Fix infinite loop in check of compressed index
;   105          0105  1 ;                buckets.
;   106          0106  1 ;
;   107          0107  1 ;        V02-008 PSK0002         Paulina S. Knibbe       17-Apr-1981
;   108          0108  1 ;                Change variable names
;   109          0109  1 ;
;   110          0110  1 ;        V02-007 PSK0001         Paulina S. Knibbe       29-Mar-1981
;   111          0111  1 ;                Add checks appropriate for prologue Version 3.0 index
;   112          0112  1 ;                and SIDR buckets
;   113          0113  1 ;
;   114          0114  1 ;        V02-006 REFORMAT        Frederick E. Deen, Jr.      23-Jul-1980
```

```
;   115          0115  1 !                This code was reformatted to adhere to RMS standards
;   116          0116  1 !
;   117          0117  1 !
;   118          0118  1 ! REVISION HISTORY:
;   119          0119  1 !
;   120          0120  1 !   Christian Saether,  14-Aug-1978
;   121          0121  1 !   X0002 - RLSBKT to check if buffer there
;   122          0122  1 !
;   123          0123  1 !   Christian Saether,  11-Oct-1978
;   124          0124  1 !   X0003 - modify GETBKT to use REC_OVHD, reduce stack at IO
;   125          0125  1 !
;   126          0126  1 !   Christian Saether,  19-Oct-1978
;   127          0127  1 !   X0004 - GETBKT always releases bucket on error if still accessed
;   128          0128  1 !
;   129          0129  1 !   Wendy Koenig,       24-Oct-1978
;   130          0130  1 !   X0005 - Make changes caused by sharing conventions
;   131          0131  1 !
;   132          0132  1 !****
;   133          0133  1
;   134          0134  1 LIBRARY 'RMSLIB:RMS';
;   135          0135  1
;   136          0136  1 REQUIRE 'RMSSRC:RMSIDXDEF';
;   137          0201  1
;   138          0202  1 ! Define default PSECTS
;   139          0203  1 !
;   140          0204  1
;   141          0205  1 PSECT
;   142          0206  1     CODE = RM$RMS3(PSECT_ATTR),
;   143          0207  1     PLIT = RM$RMS3(PSECT_ATTR);
;   144          0208  1
;   145          0209  1 ! Linkages
;   146          0210  1 !
;   147          0211  1
;   148          0212  1 LINKAGE
;   149          0213  1     L_PRESERVE1,
;   150          0214  1     L_RABREG_457,
;   151          0215  1     L_REC_OVHD,
;   152          0216  1     L_CACHE,
;   153          0217  1     L_RELEASE;
;   154          0218  1
;   155          0219  1 ! Forward Routines
;   156          0220  1 !
;   157          0221  1
;   158          0222  1 FORWARD ROUTINE
;   159          0223  1     RM$RLSBKT   : RL$PRESERVE1;
;   160          0224  1
;   161          0225  1 ! External Routines
;   162          0226  1 !
;   163          0227  1
;   164          0228  1 EXTERNAL ROUTINE
;   165          0229  1     RM$CACHE    : RL$CACHE,
;   166          0230  1     RM$REC_OVHD : RL$REC_OVHD,
;   167          0231  1     RM$RELEASE  : RL$RELEASE;
;   168          0232  1
```

```
;  170     0233  1 GLOBAL ROUTINE RM$GETBKT (VBN, SIZE) : RL$RABREG_457 =
;  171     0234  1
;  172     0235  1 !++
;  173     0236  1 !
;  174     0237  1 ! RM$GETBKT
;  175     0238  1 !
;  176     0239  1 ! This routine calls the RMS cache routine (RMOCACHE) for
;  177     0240  1 ! the requested bucket (VBN,SIZE). If an actual IO was done
;  178     0241  1 ! (i.e., the CSH$V_NOREAD, CSH$V_READAHEAD, or CSH$V_NOBUFFER bits
;  179     0242  1 ! are all off) and it was successful then the bucket's overhead
;  180     0243  1 ! area data is checked and if in error a status value of
;  181     0244  1 ! RMS$_CHK, RMS$_IRC, or RMS$_IBF is returned depending on
;  182     0245  1 ! the nature of the error.  IRAB[IRB$B_CACHEFLAGS] is always zeroed.
;  183     0246  1 !
;  184     0247  1 ! CALLING SEQUENCE:
;  185     0248  1 !       RM$GETBKT (VBN,SIZE)
;  186     0249  1 !
;  187     0250  1 ! INPUT PARAMETERS:
;  188     0251  1 !       VBN - Start VBN for bucket
;  189     0252  1 !       SIZE - Number of bytes in bucket
;  190     0253  1 !
;  191     0254  1 ! IMPLICIT INPUTS:
;  192     0255  1 !       IRAB [ CACHEFLGS ] - cacheflags for CACHE (See RMOCACHE)
;  193     0256  1 !       IDX_DFN - used by RM$GETNEXT_REC routine
;  194     0257  1 !
;  195     0258  1 ! OUTPUT PARAMETERS:
;  196     0259  1 !       BDB - Address of BDB for bucket
;  197     0260  1 !       BKT_ADDR - Address of the bucket buffer
;  198     0261  1 !       RAB [ STV ] - VBN of bucket on IRC and IBF errors
;  199     0262  1 !
;  200     0263  1 ! IMPLICIT OUTPUTS:
;  201     0264  1 !       IRAB [ CACHEFLAGS ] zeroed
;  202     0265  1 !
;  203     0266  1 ! ROUTINE VALUE:
;  204     0267  1 !       Internal RMS status code
;  205     0268  1 !
;  206     0269  1 ! SIDE EFFECTS:
;  207     0270  1 !       R1,R2,R3,AP are destroyed
;  208     0271  1 !
;  209     0272  1 !--
;  210     0273  1
;  211     0274  2     BEGIN
;  212     0275  2
;  213     0276  2     LABEL
;  214     0277  2         BLK;
;  215     0278  2
;  216     0279  2     EXTERNAL REGISTER
;  217     0280  2         COMMON_IO_STR,
;  218     0281  2         R_IDX_DFN_STR,
;  219     0282  2         COMMON_RAB_STR;
;  220     0283  2
;  221     0284  2     LITERAL
;  222     0285  3         NODATABITS = (CSH$M_NOREAD
;  223     0286  3 !           OR
;  224     0287  3 !           CSH$M_READAHEAD
;  225     0288  3             OR
;  226     0289  2             CSH$M_NOBUFFER),
```

```
;  227     0290  2          PVN1_BITS = BKT$M_LASTBKT
;  228     0291  2                  OR
;  229     0292  2                  BKT$M_ROOTBKT;
;  230     0293  2
;  231     0294  3      BEGIN
;  232     0295  3
;  233     0296  3      LOCAL
;  234     0297  3          STATUS;
;  235     0298  3
;  236     0299  4      IF (STATUS = RM$CACHE(.VBN, .SIZE, .IRAB[IRB$B_CACHEFLGS]))
;  237     0300  3      THEN
;  238     0301  4          BEGIN
;  239     0302  4
;  240     0303  5          IF (.IRAB[IRB$B_CACHEFLGS]
;  241     0304  5              AND
;  242     0305  4              NODATABITS) NEQ 0
;  243     0306  4          THEN
;  244     0307  5              BEGIN
;  245     0308  5              IRAB[IRB$B_CACHEFLGS] = 0;
;  246     0309  5              RETURN .STATUS
;  247     0310  5
;  248     0311  5              END
;  249     0312  5
;  250     0313  4          END
;  251     0314  3      ELSE
;  252     0315  4          BEGIN
;  253     0316  4          IRAB[IRB$B_CACHEFLGS] = 0;
;  254     0317  4          RETURN .STATUS
;  255     0318  4
;  256     0319  3          END;
;  257     0320  3
;  258     0321  2      END;                                    ! of this local STATUS
;  259     0322  2
;  260     0323  3      BEGIN
;  261     0324  3
;  262     0325  3      LOCAL
;  263     0326  3          STATUS;
;  264     0327  3
;  265     0328  3      STATUS =
;  266     0329  3  BLK :
;  267     0330  4      BEGIN
;  268     0331  4
;  269     0332  4      LOCAL
;  270     0333  4          REC_SIZE,
;  271     0334  4          EOB,
;  272     0335  4          LEVEL;
;  273     0336  4
;  274     0337  4      GLOBAL REGISTER
;  275     0338  4          R_REC_ADDR;
;  276     0339  4
;  277     0340  5      IF (.BKT_ADDR[BKT$W_ADRSAMPLE] NEQU .(BDB[BDB$L_VBN])<0, 16>
;  278     0341  5          OR
;  279     0342  5          .BKT_ADDR[BKT$B_CHECKCHAR] NEQU .(.BKT_ADDR + .BDB[BDB$W_NUMB] - 1)<0,
;  280     0343  5              8>)
;  281     0344  4      THEN
;  282     0345  4          LEAVE BLK WITH RMSERR(CHK);
;  283     0346  4
```

```
;  284          0347  4     ! Check to make sure that only prologue version 1 bits are on and that
;  285          0348  4     ! the beginning of freespace is somewhere in the bucket.
;  286          0349  4     !
;  287          0350  4
;  288          0351  4     IF .IFAB [IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  289          0352  4     THEN
;  290          0353  4
;  291          0354  5         BEGIN
;  292          0355  5         IF (.BKT_ADDR[BKT$B_BKTCB] AND  NOT PVN1_BITS) NEQ 0
;  293          0356  5             OR
;  294          0357  5             .BKT_ADDR[BKT$W_FREESPACE] GTRU .BDB[BDB$W_NUMB]
;  295          0358  5         THEN
;  296          0359  5             LEAVE BLK WITH RMSERR(IBF);
;  297          0360  5         END
;  298          0361  5
;  299          0362  5     !   Check the KEYREF matches index number for this bucket
;  300          0363  5     !   (If this is a prologue 3 file)
;  301          0364  5     !
;  302          0365  4     ELSE
;  303          0366  4         IF .IDX_DFN [IDX$B_KEYREF] NEQ .BKT_ADDR [BKT$B_INDEXNO]
;  304          0367  4         THEN
;  305          0368  4             LEAVE BLK WITH RMSERR (IBF);
;  306          0369  4
;  307          0370  4     IF NOT .BBLOCK[IRAB[IRB$B_CACHEFLGS], CSH$V_LOCK]
;  308          0371  4     THEN
;  309          0372  4
;  310          0373  5         BEGIN
;  311          0374  5         STATUS = RMSSUC(SUC);
;  312          0375  5         IRAB[IRB$B_CACHEFLGS] = 0;
;  313          0376  5         RETURN .STATUS;
;  314          0377  4         END;
;  315          0378  4
;  316          0379  4     ! Make one scan through entire bucket confirming that records actually end
;  317          0380  4     ! where freespace says they do.
;  318          0381  4     !
;  319          0382  4     EOB = .BKT_ADDR + .BKT_ADDR[BKT$W_FREESPACE];
;  320          0383  4     REC_ADDR = .BKT_ADDR + BKT$C_OVERHDSZ;
;  321          0384  4
;  322          0385  4     IF (LEVEL = .BKT_ADDR[BKT$B_LEVEL]) EQL 0
;  323          0386  4     THEN
;  324          0387  4         IF .IDX_DFN[IDX$B_KEYREF] NEQ 0
;  325          0388  4         THEN
;  326          0389  4             LEVEL = .LEVEL - 1;
;  327          0390  4
;  328          0391  4     WHILE .REC_ADDR LSSA .EOB
;  329          0392  4     DO
;  330          0393  5         BEGIN
;  331          0394  5         REC_ADDR = RM$REC_OVHD(.LEVEL; REC_SIZE) + .REC_ADDR;
;  332          0395  5         REC_ADDR = .REC_ADDR + .REC_SIZE;
;  333          0396  4         END;
;  334          0397  4
;  335          0398  4     IF .REC_ADDR EQLA .EOB
;  336          0399  4     THEN
;  337          0400  5         BEGIN
;  338          0401  5         STATUS = RMSSUC(SUC);
;  339          0402  5         IRAB [IRB$B_CACHEFLGS] = 0;
;  340          0403  5         RETURN .STATUS;
```

```
;  341          0404  4          END;
;  342          0405  4
;  343          0406  4      ! REC_ADDR did not match where FREESPACE said it should
;  344          0407  4      !
;  345          0408  5      RMSERR(IRC)
;  346          0409  3      END;                                        ! of block BLK
;  347          0410  3
;  348          0411  3      ! If we got this far there was an error checking something, so release
;  349          0412  3      ! this bucket and return the error save VBN in the STV. Mark the bucket
;  350          0413  3      ! invalid.
;  351          0414  3      RAB[RAB$L_STV] = .BDB[BDB$L_VBN];
;  352          0415  3      BDB[BDB$V_VAL] = 0;
;  353          0416  3
;  354          0417  4      BEGIN
;  355          0418  4
;  356          0419  4      GLOBAL REGISTER
;  357          0420  4          R_REC_ADDR;
;  358          0421  4
;  359          0422  4      RM$RLSBKT(0);
;  360          0423  3      END;
;  361          0424  3
;  362          0425  3      IRAB[IRB$B_CACHEFLGS] = 0;
;  363          0426  3      RETURN .STATUS;
;  364          0427  3
;  365          0428  3      END                                     ! of second block defining STATUS
;  366          0429  1      END;


                                                .TITLE  RM3BKTIO
                                                .IDENT  \V04-000\

                                                .EXTRN  RM$CACHE, RM$REC_OVHD
                                                .EXTRN  RM$RELEASE

                                                .PSECT  RM$RMS3,NOWRT,  GBL,  PIC,2

                    004C    8F   BB 00000 RM$GETBKT::
                                           PUSHR   #^M<R2,R3,R6>                        ; 0233
                         5E      08   C2 00004         SUBL2   #8, SP
                         40      A9   9F 00007         PUSHAB  64(IRAB)                ; 0299
                         53      00   BE 9A 0000A      MOVZBL  @0(SP), R3
                         51      1C   AE 7D 0000E      MOVQ    VBN, R1
                               0000G  30 00012         BSBW    RM$CACHE
                         06        50 E9 00015         BLBC    STATUS, 1$
                         0C      00   BE 93 00018      BITB    @0(SP), #12            ; 0305
                                 06   13 0001C         BEQL    2$
                                 00   BE 94 0001E 1$:  CLRB    @0(SP)                 ; 0316
                                 009B 31 00021         BRW     14$                    ; 0317
                   1C    A4      02   A5 B1 00024 2$:  CMPW    2(BKT_ADDR), 28(BDB)   ; 0340
                                 0B   12 00029         BNEQ    3$
                         50      14   A4 3C 0002B      MOVZWL  20(BDB), R0            ; 0342
                      FF A045          65 91 0002F      CMPB    (BKT_ADDR), -1(R0)[BKT_ADDR]
                                       07 13 00034      BEQL    4$
                         53    84A4    8F 3C 00036 3$:  MOVZWL  #33956, STATUS         ; 0345
                                       6B 11 0003B      BRB     12$
                         03    00B7    CA 91 0003D 4$:  CMPB    183(IFAB), #3          ; 0351
                                       10 1E 00042      BGEQU   5$
```

```
                        FC   8F      0D    A5  93 00044          BITB    13(BKT_ADDR), #252                      ; 0355
                                           10  12 00049          BNEQ    6$
                        14   A4      04    A5  B1 0004B          CMPW    4(BKT_ADDR), 20(BDB)                    ; 0357
                                           10  1B 00050          BLEQU   7$
                                           07  11 00052          BRB     6$                                      ; 0359
                        01   A5      21    A7  91 00054 5$:      CMPB    33(IDX_DFN), 1(BKT_ADDR)                ; 0366
                                           07  13 00059          BEQL    7$
                             53    8754    8F  3C 0005B 6$:      MOVZWL  #34644, STATUS                          ; 0368
                                           46  11 00060          BRB     12$
                             35    00      BE  E9 00062 7$:      BLBC    @0(SP), 10$                             ; 0370
                             50    04      A5  3C 00066          MOVZWL  4(BKT_ADDR), R0                         ; 0382
                   04   AE        55          50  C1 0006A      ADDL3   R0, BRT_ADDR, EOB
                             56    0E      A5  9E 0006F          MOVAB   14(R5), REC_ADDR                        ; 0383
                             52    0C      A5  9A 00073          MOVZBL  12(BKT_ADDR), LEVEL                     ; 0385
                                           07  12 00077          BNEQ    8$
                                   21      A7  95 00079          TSTB    33(IDX_DFN)                             ; 0387
                                           02  13 0007C          BEQL    8$
                                           52  D7 0007E          DECL    LEVEL                                   ; 0389
                        04   AE             56  D1 00080 8$:      CMPL    REC_ADDR, EOB                           ; 0391
                                           13  1E 00084          BGEQU   9$
                             51            52  D0 00086          MOVL    LEVEL, R1                               ; 0394
                                      0000G 30  00089          BSBW    RM$REC_OVHD
                        08   AE             51  D0 0008C          MOVL    R1, 8(SP)
                             56            50  C0 00090          ADDL2   R0, REC_ADDR
                             56    08      AE  C0 00093          ADDL2   REC_SIZE, REC_ADDR                      ; 0395
                                           E7  11 00097          BRB     8$                                      ; 0391
                                           08  12 00099 9$:      BNEQ    11$                                     ; 0398
                             53            01  D0 0009B 10$:     MOVL    #1, STATUS                              ; 0401
                                   00      BE  94 0009E          CLRB    @0(SP)                                  ; 0402
                                           19  11 000A1          BRB     13$                                     ; 0403
                             53    857C    8F  3C 000A3 11$:     MOVZWL  #34172, STATUS                          ; 0408
                        0C   A8      1C    A4  D0 000A8 12$:     MOVL    28(BDB), 12(RAB)                        ; 0414
                        0A   A4              01  8A 000AD          BICB2   #1, 10(BDB)                             ; 0415
                                           7E  D4 000B1          CLRL    -(SP)                                   ; 0422
                                      0000V 30  000B3          BSBW    RM$RLSBKT
                             5E            04  C0 000B6          ADDL2   #4, SP
                                   40      A9  94 000B9          CLRB    64(IRAB)                                ; 0425
                             50            53  D0 000BC 13$:     MOVL    STATUS, R0                              ; 0426
                             5E            0C  C0 000BF 14$:     ADDL2   #12, SP                                 ; 0429
                                   004C    8F  BA 000C2          POPR    #^M<R2,R3,R6>
                                           05  000C6          RSB

; Routine Size:  199 bytes,    Routine Base:  RM$RMS3 + 0000

;  367          0430  1
```

```
;  369     0431  1 GLOBAL ROUTINE RM$RLSBKT (FLAGS) : RL$PRESERVE1 =
;  370     0432  1
;  371     0433  1 !++
;  372     0434  1 !
;  373     0435  1 ! FUNCTIONAL DESCRIPTION:
;  374     0436  1 !
;  375     0437  1 ! This routine releases access to the bucket described by
;  376     0438  1 ! the BDB and if a write may be performed (i.e; there is a
;  377     0439  1 ! buffer and it's dirty) then the bucket check characters
;  378     0440  1 ! are incremented.  Bug check is called if the address sample
;  379     0441  1 ! in the buffer does not match the VBN being released.
;  380     0442  1 !
;  381     0443  1 ! CALLING SEQUENCE:
;  382     0444  1 !       RM$RLSBKT(FLAGS,BDB)
;  383     0445  1 !
;  384     0446  1 ! INPUT PARAMETERS:
;  385     0447  1 !       BDB - The address of the BDB for the bucket
;  386     0448  1 !       flags - The release control flags (see RMORELEAS)
;  387     0449  1 !
;  388     0450  1 ! IMPLICIT INPUTS:
;  389     0451  1 !       None
;  390     0452  1 !
;  391     0453  1 ! OUTPUT PARAMETERS:
;  392     0454  1 !       None
;  393     0455  1 !
;  394     0456  1 ! IMPLICIT OUTPUTS:
;  395     0457  1 !       None
;  396     0458  1 !
;  397     0459  1 ! ROUTINE VALUE:
;  398     0460  1 !       Internal RMS status code
;  399     0461  1 !
;  400     0462  1 ! SIDE EFFECTS:
;  401     0463  1 !       R1,R2,AP Destroyed
;  402     0464  1 !       Bug check called on address sample and VBN mismatch
;  403     0465  1 !
;  404     0466  1 !--
;  405     0467  1
;  406     0468  2     BEGIN
;  407     0469  2
;  408     0470  2     EXTERNAL REGISTER
;  409     0471  2         COMMON_RABREG,
;  410     0472  2         R_BDB_STR;
;  411     0473  2
;  412     0474  2     LOCAL
;  413     0475  2         BUCKET  : REF BBLOCK;
;  414     0476  2
;  415     0477  2     BUILTIN
;  416     0478  2         TESTBITSC;
;  417     0479  2
;  418     0480  2     ! Note that BDB's and GBPB's share identical fields in general,
;  419     0481  2     ! therefore the references to BDB fields may be GBPB's also.
;  420     0482  2     !
;  421     0483  2
;  422     0484  3     IF .BDB EQL 0 OR (.BDB[BDB$B_BID] EQL BLB$C_BID)
;  423     0485  2     THEN RETURN RM$RELEASE(.FLAGS);
;  424     0486  2
;  425     0487  2     ! If the bucket may be written then update the check characters.
```

```
!

IF .BDB[BDB$V_VAL]
    AND
    (.BDB[BDB$W_SIZE] NEQ 0)
THEN
    BEGIN
    BUCKET = .BDB[BDB$L_ADDR];

    ! check the address sample for validity
    !

    IF .BUCKET[BKT$W_ADRSAMPLE] NEQU .(BDB[BDB$L_VBN])<0, 16>
    THEN
        BUG_CHECK;

    IF .BDB[BDB$V_DRT]
    THEN
        BEGIN
        LOCAL PTR : REF BBLOCK;
        PTR = .BDB[BDB$L_BLB_PTR];
        IF .PTR EQL 0
        THEN
            BEGIN
            BUCKET[BKT$B_CHECKCHAR] = .BUCKET[BKT$B_CHECKCHAR] + 1;
            (.BUCKET + .BDB[BDB$W_NUMB] - 1)<0, 8> = .BUCKET[BKT$B_CHECKCHAR];
            END
        ELSE IF .PTR[BLB$V_LOCK]
            THEN
            BEGIN
            BUCKET[BKT$B_CHECKCHAR] = .BUCKET[BKT$B_CHECKCHAR] + 1;
            (.BUCKET + .BDB[BDB$W_NUMB] - 1)<0, 8> = .BUCKET[BKT$B_CHECKCHAR];
            END;
        END;

    ! Use the level in the tree as the cache value of the bucket.
    ! If permanence was asked for, give it another boost.
    !

    BDB[BDB$B_CACHE_VAL] = .BUCKET[BKT$B_LEVEL];
    IF TESTBITSC (BDB[BDB$V_PRM])
    THEN
        BDB[BDB$B_CACHE_VAL] = .BDB[BDB$B_CACHE_VAL] + 1;

    END;

RETURN RM$RELEASE(.FLAGS);

END;
```

```
                                  .EXTRN  RM$BUG3

           0E  BB 00000 RM$RLSBKT::
                                  PUSHR   #^M<R1,R2,R3>                    ; 0431
           54  D5 00002           TSTL    BDB                              ; 0484
```

```
                          44  13 00004       BEQL    4$                                   ;
          10       08     A4  91 00006       CMPB    8(BDB), #16                          ;
                          3E  13 0000A       BEQL    4$                                   ;
          3A       0A     A4  E9 0000C       BLBC    10(BDB), 4$                          ; 0490
                   16     A4  B5 00010       TSTW    22(BDB)                              ; 0492
                          35  13 00013       BEQL    4$                                   ;
          52       18     A4  D0 00015       MOVL    24(BDB), BUCKET                      ; 0495
     1C   A4       02     A2  B1 00019       CMPW    2(BUCKET), 28(BDB)                   ; 0500
                          03  13 0001E       BEQL    1$                                   ;
                       0000G  30 00020       BSBW    RM$BUG3                              ; 0501
 15  0A   A4              01  E1 00023 1$:   BBC     #1, 10(BDB), 3$                      ; 0504
          50       10     A4  D0 00028       MOVL    16(BDB), PTR                         ; 0508
                          04  13 0002C       BEQL    2$                                   ; 0509
          0B       0A     A0  E9 0002E       BLBC    10(PTR), 3$                          ; 0515
                          62  96 00032 2$:   INCB    (BUCKET)                             ; 0518
          50       14     A4  3C 00034       MOVZWL  20(BDB), R0                          ; 0519
     FF A042              62  90 00038       MOVB    (BUCKET), -1(R0)[BUCKET]             ;
     0B   A4       0C     A2  90 0003D 3$:   MOVB    12(BUCKET), 11(BDB)                  ; 0527
 03  0A   A4              03  E5 00042       BBCC    #3, 10(BDB), 4$                      ; 0528
                   0B     A4  96 00047       INCB    11(BDB)                              ; 0530
          53       10     AE  D0 0004A 4$:   MOVL    FLAGS, R3                            ; 0534
                       0000G  30 0004E       BSBW    RM$RELEASE                           ;
                          0E  BA 00051       POPR    #^M<R1,R2,R3>                        ; 0536
                              05 00053       RSB                                          ;

; Routine Size:  84 bytes,    Routine Base:  RM$RMS3 + 00C7


;  475          0537  1
;  476          0538  1 END
;  477          0539  1
;  478          0540  0 ELUDOM
```

```
;                                PSECT SUMMARY
;
;      Name                   Bytes                        Attributes
;
;  RM$RMS3                          283  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  GBL,  REL,  CON,  PIC,ALIGN(2)


;                           Library Statistics
;
;                                          -------- Symbols --------        Pages      Processing
;      File                                Total    Loaded   Percent        Mapped     Time
;
;  _$255$DUA28:[RMS.OBJ]RMS.L32;1           3109       63          2          154       00:00.4
```

```
;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3BKTIO/OBJ=OBJ$:RM3BKTIO MSRC$:RM3BKTIO/UPDATE=(ENH$:RM3BKTIO)

; Size:         283 code + 0 data bytes
; Run Time:       00:09.6
; Elapsed Time:   00:22.1
; Lines/CPU Min:   3382
; Lexemes/CPU-Min: 18068
; Memory Used:  102 pages
; Compilation Complete
```

RM2UPDDEL
LIS

RM3CLOSE
LIS

RM3ALLBKT
LIS

RM2CREATE
LIS

RM2GET
LIS

RM2PUT
LIS

RM3BKTIO
LIS

RM3BKTSPL
LIS

RM3CMPKEY
LIS

RM3CMPRSS
LIS

RM2EXTEND
LIS

RM2FMTBKT
LIS

RM3BUG
LIS

RM2OPEN
LIS