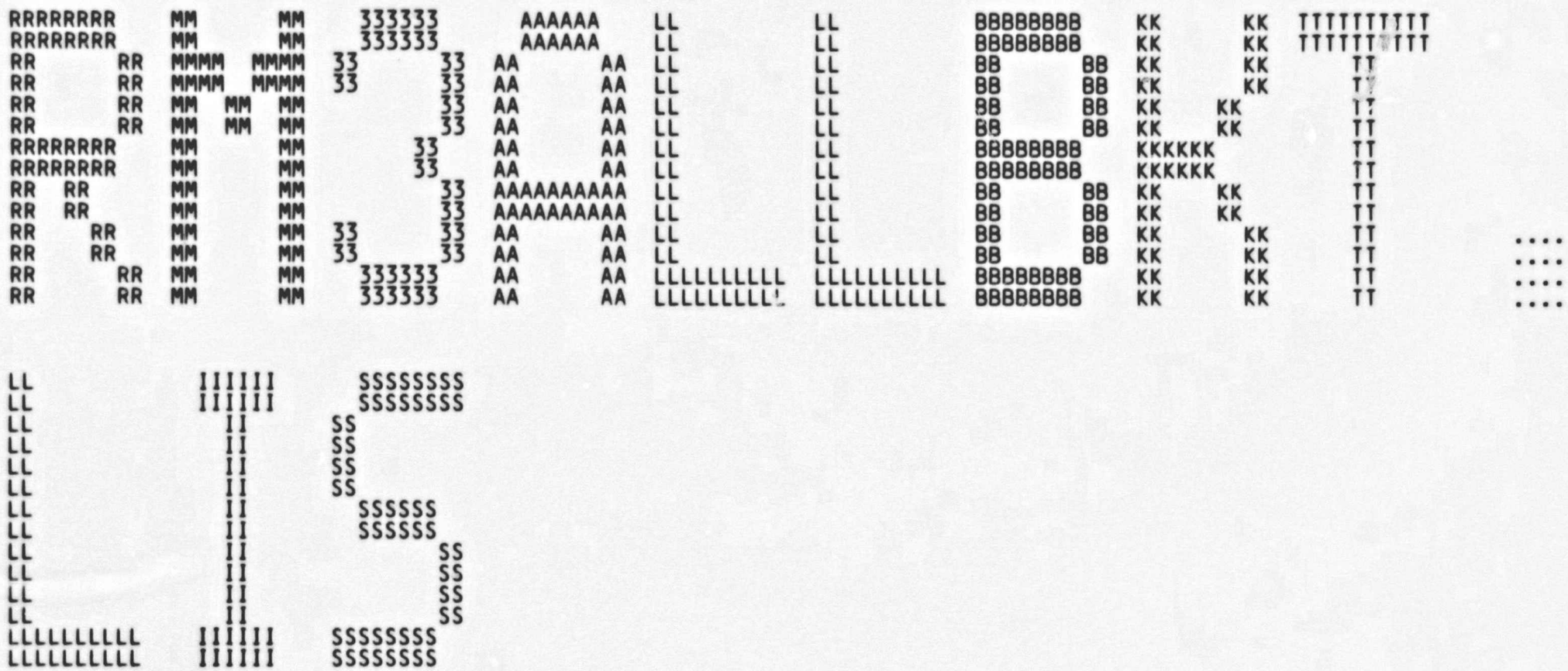
```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     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  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

FILEID**RM3ALLBKT



RM3
VO4

: R

```
;     1          0001  0 MODULE RM3ALLBKT (LANGUAGE (BLISS32) ,
;     2          0002  0                    IDENT = 'V04-000'
;     3          0003  0                    ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !**************************************************************************
;     7          0007  1 !*                                                                        *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                  *
;    11          0011  1 !*                                                                        *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17          0017  1 !*  TRANSFERRED.                                                           *
;    18          0018  1 !*                                                                        *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
;    21          0021  1 !*  CORPORATION.                                                           *
;    22          0022  1 !*                                                                        *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !*                                                                        *
;    27          0027  1 !**************************************************************************
```

```
;    29     0028  1 !++
;    30     0029  1 !
;    31     0030  1 ! FACILITY:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
;    32     0031  1 !
;    33     0032  1 ! ABSTRACT:
;    34     0033  1 !               This module handles the allocation of buckets and their formatting
;    35     0034  1 !
;    36     0035  1 !
;    37     0036  1 ! ENVIRONMENT:
;    38     0037  1 !
;    39     0038  1 !               VAX/VMS OPERATING SYSTEM
;    40     0039  1 !
;    41     0040  1 !--
;    42     0041  1
;    43     0042  1 !
;    44     0043  1 ! AUTHOR:        D. H. Gillespie                 28-Jul-1978
;    45     0044  1 !
;    46     0045  1 !
;    47     0046  1 ! MODIFIED BY:
;    48     0047  1 !
;    49     0048  1 !       V03-009 MCN0015         Maria del C. Nasr       04-Apr-1983
;    50     0049  1 !               Modify calling syntax to RM$ALLOC3 to match new linkage.
;    51     0050  1 !
;    52     0051  1 !       V03-008 TMK0002         Todd M. Katz            02-Apr-1983
;    53     0052  1 !               If this ISAM file is being BI Journalled, then in
;    54     0053  1 !               RM$AL_FRMT_BKT, after formatting the bucket, move the formatted
;    55     0054  1 !               portion into the BI Journal record buffer controlled by
;    56     0055  1 !               the BI BDB associated with the BDB for the new bucket.
;    57     0056  1 !
;    58     0057  1 !       V03-007 MCN0014         Maria del C. Nasr       31-Mar-1983
;    59     0058  1 !               More linkages reorganization
;    60     0059  1 !
;    61     0060  1 !       V03-006 MCN0013         Maria del C. Nasr       24-Feb-1983
;    62     0061  1 !               Reorganize linkages.
;    63     0062  1 !
;    64     0063  1 !       V03-005 KBT0488         Keith B. Thompson       2-Feb-1983
;    65     0064  1 !               Raise the file lock when doing an extend
;    66     0065  1 !
;    67     0066  1 !       V03-004 MCN0012         Maria del C. Nasr       08-Nov-1982
;    68     0067  1 !               Add new extended quantity to TOTAL_ALLOC field in
;    69     0068  1 !               the area descriptor.  Also clear AREA$L_NXBLK when
;    70     0069  1 !               the next extend is used.  (Routine GET_VBN)
;    71     0070  1 !
;    72     0071  1 !       V03-003 KBT0153         Keith B. Thompson       21-Aug-1982
;    73     0072  1 !               Reorganize psects
;    74     0073  1 !
;    75     0074  1 !       V03-002 KBT0111         Keith B. Thompson       6-Aug-1982
;    76     0075  1 !               No need to stuff the sifb anymore
;    77     0076  1 !
;    78     0077  1 !       V03-001 TMK0001         Todd M. Katz            12-Jun-1982
;    79     0078  1 !               Fix linkage bug by describing the external routine
;    80     0079  1 !               RM$CHKSUM with ADDRESSING_MODE(RELATIVE).
;    81     0080  1 !
;    82     0081  1 !       V02-010 TMK0001         Todd M. Katz            09-Feb-1982
;    83     0082  1 !               Fix linkage bug by describing the external routine
;    84     0083  1 !               RM$MAKESUM with ADDRESSING_MODE(RELATIVE).
;    85     0084  1 !
```

```
;    86          0085  1 !       V02-009 kpl0002         Peter Lieberwirth       23-Nov-1981
;    87          0086  1 !               Fix bug in v02-008.  Keep lock on area descriptor where
;    88          0087  1 !               commentary says to...
;    89          0088  1 !
;    90          0089  1 !       V02-008 kpl0001         Peter Lieberwirth       11-Aug-1981
;    91          0090  1 !               Add routines to reclaim buckets off the
;    92          0091  1 !               AVAIL list.  (space reclamation)
;    93          0092  1 !
;    94          0093  1 !       V02-007 MCN0011         Maria del C. Nasr       26-May-1981
;    95          0094  1 !               Add code to format prologue 3 buckets.
;    96          0095  1 !
;    97          0096  1 !       V02-006 REFORMAT        Frederick E. Deen, Jr.  23-Jul-1980
;    98          0097  1 !               This code was reformatted to adhere to RMS standards
;    99          0098  1 !
;   100          0099  1 !*****
;   101          0100  1
;   102          0101  1 LIBRARY 'RMSLIB:RMS';
;   103          0102  1
;   104          0103  1 REQUIRE 'RMSSRC:RMSIDXDEF';
;   105          0168  1
;   106          0169  1
;   107          0170  1 ! define default psects for code
;   108          0171  1 !
;   109          0172  1
;   110          0173  1 PSECT
;   111          0174  1     CODE = RM$RMS3(PSECT_ATTR),
;   112          0175  1     PLIT = RM$RMS3(PSECT_ATTR);
;   113          0176  1
;   114          0177  1 ! Linkage
;   115          0178  1 !
;   116          0179  1 LINKAGE
;   117          0180  1     L_ALLOC3,
;   118          0181  1     L_CACHE,
;   119          0182  1     L_CHKSUM,
;   120          0183  1     L_RABREG,
;   121          0184  1     L_RABREG_7,
;   122          0185  1     L_RELEASE,
;   123          0186  1
;   124          0187  1     RL$GET_BKT          = JSB () : GLOBAL (COMMON_RABREG, R_BDB, NEXT_RECID = 6),
;   125          0188  1     RL$FOR_REC_BKT      = JSB () : GLOBAL (COMMON_RABREG, R_IDX_DFN),
;   126          0189  1     RL$GET_VBN          = JSB () : GLOBAL (COMMON_RABREG, VBN = 6),
;   127          0190  1     RL$LOCK_AREA        = JSB () : GLOBAL (R_BDB, COMMON_RABREG, AREA_DESC = 7)
;   128          0191  1                                    NOPRESERVE (2, 3),
;   129          0192  1     RL$CHECK_FOR_RO     = JSB () : GLOBAL (COMMON_RABREG, R_BDB, VBN = 6, AREA_DESC = 7);
;   130          0193  1
;   131          0194  1 ! Forward Routines
;   132          0195  1 !
;   133          0196  1 FORWARD ROUTINE
;   134          0197  1         RM$LOCK_AREA    : RL$LOCK_AREA;
;   135          0198  1
;   136          0199  1 ! External Routines
;   137          0200  1 !
;   138          0201  1 EXTERNAL ROUTINE
;   139          0202  1         RM$ALLOC3       : RL$ALLOC3,
;   140          0203  1         RM$CACHE        : RL$CACHE,
;   141          0204  1         RM$CHKSUM       : RL$CHKSUM,
;   142          0205  1         RM$LOWER_LOCK   : RL$RABREG,
```

```
;  143          0206  1          RM$MAKSUM         : RL$CHKSUM,
;  144          0207  1          RM$RAISE_LOCK     : RL$RABREG,
;  145          0208  1          RM$RELEASE        : RL$RELEASE;
;  146          0209  1
```

```
;  148          0210  1 %SBTTL  'GET_BKT'
;  149          0211  1 ROUTINE GET_BKT ( AREA_NO ) : RL$GET_BKT =
;  150          0212  1
;  151          0213  1 !++
;  152          0214  1 !
;  153          0215  1 ! FUNCTIONAL DESCRIPTION:
;  154          0216  1 !
;  155          0217  1 !       This routine attempts to reclaim a bucket from the area AVAIL list.
;  156          0218  1 !       it updates the area descriptor and writes it out to the disk if
;  157          0219  1 !       necessary.  It is called only by RECLAIM_BKT.
;  158          0220  1 !
;  159          0221  1 ! CALLING SEQUENCE:
;  160          0222  1 !       GET_BKT (.AREA_NO);
;  161          0223  1 !
;  162          0224  1 ! INPUT PARAMETERS:
;  163          0225  1 !       AREA_NO - area from which to reclaim the bucket
;  164          0226  1 !
;  165          0227  1 ! IMPLICIT INPUTS:
;  166          0228  1 !       IRAB    - address of internal RAB
;  167          0229  1 !       IFAB    - address of IFAB needed for I/O and prologue version
;  168          0230  1 !       IMPURE  - address of impure region(needed for I/O)
;  169          0231  1 !       IDX_DFN - index descriptor to get key of reference
;  170          0232  1 !
;  171          0233  1 ! OUTPUT PARAMETERS:
;  172          0234  1 !       NEXT_RECID - value of lowest record id permitted for reclaimed bucket
;  173          0235  1 !
;  174          0236  1 ! IMPLICIT OUTPUTS:
;  175          0237  1 !       IRAB[IRB$L_NXTBDB] - address of BDB describing new reclaimed and
;  176          0238  1 !                            partially formatted bucket.
;  177          0239  1 !
;  178          0240  1 ! ROUTINE VALUE:
;  179          0241  1 !       various errors, including those from CACHE and RELEASE
;  180          0242  1 !
;  181          0243  1 ! SIDE EFFECTS:
;  182          0244  1 !       If there is a reclaimable bucket, it is reclaimed.
;  183          0245  1 !       The area descriptor is updated and written to the disk.
;  184          0246  1 !
;  185          0247  1 !--
;  186          0248  1
;  187          0249  2     BEGIN
;  188          0250  2
;  189          0251  2     EXTERNAL REGISTER
;  190          0252  2         NEXT_RECID = 6,
;  191          0253  2         R_BDB_STR,
;  192          0254  2         COMMON_RAB_STR;
;  193          0255  2
;  194          0256  2     GLOBAL REGISTER
;  195          0257  2         AREA_DESC = 7   : REF BBLOCK;
;  196          0258  2
;  197          0259  2     ! Lock the area descriptor
;  198          0260  2     !
;  199          0261  2     RETURN_ON_ERROR( RM$LOCK_AREA( .AREA_NO ));
;  200          0262  2
;  201          0263  2     ! Check the available list for reclaimable buckets
;  202          0264  2     !
;  203          0265  2     IF .AREA_DESC[AREA$L_AVAIL] NEQU 0
;  204          0266  2     THEN
```

```
;  205     0267  3          BEGIN
;  206     0268  3
;  207     0269  3          LOCAL
;  208     0270  3              BUCKET      : REF BBLOCK,
;  209     0271  3              AVAIL_VBN,
;  210     0272  3              NEXT_VBN;
;  211     0273  3
;  212     0274  4          BEGIN
;  213     0275  4
;  214     0276  4          GLOBAL REGISTER
;  215     0277  4              R_BKT_ADDR_STR;
;  216     0278  4
;  217     0279  4          ! Release the area descriptor with keep lock, so no one else tries
;  218     0280  4          ! to get our reclaimed VBN.
;  219     0281  4          !
;  220     0282  4          RETURN_ON_ERROR( RM$RELEASE(RLS$M_KEEP_LOCK) );
;  221     0283  4
;  222     0284  4          ! Remember the available bucket
;  223     0285  4          !
;  224     0286  4          AVAIL_VBN = .AREA_DESC[AREA$L_AVAIL];
;  225     0287  4
;  226     0288  4          ! Get the reclaimed bucket in the cache
;  227     0289  4          !
;  228   P 0290  4          RETURN_ON_ERROR( RM$CACHE( .AVAIL_VBN,
;  229   P 0291  4                                     .BBLOCK[.IRAB[IRB$L_CURBDB],BDB$W_NUMB],
;  230     0292  4                                    CSH$M_LOCK));
;  231     0293  4
;  232     0294  4          ! Get from the reclaimed bucket the VBN of the next reclaimable bucket
;  233     0295  4          ! and the lowest record ID usable for this incarnation of the bucket.
;  234     0296  4          !
;  235     0297  4          BUCKET = .BDB[BDB$L_ADDR];
;  236     0298  4
;  237     0299  4          NEXT_VBN = .BUCKET[BKT$L_NXTBKT];
;  238     0300  4
;  239     0301  4          IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  240     0302  4          THEN
;  241     0303  4              NEXT_RECID = .BUCKET[BKT$B_NXTRECID]
;  242     0304  4          ELSE
;  243     0305  4              NEXT_RECID = .BUCKET[BKT$W_NXTRECID];
;  244     0306  4
;  245     0307  3          END;                                    ! of global register definition
;  246     0308  3
;  247     0309  3          ! Release the reclaimed bucket, freeing space in the cache for the
;  248     0310  3          ! area descriptor again.
;  249     0311  3          !
;  250     0312  3          RETURN_ON_ERROR( RM$RELEASE(0) );
;  251     0313  3
;  252     0314  3          ! Get the area descriptor again
;  253     0315  3          !
;  254     0316  3          RETURN_ON_ERROR( RM$LOCK_AREA( .AREA_NO ));
;  255     0317  3
;  256     0318  3          ! Update the area descriptor avail listhead, calculate the new bucket
;  257     0319  3          ! checksum, and write the modified area descriptor back to disk.
;  258     0320  3          !
;  259     0321  3          ! ** DO WE NEED TO BACK OUT THE AREA DESCRIPTOR IF THE SPLIT FAILS? **
;  260     0322  3          !
;  261     0323  3          AREA_DESC[AREA$L_AVAIL] = .NEXT_VBN;
```

```
;  262          0324  3
;  263          0325  3        RM$MAKSUM( .BDB[BDB$L_ADDR] );
;  264          0326  3
;  265          0327  3        BDB[BDB$V_VAL] = 1;
;  266          0328  3        BDB[BDB$V_DRT] = 1;
;  267          0329  3
;  268          0330  3        RETURN_ON_ERROR( RM$RELEASE( RLS$M_WRT_THRU ));
;  269          0331  3
;  270          0332  4        BEGIN
;  271          0333  4
;  272          0334  4        GLOBAL REGISTER
;  273          0335  4            R_BKT_ADDR_STR;
;  274          0336  4
;  275          0337  4        ! Now, get a free buffer to format the new bucket.  Note that we
;  276          0338  4        ! don't need to read in the reclaimed bucket again because we saved
;  277          0339  4        ! everything we needed from it the last time it was here.
;  278          0340  4        !
;  279        P 0341  4        RETURN_ON_ERROR( RM$CACHE( .AVAIL_VBN,
;  280        P 0342  4                                   .BBLOCK[.IRAB[IRB$L_CURBDB],BDB$W_NUMB],
;  281          0343  4                                  CSH$M_NOREAD OR CSH$M_LOCK));
;  282          0344  4
;  283          0345  3        END;                                ! of global register definition
;  284          0346  3        IRAB[IRB$L_NXTBDB] = .BDB;
;  285          0347  3
;  286          0348  3        BDB[BDB$V_VAL] = 1;
;  287          0349  3
;  288          0350  3        RETURN 1;
;  289          0351  3
;  290          0352  3        END
;  291          0353  3
;  292          0354  2    ELSE
;  293          0355  2
;  294          0356  3        BEGIN
;  295          0357  3
;  296          0358  3        ! Release the area descriptor after all
;  297          0359  3        !
;  298          0360  3        RETURN_ON_ERROR( RM$RELEASE (0));
;  299          0361  3
;  300          0362  3        RETURN 0;
;  301          0363  3
;  302          0364  2        END;
;  303          0365  2
;  304          0366  1    END;


                                                .TITLE  RM3ALLBKT
                                                .IDENT  \V04-000\

                                                .EXTRN  RM$ALLOC3, RM$CACHE
                                                .EXTRN  RM$CHKSUM, RM$LOWER_LOCK
                                                .EXTRN  RM$MAKSUM, RM$RAISE_LOCK
                                                .EXTRN  RM$RELEASE

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

                    00AC   8F  BB 00000 GET_BKT:PUSHR   #^M<R2,R3,R5,R7>                ; 0211
                5E          04  C2 00004        SUBL2   #4, SP                          ;
```

```
                     18   AE   DD 00007          PUSHL   AREA_NO                                 ; 0261
                   0000V  30   0000A            BSBW    RM$LOCK_AREA                            ;
                5E      04   C0 0000D            ADDL2   #4, SP                                  ;
                7F      50   E9 00010            BLBC    STATUS, 4$                              ;
                   08   A7   D5 00013            TSTL    8(AREA_DESC)                            ; 0265
                        03   12 00016            BNEQ    1$                                      ;
                      0087   31 00018            BRW     6$                                      ;
                53      04   D0 0001B 1$:        MOVL    #4, R3                                  ; 0282
                     0000G   30 0001E            BSBW    RM$RELEASE                              ;
                7C      50   E9 00021            BLBC    STATUS, 5$                              ;
                6E   08   A7   D0 00024          MOVL    8(AREA_DESC), AVAIL_VBN                 ; 0286
                50   20   A9   D0 00028          MOVL    32(IRAB), R0                            ; 0292
                53      01   D0 0002C            MOVL    #1, R3                                  ;
                52   14   A0   3C 0002F          MOVZWL  20(R0), R2                              ;
                51      6E   D0 00033            MOVL    AVAIL_VBN, R1                           ;
                     0000G   30 00036            BSBW    RM$CACHE                                ;
                70      50   E9 00039            BLBC    STATUS, 7$                              ;
                50   18   A4   D0 0003C          MOVL    24(BDB), BUCKET                         ; 0297
                55   08   A0   D0 00040          MOVL    8(BUCKET), NEXT_VBN                     ; 0299
                03 00B7   CA   91 00044          CMPB    183(IFAB), #3                           ; 0301
                        06   1E 00049            BGEQU   2$                                      ;
                56   06   A0   9A 0004B          MOVZBL  6(BUCKET), NEXT_RECID                   ; 0303
                        04   11 0004F            BRB     3$                                      ;
                56   06   A0   3C 00051 2$:      MOVZWL  6(BUCKET), NEXT_RECID                   ; 0305
                53      D4   00055 3$:          CLRL    R3                                      ; 0312
                     0000G   30 00057            BSBW    RM$RELEASE                              ;
                4F      50   E9 0005A            BLBC    STATUS, 7$                              ;
                   18   AE   DD 0005D          PUSHL   AREA_NO                                 ; 0316
                   0000V   30 00060            BSBW    RM$LOCK_AREA                            ;
                5E      04   C0 00063            ADDL2   #4, SP                                  ;
                43      50   E9 00066            BLBC    STATUS, 7$                              ;
             08 A7      55   D0 00069          MOVL    NEXT_VBN, 8(AREA_DESC)                  ; 0323
                55   18   A4   D0 0006D          MOVL    24(BDB), R5                             ; 0325
                     0000G   30 00071            BSBW    RM$MAKSUM                               ;
             0A A4         03   88 00074        BISB2   #3, 10(BDB)                             ; 0328
                53      02   D0 00078            MOVL    #2, R3                                  ; 0330
                     0000G   30 0007B            BSBW    RM$RELEASE                              ;
                2B      50   E9 0007E            BLBC    STATUS, 7$                              ;
                50   20   A9   D0 00081          MOVL    32(IRAB), R0                            ; 0343
                53      05   D0 00085            MOVL    #5, R3                                  ;
                52   14   A0   3C 00088          MOVZWL  20(R0), R2                              ;
                51      6E   D0 0008C            MOVL    AVAIL_VBN, R1                           ;
                     0000G   30 0008F            BSBW    RM$CACHE                                ;
                17      50   E9 00092 4$:        BLBC    STATUS, 7$                              ;
             3C A9      54   D0 00095          MOVL    BDB, 60(IRAB)                           ; 0346
             0A A4         01   88 00099        BISB2   #1, 10(BDB)                             ; 0348
                50      01   D0 0009D            MOVL    #1, R0                                  ; 0356
                        0A   11 000A0 5$:        BRB     7$                                      ;
                53      D4   000A2 6$:          CLRL    R3                                      ; 0360
                     0000G   30 000A4            BSBW    RM$RELEASE                              ;
                02      50   E9 000A7            BLBC    STATUS, 7$                              ;
                50      D4   000AA              CLRL    R0                                      ; 0362
                5E      04   C0 000AC 7$:        ADDL2   #4, SP                                  ; 0366
                     00AC   8F   BA 000AF        POPR    #^M<R2,R3,R5,R7>                        ;
                            05   000B3          RSB                                             ;

; Routine Size:  180 bytes,    Routine Base:  RM$RMS3 + 0000
```

```
;  306     0367  1 %SBTTL  'FORMAT_BKT'
;  307     0368  1 ROUTINE FORMAT_BKT (AREA_NO, RECORD_ID, NO_BYTES) : RL$FOR_REC_BKT =
;  308     0369  1
;  309     0370  1 !++
;  310     0371  1 !
;  311     0372  1 ! FUNCTIONAL DESCRIPTION:
;  312     0373  1 !       This routine begins the bucket formatting process.  It is called by
;  313     0374  1 !       RECLAIM_BKT and AL_FRMT_BKT.
;  314     0375  1 !
;  315     0376  1 ! CALLING SEQUENCE:
;  316     0377  1 !       FORMAT_BKT( AREA_NO, RECORD_ID, NO_BYTES );
;  317     0378  1 !
;  318     0379  1 ! INPUT PARAMETERS:
;  319     0380  1 !       AREA_NO         -       area from which to reclaim the bucket
;  320     0381  1 !       RECORD_ID       -       lowest record ID allowable for this bucket
;  321     0382  1 !       NO_BYTES        -       size in bytes of bucket to format
;  322     0383  1 !
;  323     0384  1 ! IMPLICIT INPUTS:
;  324     0385  1 !       IFAB    - address of internal FAB
;  325     0386  1 !       IRAB    - address of internal RAB
;  326     0387  1 !       IDX_DFN - index descriptor to get key of reference
;  327     0388  1 !
;  328     0389  1 ! OUTPUT PARAMETERS:
;  329     0390  1 !       None
;  330     0391  1 !
;  331     0392  1 ! IMPLICIT OUTPUTS:
;  332     0393  1 !       None
;  333     0394  1 !
;  334     0395  1 ! ROUTINE VALUE:
;  335     0396  1 !       success
;  336     0397  1 !
;  337     0398  1 ! SIDE EFFECTS:
;  338     0399  1 !       bucket partially formatted (see code)
;  339     0400  1 !
;  340     0401  1 !--
;  341     0402  1     BEGIN
;  342     0403  2
;  343     0404  2     LOCAL
;  344     0405  2         BUCKET : REF BBLOCK;
;  345     0406  2
;  346     0407  2     EXTERNAL REGISTER
;  347     0408  2         COMMON_RAB_STR,
;  348     0409  2         R_IDX_DFN_STR;
;  349     0410  2
;  350     0411  2     GLOBAL REGISTER
;  351     0412  2         COMMON_IO_STR;
;  352     0413  2
;  353     0414  2     BUCKET = .BBLOCK[.IRAB[IRB$L_NXTBDB], BDB$L_ADDR];
;  354     0415  2
;  355     0416  2     CH$FILL(0, .NO_BYTES, .BUCKET);
;  356     0417  2
;  357     0418  2     BUCKET[BKT$W_ADRSAMPLE] = .(BBLOCK[.IRAB[IRB$L_NXTBDB], BDB$L_VBN])<0, 16>;
;  358     0419  2
;  359     0420  2     BUCKET[BKT$W_FREESPACE] = BKT$C_OVERHDSZ;
;  360     0421  2
;  361     0422  2     IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  362     0423  2
```

```
;  363      0424  2      THEN
;  364      0425  3          BEGIN
;  365      0426  3          BUCKET[BKT$B_AREANO] = .AREA_NO;                 ! area number is input
;  366      0427  3          BUCKET[BKT$B_NXTRECID] = .RECORD_ID;
;  367      0428  3          BUCKET[BKT$B_LSTRECID] = 255;
;  368      0429  3          END
;  369      0430  2      ELSE
;  370      0431  3          BEGIN
;  371      0432  3          BUCKET[BKT$B_INDEXNO] = .IDX_DFN[IDX$B_KEYREF];
;  372      0433  3          BUCKET[BKT$W_NXTRECID] = .RECORD_ID;
;  373      0434  2          END;
;  374      0435  2
;  375      0436  2      RETURN 1;
;  376      0437  2
;  377      0438  1      END;



                              007C   8F   BB 00000 FORMAT_BKT:
                                                         PUSHR   #^M<R2,R3,R4,R5,R6>                       ; 0368
                               3C    A9   DD 00004       PUSHL   60(IRAB)                                  ; 0415
           50          6E            18   C1 00007       ADDL3   #24, (SP), R0                             ;
                       56            60   D0 0000B       MOVL    (R0), BUCKET                              ;
   24  AE  00          6E            00   2C 0000E       MOVC5   #0, (SP), #0, NO_BYTES, (BUCKET)          ; 0417
                       66                    00014                                                         ;
           50          6E            1C   C1 00015       ADDL3   #28, (SP), R0                             ; 0419
                  02   A6            60   B0 00019       MOVW    (R0), 2(BUCKET)                           ;
                  04   A6            0E   B0 0001D       MOVW    #14, 4(BUCKET)                            ; 0421
                       03     00B7   CA   91 00021       CMPB    183(IFAB), #3                             ; 0423
                                     10   1E 00026       BGEQU   1$                                        ;
                  01   A6      1C    AE   90 00028       MOVB    AREA_NO, 1(BUCKET)                        ; 0426
                  06   A6      20    AE   90 0002D       MOVB    RECORD_ID, 6(BUCKET)                      ; 0427
                  07   A6            01   8E 00032       MNEGB   #1, 7(BUCKET)                             ; 0428
                                     0A   11 00036       BRB     2$                                        ; 0423
                  01   A6      21    A7   90 00038 1$:   MOVB    33(IDX_DFN), 1(BUCKET)                    ; 0432
                  06   A6      20    AE   B0 0003D       MOVW    RECORD_ID, 6(BUCKET)                      ; 0433
                       50            01   D0 00042 2$:   MOVL    #1, R0                                    ; 0436
                       5E            04   C0 00045       ADDL2   #4, SP                                    ; 0438
                              007C   8F   BA 00048       POPR    #^M<R2,R3,R4,R5,R6>                       ;
                                          05 0004C       RSB                                               ;

; Routine Size:  77 bytes,    Routine Base:  RM$RMS3 + 00B4
```

```
;  379     0439  1 %SBTTL  'RECLAIM_BKT'
;  380     0440  1 ROUTINE RECLAIM_BKT (AREA_NO) : RL$FOR_REC_BKT =
;  381     0441  1
;  382     0442  1 !++
;  383     0443  1 !
;  384     0444  1 ! FUNCTIONAL DESCRIPTION:
;  385     0445  1 !       This routine serves as the high level control routine for bucket
;  386     0446  1 !       reclamation.  It calls the routine GET_BKT, which handles reading
;  387     0447  1 !       and updating the area descriptor if necessary.  This requires
;  388     0448  1 !       that the reclaimed bucket be read, in order to get the VBN of
;  389     0449  1 !       the next reclaimable VBN.  Finally, FORMAT_BKT is called to begin
;  390     0450  1 !       the process of formatting the bucket.
;  391     0451  1 !
;  392     0452  1 !       Note that this code will run faster if at least three buffers are
;  393     0453  1 !       available.  One is used for CURBDB, one for the area descriptor,
;  394     0454  1 !       and one for NXTBDB.  CURBDB represents the bucket being split,
;  395     0455  1 !       and NXTBDB represents the new bucket being split into.
;  396     0456  1 !
;  397     0457  1 ! CALLING SEQUENCE:
;  398     0458  1 !       RECLAIM_BKT( .AREA_NO );
;  399     0459  1 !
;  400     0460  1 ! INPUT PARAMETERS:
;  401     0461  1 !       AREA_NO - area from which to reclaim the bucket
;  402     0462  1 !
;  403     0463  1 ! IMPLICIT INPUTS:
;  404     0464  1 !       IRAB    - address of internal RAB
;  405     0465  1 !       IFAB    - address of IFAB needed for I/O and prologue version
;  406     0466  1 !       IMPURE  - address of impure region(needed for I/O)
;  407     0467  1 !       IDX_DFN - index descriptor to get key of reference
;  408     0468  1 !
;  409     0469  1 ! OUTPUT PARAMETERS:
;  410     0470  1 !       None
;  411     0471  1 !
;  412     0472  1 ! IMPLICIT OUTPUTS:
;  413     0473  1 !       IRAB[IRB$L_NXTBDB] - address of BDB describing new reclaimed and
;  414     0474  1 !                            partially formatted bucket.
;  415     0475  1 !
;  416     0476  1 ! ROUTINE VALUE:
;  417     0477  1 !       success or failure
;  418     0478  1 !
;  419     0479  1 ! SIDE EFFECTS:
;  420     0480  1 !       None, see GET_BKT.
;  421     0481  1 !
;  422     0482  1 !--
;  423     0483  1
;  424     0484  2     BEGIN
;  425     0485  2
;  426     0486  2     EXTERNAL REGISTER
;  427     0487  2         R_IDX_DFN_STR,
;  428     0488  2         COMMON_RAB_STR;
;  429     0489  2
;  430     0490  2     GLOBAL REGISTER
;  431     0491  2         COMMON_IO_STR,
;  432     0492  2         NEXT_RECID = 6;
;  433     0493  2
;  434     0494  2     LOCAL
;  435     0495  2         STATUS;
```

```
;  436     0496  2
;  437     0497  2      ! Go try to reclaim a bucket
;  438     0498  2      !
;  439     0499  2      STATUS = GET_BKT( .AREA_NO );
;  440     0500  2
;  441     0501  2      IF .STATUS
;  442     0502  2      THEN
;  443     0503  2          BEGIN
;  444     0504  2
;  445     0505  2          ! Format the new bucket and return to caller
;  446     0506  2          !
;  447     0507  2          FORMAT_BKT( .AREA_NO, .NEXT_RECID,
;  448     0508  2                      .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$W_NUMB] );
;  449     0509  2
;  450     0510  2          RETURN 1;
;  451     0511  2
;  452     0512  2          END
;  453     0513  2      ELSE
;  454     0514  2          RETURN .STATUS;
;  455     0515  2
;  456     0516  2
;  457     0517  1      END;


                        0050   8F  BB 00000 RECLAIM_BKT:
                                             PUSHR   #^M<R4,R6>                      ; 0440
                          0C    AE  DD 00004           PUSHL   AREA_NO               ; 0499
                              FEF5  30 00007           BSBW    GET_BKT               ;
                     5E             04  C0 0000A       ADDL2   #4, SP                ;
                     15             50  E9 0000D       BLBC    STATUS, 1$            ; 0501
                     50        20   A9  D0 00010       MOVL    32(IRAB), R0          ; 0508
                     7E        14   A0  3C 00014       MOVZWL  20(R0), -(SP)         ;
                                    56  DD 00018       PUSHL   NEXT_RECID            ; 0507
                              14    AE  DD 0001A       PUSHL   AREA_NO               ;
                                    94  10 0001D       BSBB    FORMAT_BKT            ;
                     5E             0C  C0 0001F       ADDL2   #12, SP               ;
                     50             01  D0 00022       MOVL    #1, R0                ; 0514
                                 0050   8F  BA 00025 1$:    POPR    #^M<R4,R6>        ; 0517
                                            05 00029       RSB                       ;

; Routine Size:  42 bytes,    Routine Base:  RM$RMS3 + 0101
```

```
;  459          0518  1 %SBTTL  'CHECK_FOR_ROOM'
;  460          0519  1 ROUTINE CHECK_FOR_ROOM : RL$CHECK_FOR_RO =
;  461          0520  1
;  462          0521  1 !++
;  463          0522  1 !
;  464          0523  1 ! FUNCTIONAL DESCRIPTION:
;  465          0524  1 !       This routine checks for room in current extent. If there is room,
;  466          0525  1 !       it updates the area descriptor and writes it out to the disk returning
;  467          0526  1 !       the VBN to use.
;  468          0527  1 !
;  469          0528  1 ! CALLING SEQUENCE:
;  470          0529  1 !       CHECK_FOR_ROOM()
;  471          0530  1 !
;  472          0531  1 ! INPUT PARAMETERS:
;  473          0532  1 !       None
;  474          0533  1 !
;  475          0534  1 ! IMPLICIT INPUTS:
;  476          0535  1 !       BDB             - address of BDB for area descriptor's buffer
;  477          0536  1 !       AREA_DESC       - address in memory of area descriptor
;  478          0537  1 !
;  479          0538  1 ! OUTPUT PARAMETERS:
;  480          0539  1 !       None
;  481          0540  1 !
;  482          0541  1 ! IMPLICIT OUTPUTS:
;  483          0542  1 !       VBN             - number of VBN to use for bucket
;  484          0543  1 !                         if equal to zero, no room
;  485          0544  1 !
;  486          0545  1 ! ROUTINE VALUE:
;  487          0546  1 !       various errors from RM$RELEASE
;  488          0547  1 !
;  489          0548  1 ! SIDE EFFECTS:
;  490          0549  1 !       If there is room in current extention,
;  491          0550  1 !       the area descriptor is updated and written to the disk.
;  492          0551  1 !
;  493          0552  1 !--
;  494          0553  1
;  495          0554  2     BEGIN
;  496          0555  2
;  497          0556  2     LOCAL
;  498          0557  2         NO_VBN;
;  499          0558  2
;  500          0559  2     EXTERNAL REGISTER
;  501          0560  2         COMMON_RAB_STR,
;  502          0561  2         R_BDB_STR,
;  503          0562  2         AREA_DESC = 7   : REF BBLOCK,
;  504          0563  2         VBN = 6;
;  505          0564  2
;  506          0565  2     NO_VBN = .AREA_DESC[AREA$B_ARBKTSZ];        ! number of blocks needed
;  507          0566  2     VBN = 0;
;  508          0567  2
;  509          0568  2     ! There is room when the number used and the number needed is less than
;  510          0569  2     ! or equal to total # in extent.
;  511          0570  2     !
;  512          0571  2
;  513          0572  2     IF (.AREA_DESC[AREA$L_USED] + .NO_VBN) LEQU .AREA_DESC[AREA$L_CNBLK]
;  514          0573  2     THEN
;  515          0574  3         BEGIN
```

```
;  516          0575  3
;  517          0576  3          ! update # used
;  518          0577  3          !
;  519          0578  3          AREA_DESC[AREA$L_USED] = .AREA_DESC[AREA$L_USED] + .NO_VBN;
;  520          0579  3          VBN = .AREA_DESC[AREA$L_NXTVBN];          ! VBN to be used
;  521          0580  3
;  522          0581  3          ! update next vbn to use
;  523          0582  3          !
;  524          0583  3          AREA_DESC[AREA$L_NXTVBN] = .AREA_DESC[AREA$L_NXTVBN] + .NO_VBN;
;  525          0584  3          RM$MAKSUM(.BDB[BDB$L_ADDR]);              ! recalculate checksum
;  526          0585  3          BDB[BDB$V_VAL] = 1;                ! write out updated area descriptor
;  527          0586  3          BDB[BDB$V_DRT] = 1;
;  528          0587  3
;  529          0588  3          RETURN RM$RELEASE( RLS$M_WRT_THRU )
;  530          0589  3
;  531          0590  2          END;
;  532          0591  2
;  533          0592  2      RETURN 1
;  534          0593  2
;  535          0594  1      END;                                  ! end of routine CHECK_FOR_ROOM



                                    2C  BB 00000 CHECK_FOR_ROOM:
                                                 PUSHR   #^M<R2,R3,R5>                          ; 0519
                          51        03    A7  9A 00002           MOVZBL  3(AREA_DESC), NO_VBN   ; 0565
                                          56  D4 00006           CLRL    VBN                    ; 0566
                50        51        14    A7  C1 00008           ADDL3   20(AREA_DESC), NO_VBN, R0 ; 0572
                      10  A7              50  D1 0000D           CMPL    R0, 16(AREA_DESC)      ;
                                          1F  1A 00011           BGTRU   1$                     ;
                      14  A7              51  C0 00013           ADDL2   NO_VBN, 20(AREA_DESC)  ; 0578
                          56        18    A7  D0 00017           MOVL    24(AREA_DESC), VBN     ; 0579
                      18  A7              51  C0 0001B           ADDL2   NO_VBN, 24(AREA_DESC)  ; 0583
                          55        18    A4  D0 0001F           MOVL    24(BDB), R5            ; 0584
                                    0000G 30  00023           BSBW    RM$MAKSUM              ;
                      0A  A4              03  88 00026           BISB2   #3, 10(BDB)            ; 0586
                          53              02  D0 0002A           MOVL    #2, R3                 ; 0588
                                    0000G 30  0002D           BSBW    RM$RELEASE             ;
                                          03  11 00030           BRB     2$                     ;
                          50              01  D0 00032 1$:       MOVL    #1, R0                 ; 0592
                                          2C  BA 00035 2$:       POPR    #^M<R2,R3,R5>          ; 0594
                                              05 00037           RSB                            ;

; Routine Size:  56 bytes,    Routine Base:  RM$RMS3 + 012B

;  536          0595  1
```

```
;  538     0596  1 %SBTTL  'GET_VBN'
;  539     0597  1 ROUTINE GET_VBN (AREA_NO) : RL$GET_VBN =
;  540     0598  1
;  541     0599  1 !++
;  542     0600  1 !
;  543     0601  1 ! FUNCTIONAL DESCRIPTION:
;  544     0602  1 !
;  545     0603  1 !       This routine attempts to allocate VBN's from the current extent, next it
;  546     0604  1 !       tries the next extents and then lastly extends the file.
;  547     0605  1 !
;  548     0606  1 ! CALLING SEQUENCE:
;  549     0607  1 !
;  550     0608  1 !       GET_VBN(AREA_NO)
;  551     0609  1 !
;  552     0610  1 ! INPUT PARAMETERS:
;  553     0611  1 !
;  554     0612  1 !       AREA_NO         - area number to allocate VBN from
;  555     0613  1 !
;  556     0614  1 ! IMPLICIT INPUTS:
;  557     0615  1 !
;  558     0616  1 !       IRAB            - address of internal RAB
;  559     0617  1 !
;  560     0618  1 ! OUTPUT PARAMETERS:
;  561     0619  1 !       None
;  562     0620  1 !
;  563     0621  1 ! IMPLICIT OUTPUTS:
;  564     0622  1 !
;  565     0623  1 !       VBN - VBN to use for bucket
;  566     0624  1 !
;  567     0625  1 ! ROUTINE VALUE:
;  568     0626  1 !       None
;  569     0627  1 !
;  570     0628  1 ! SIDE EFFECTS:
;  571     0629  1 !       None
;  572     0630  1 !
;  573     0631  1 !--
;  574     0632  1
;  575     0633  2     BEGIN
;  576     0634  2
;  577     0635  2     LOCAL
;  578     0636  2         STATUS;
;  579     0637  2
;  580     0638  2     GLOBAL REGISTER
;  581     0639  2         R_BDB_STR,
;  582     0640  2         AREA_DESC = 7   : REF BBLOCK;
;  583     0641  2
;  584     0642  2     EXTERNAL REGISTER
;  585     0643  2         COMMON_RAB_STR,
;  586     0644  2         VBN = 6;
;  587     0645  2
;  588     0646  2     IF .IRAB [ IRB$L_NXTBDB ] NEQ 0
;  589     0647  2     THEN
;  590     0648  2         RETURN RMSERR( BUG );
;  591     0649  2
;  592     0650  2     RETURN_ON_ERROR( RM$LOCK_AREA( .AREA_NO ) );
;  593     0651  2
;  594     0652  2     ! Examine the primary and secondary extents
```

```
;  595     0653  2     !
;  596     0654  2     WHILE 1
;  597     0655  2     DO
;  598     0656  2         BEGIN
;  599     0657  2
;  600     0658  2         RETURN_ON_ERROR( CHECK_FOR_ROOM() );
;  601     0659  2
;  602     0660  2         IF .VBN NEQ 0
;  603     0661  2         THEN
;  604     0662  2             RETURN 1;
;  605     0663  2
;  606     0664  2         IF .AREA_DESC [ AREA$L_NXT ] EQL 0
;  607     0665  2         THEN
;  608     0666  2             EXITLOOP;
;  609     0667  2
;  610     0668  2         AREA_DESC [ AREA$L_CVBN ]      = .AREA_DESC [ AREA$L_NXT ];
;  611     0669  2         AREA_DESC [ AREA$L_CNBLK ]     = .AREA_DESC [ AREA$L_NXBLK ];
;  612     0670  2         AREA_DESC [ AREA$L_USED ]      = 0;
;  613     0671  2         AREA_DESC [ AREA$L_NXTVBN ]    = .AREA_DESC [ AREA$L_CVBN ];
;  614     0672  2         AREA_DESC [ AREA$L_NXT ]       = 0;
;  615     0673  2         AREA_DESC [ AREA$L_NXBLK ]     = 0;
;  616     0674  2
;  617     0675  2         END;
;  618     0676  2
;  619     0677  2     ! The file must be extended.  Lock VBN 1.
;  620     0678  2     !
;  621     0679  2     BEGIN
;  622     0680  2
;  623     0681  2     LOCAL
;  624     0682  2         SAV_BDB;
;  625     0683  2
;  626     0684  2     GLOBAL REGISTER
;  627     0685  2         R_BKT_ADDR_STR;
;  628     0686  2
;  629     0687  2     ! Raise the file lock to kick out people doing file operations
;  630     0688  2     !
;  631     0689  2     RETURN_ON_ERROR( RM$RAISE_LOCK() );
;  632     0690  2
;  633     0691  2     SAV_BDB = .BDB;
;  634     0692  2
;  635   P 0693  2     RETURN_ON_ERROR( RM$CACHE(1, 0,
;  636   P 0694  2                               CSH$M_NOREAD
;  637   P 0695  2                               OR
;  638   P 0696  2                               CSH$M_LOCK
;  639   P 0697  2                               OR
;  640     0698  2                               CSH$M_NOBUFFER ) );
;  641     0699  2
;  642     0700  2     IRAB [ IRB$L_NXTBDB ] = .BDB;
;  643     0701  2     BDB = .SAV_BDB
;  644     0702  2
;  645     0703  2     END;                            ! end of local definition of SAV_BDB
;  646     0704  2
;  647     0705  2     BEGIN
;  648     0706  2
;  649     0707  2     LOCAL
;  650     0708  2         STARTVBN,
;  651     0709  3         ENDVBNP1;
```

```
;  652     0710  3
;  653     0711  3      ! Do the extend
;  654     0712  3      !
;  655     0713  3      IF STATUS = RM$ALLOC3 ( .AREA_DESC; STARTVBN, ENDVBNP1 )
;  656     0714  3      THEN
;  657     0715  4          BEGIN
;  658     0716  4
;  659     0717  4          ! Update EOF block of file attributes
;  660     0718  4          !
;  661     0719  4          IFAB [ IFB$L_EBK ] = .ENDVBNP1;
;  662     0720  4          IFAB [ IFB$L_HBK ] = .ENDVBNP1 - 1;
;  663     0721  4
;  664     0722  4          ! To keep the total allocation correct, we must distinguish between
;  665     0723  4          ! files that did not have this field defined when they were created,
;  666     0724  4          ! and those that did.  If the TOTAL_ALLOC field is not zero, then it
;  667     0725  4          ! is a new file.  If it is zero, check the VBN of the extents.  If
;  668     0726  4          ! they are both zero, then the area has never been extended, and the
;  669     0727  4          ! allocation can be computed correctly.
;  670     0728  4          !
;  671     0729  4
;  672     0730  4          IF .AREA_DESC [AREA$L_TOTAL_ALLOC] NEQ 0
;  673     0731  5            OR ( .AREA_DESC [AREA$L_TOTAL_ALLOC] EQL 0
;  674     0732  5              AND .AREA_DESC [AREA$L_CVBN] EQL 0
;  675     0733  5                AND .AREA_DESC [AREA$L_NXTVBN] EQL 0 )
;  676     0734  4          THEN
;  677     0735  4              AREA_DESC [AREA$L_TOTAL_ALLOC] = .AREA_DESC [AREA$L_TOTAL_ALLOC]
;  678     0736  4                      + ( .ENDVBNP1 - .STARTVBN );
;  679     0737  4
;  680     0738  4          ! Update the rest of the area descriptor.
;  681     0739  4          !
;  682     0740  4          AREA_DESC [ AREA$L_CNBLK ]      = .ENDVBNP1 - .STARTVBN;
;  683     0741  4          AREA_DESC [ AREA$L_USED ]       = 0;
;  684     0742  4          AREA_DESC [ AREA$L_CVBN ]       = .STARTVBN;
;  685     0743  4          AREA_DESC [ AREA$L_NXTVBN ]     = .STARTVBN;
;  686     0744  4          AREA_DESC [ AREA$L_NXT ]        = 0;
;  687     0745  4          AREA_DESC [ AREA$L_NXBLK ]      = 0;
;  688     0746  4
;  689     0747  4          STATUS = CHECK_FOR_ROOM();
;  690     0748  4
;  691     0749  5          IF .STATUS AND ( .VBN EQL 0 )
;  692     0750  4          THEN
;  693     0751  5              BEGIN
;  694     0752  5              RM$RELEASE(0);
;  695     0753  5              STATUS = RMSERR(FUL);
;  696     0754  4              END;
;  697     0755  4
;  698     0756  4          END
;  699     0757  3      ELSE
;  700     0758  3          RM$RELEASE( 0 )
;  701     0759  3
;  702     0760  2      END;
;  703     0761  2
;  704     0762  2      ! If caller doesn't have VBN 1 locked for it's own usage, then unlock it.
;  705     0763  2      !
;  706     0764  2      BDB = .IRAB [ IRB$L_NXTBDB ];
;  707     0765  2      IRAB [ IRB$L_NXTBDB ] = 0;
;  708     0766  2
```

```
;  709          0767  2     IF .BDB NEQ .IRAB [ IRB$L_LOCK_BDB ]
;  710          0768  2     THEN
;  711          0769  2         RM$RELEASE( 0 );
;  712          0770  2
;  713          0771  2     ! If everything worked then lower the file lock (if they didn't then
;  714          0772  2     ! we are on the way out anyway
;  715          0773  2     !
;  716          0774  2     IF .STATUS
;  717          0775  2     THEN
;  718          0776  2         STATUS = RM$LOWER_LOCK();
;  719          0777  2
;  720          0778  2     RETURN .STATUS
;  721          0779  2
;  722          0780  1     END;


                        00BC    8F   BB 00000 GET_VBN:PUSHR   #^M<R2,R3,R4,R5,R7>                ; 0597
                     5E           04   C2 00004          SUBL2   #4, SP                          ;
                        3C    A9   D5 00007          TSTL    60(IRAB)                        ; 0646
                              07   13 0000A          BEQL    1$                              ;
                     50     8434   8F   3C 0000C          MOVZWL  #33844, R0                      ; 0648
                              18   11 00011          BRB     3$                              ;
                        1C    AE   DD 00013 1$:      PUSHL   AREA_NO                         ; 0650
                           0000V   30 00016          BSBW    RM$LOCK_AREA                    ;
                     5E           04   C0 00019          ADDL2   #4, SP                          ;
                     38           50   E9 0001C          BLBC    STATUS, 6$                      ;
                                  A7   10 0001F 2$:      BSBB    CHECK_FOR_ROOM                  ; 0658
                     33           50   E9 00021          BLBC    STATUS, 6$                      ;
                                  56   D5 00024          TSTL    VBN                             ; 0660
                                  06   13 00026          BEQL    4$                              ;
                     50           01   D0 00028          MOVL    #1, R0                          ; 0662
                              00AF   31 0002B 3$:       BRW     13$                             ;
                           1C    A7   D5 0002E 4$:      TSTL    28(AREA_DESC)                   ; 0664
                                 12   13 00031          BEQL    5$                              ;
                  0C    A7       1C    A7   7D 00033          MOVQ    28(AREA_DESC), 12(AREA_DESC)    ; 0668
                                 14    A7   D4 00038          CLRL    20(AREA_DESC)                   ; 0670
                  18    A7       0C    A7   D0 0003B          MOVL    12(AREA_DESC), 24(AREA_DESC)    ; 0671
                                 1C    A7   7C 00040          CLRQ    28(AREA_DESC)                   ; 0672
                                       DA   11 00043          BRB     2$                              ; 0654
                                    0000G   30 00045 5$:      BSBW    RM$RAISE_LOCK                   ; 0689
                        E0           50   E9 00048          BLBC    STATUS, 3$                      ;
                        6E           54   D0 0004B          MOVL    BDB, SAV_BDB                    ; 0691
                        53           0D   D0 0004E          MOVL    #13, R3                         ; 0698
                        51           01   7D 00051          MOVQ    #1, R1                          ;
                                    0000G   30 00054          BSBW    RM$CACHE                        ;
                        D1           50   E9 00057 6$:      BLBC    STATUS, 3$                      ;
                  3C    A9           54   D0 0005A          MOVL    BDB, 60(IRAB)                   ; 0700
                        54           6E   D0 0005E          MOVL    SAV_BDB, BDB                    ; 0701
                                    0000G   30 00061          BSBW    RM$ALLOC3                       ; 0713
                        55           50   D0 00064          MOVL    R0, STATUS                      ;
                        53           52   D0 00067          MOVL    R2, R3                          ;
                        4C           55   E9 0006A          BLBC    STATUS, 9$                      ;
                  74    AA           53   D0 0006D          MOVL    ENDVBNP1, 116(IFAB)             ; 0719
                  70    AA     FF    A3   9E 00071          MOVAB   -1(R3), 112(IFAB)               ; 0720
```

```
                                32  A7  D5 00076          TSTL    50(AREA_DESC)                          ; 0730
                                    0A  12 00079          BNEQ    7$                                     ;
                                0C  A7  D5 0007B          TSTL    12(AREA_DESC)                          ; 0732
                                    0D  12 0007E          BNEQ    8$                                     ;
                                18  A7  D5 00080          TSTL    24(AREA_DESC)                          ; 0733
                                    08  12 00083          BNEQ    8$                                     ;
            52          53          51  C3 00085 7$:      SUBL3   STARTVBN, ENDVBNP1, R2                 ; 0736
                    32  A7          52  C0 00089          ADDL2   R2, 50(AREA_DESC)                      ;
       10   A7          53          51  C3 0008D 8$:      SUBL3   STARTVBN, ENDVBNP1, 16(AREA_DESC)      ; 0740
                                14  A7  D4 00092          CLRL    20(AREA_DESC)                          ; 0741
                    0C  A7          51  D0 00095          MOVL    STARTVBN, 12(AREA_DESC)                ; 0742
                    18  A7          51  D0 00099          MOVL    STARTVBN, 24(AREA_DESC)                ; 0743
                                1C  A7  7C 0009D          CLRQ    28(AREA_DESC)                          ; 0744
                                  FF25  30 000A0          BSBW    CHECK_FOR_ROOM                         ; 0747
                        55          50  D0 000A3          MOVL    R0, STATUS                             ;
                        15          55  E9 000A6          BLBC    STATUS, 10$                            ; 0749
                                    56  D5 000A9          TSTL    VBN                                    ;
                                    11  12 000AB          BNEQ    10$                                    ;
                                    53  D4 000AD          CLRL    R3                                     ; 0752
                                 0000G  30 000AF          BSBW    RM$RELEASE                             ;
                        55    8544  8F  3C 000B2          MOVZWL  #34116, STATUS                         ; 0753
                                    05  11 000B7          BRB     10$                                    ; 0713
                                    53  D4 000B9 9$:      CLRL    R3                                     ; 0758
                                 0000G  30 000BB          BSBW    RM$RELEASE                             ;
                        54      3C  A9  D0 000BE 10$:     MOVL    60(IRAB), BDB                          ; 0764
                                3C  A9  D4 000C2          CLRL    60(IRAB)                               ; 0765
                  0084  C9          54  D1 000C5          CMPL    BDB, 132(IRAB)                         ; 0767
                                    05  13 000CA          BEQL    11$                                    ;
                                    53  D4 000CC          CLRL    R3                                     ; 0769
                                 0000G  30 000CE          BSBW    RM$RELEASE                             ;
                        06          55  E9 000D1 11$:     BLBC    STATUS, 12$                            ; 0774
                                 0000G  30 000D4          BSBW    RM$LOWER_LOCK                          ; 0776
                        55          50  D0 000D7          MOVL    R0, STATUS                             ;
                        50          55  D0 000DA 12$:     MOVL    STATUS, R0                             ; 0778
                        5E          04  C0 000DD 13$:     ADDL2   #4, SP                                 ; 0780
                          00BC      8F  BA 000E0          POPR    #^M<R2,R3,R4,R5,R7>                    ;
                                    05     000E4          RSB                                            ;

; Routine Size:  229 bytes,    Routine Base:  RM$RMS3 + 0163


;  723          0781  1
```

```
;  725          0782  1 %SBTTL  'RM$AL_FRMT_BKT'
;  726          0783  1 GLOBAL ROUTINE RM$AL_FRMT_BKT (AREA_NO, NO_BYTES) : RL$RABREG_7 =
;  727          0784  1
;  728          0785  1 !++
;  729          0786  1 !
;  730          0787  1 ! FUNCTIONAL DESCRIPTION:
;  731          0788  1 !       This routine gets a bucket allocated in the given area.  The bucket is
;  732          0789  1 !       cleared for its entire length and then the basic formatting is done.
;  733          0790  1 !
;  734          0791  1 ! CALLING SEQUENCE:
;  735          0792  1 !       RM$AL_FRMT_BKT(AREA_NO, NO_BYTES)
;  736          0793  1 !
;  737          0794  1 ! INPUT PARAMETERS:
;  738          0795  1 !       AREA_NO - area number to allocate bucket from
;  739          0796  1 !       NO_BYTES - number of bytes in bucket
;  740          0797  1 !
;  741          0798  1 ! IMPLICIT INPUTS:
;  742          0799  1 !       IRAB    - address of internal RAB
;  743          0800  1 !       RAB     - address of user RAB
;  744          0801  1 !       IFAB    - address of IFAB needed for I/O and prologue version
;  745          0802  1 !       IMPURE  - address of impure region(needed for I/O)
;  746          0803  1 !       IDX_DFN - index descriptor to get key of reference
;  747          0804  1 !
;  748          0805  1 ! OUTPUT PARAMETERS:
;  749          0806  1 !       None
;  750          0807  1 !
;  751          0808  1 ! IMPLICIT OUTPUTS:
;  752          0809  1 !       IRAB[IRB$L_NXTBDB] - address of BDB describing newly allocated and
;  753          0810  1 !                            partially formatted bucket
;  754          0811  1 !
;  755          0812  1 ! ROUTINE VALUE:
;  756          0813  1 !       Various I/O errors, extend errors
;  757          0814  1 !
;  758          0815  1 ! SIDE EFFECTS:
;  759          0816  1 !
;  760          0817  1 !       If the ISAM file is being BI Journalled, then the formatted portion of
;  761          0818  1 !       the new bucket will have been moved into the BI Journalling buffer
;  762          0819  1 !       associated with the BDB controlling the bucket.
;  763          0820  1 !
;  764          0821  1 !--
;  765          0822  1
;  766          0823  2     BEGIN
;  767          0824  2
;  768          0825  2     EXTERNAL REGISTER
;  769          0826  2         COMMON_RAB_STR,
;  770          0827  2         R_IDX_DFN_STR;
;  771          0828  2
;  772          0829  2     GLOBAL REGISTER
;  773          0830  2         COMMON_IO_STR,
;  774          0831  2         VBN = 6;
;  775          0832  2
;  776          0833  2     RETURN_ON_ERROR ( GET_VBN( .AREA_NO ) );
;  777          0834  2
;  778        P 0835  2     RETURN_ON_ERROR (RM$CACHE(.VBN, .NO_BYTES,
;  779        P 0836  2             CSH$M_NOREAD
;  780        P 0837  2             OR
;  781          0838  2             CSH$M_LOCK));
```

```
;  782     0839  2
;  783     0840  2      IRAB[IRB$L_NXTBDB] = .BDB;
;  784     0841  2      BDB[BDB$V_VAL] = 1;                             ! mark the new bucket valid
;  785     0842  2
;  786     0843  2      ! Set up the bucket overhead fields
;  787     0844  2      !
;  788     0845  2      FORMAT_BKT( .AREA_NO, 1, .NO_BYTES );
;  789     0846  2
;  790     0847  2      ! If this ISAM file is being BI Journalled, then after formatting the bucket
;  791     0848  2      ! move the formatted portion into the BI Journal record buffer controlled by
;  792     0849  2      ! the BI BDB associated with the BDB for the new bucket.
;  793     0850  2      !
;  794     0851  2      IF .IFAB[IFB$V_BI]
;  795     0852  2      THEN
;  796     0853  3          BEGIN
;  797     0854  3
;  798     0855  3          LOCAL
;  799     0856  3              JNL_BUCKET : REF BBLOCK;
;  800     0857  3
;  801     0858  3          JNL_BUCKET = .BBLOCK[.BDB[BDB$L_BI_BDB], BDB$L_ADDR] + RJR$C_BKTLEN;
;  802     0859  3          CH$MOVE (.BKT_ADDR[BKT$W_FREESPACE], .BKT_ADDR, .JNL_BUCKET);
;  803     0860  2          END;
;  804     0861  2
;  805     0862  2      RETURN 1;
;  806     0863  2
;  807     0864  1      END;


                         007C   8F   BB 00000 RM$AL_FRMT_BKT::
                                             PUSHR   #^M<R2,R3,R4,R5,R6>                          ; 0783
                           18    AE   DD 00004                                                    ; 0833
                                      PUSHL   AREA_NO
                              FF11   30 00007                                                     ;
                                      BSBW    GET_VBN
                        5E             04   C0 0000A                                             ;
                                      ADDL2   #4, SP
                        41             50   E9 0000D                                             ;
                                      BLBC    STATUS, 2$
                        53             05   D0 00010                                             ; 0838
                                      MOVL    #5, R3
                        52           1C    AE   D0 00013                                         ;
                                      MOVL    NO_BYTES, R2
                        51             56   D0 00017                                             ;
                                      MOVL    VBN, R1
                                    0000G  30 0001A                                               ;
                                      BSBW    RM$CACHE
                        31             50   E9 0001D                                             ;
                                      BLBC    STATUS, 2$
                     3C   A9           54   D0 00020                                             ; 0840
                                      MOVL    BDB, 60(IRAB)
                     0A   A4           01   88 00024                                             ; 0841
                                      BISB2   #1, 10(BDB)
                           1C    AE   DD 00028                                                    ; 0845
                                      PUSHL   NO_BYTES
                           01    DD 0002B                                                        ;
                                      PUSHL   #1
                           20    AE   DD 0002D                                                    ;
                                      PUSHL   AREA_NO
                              FE39   30 00030                                                     ;
                                      BSBW    FORMAT_BKT
                        5E             0C   C0 00033                                             ;
                                      ADDL2   #12, SP
12        00A0      CA               02   E1 00036                                             ; 0851
                                      BBC     #2, 160(IFAB), 1$
                        50       30    A4   D0 0003C                                             ; 0858
                                      MOVL    48(BDB), R0
50          18      A0 00000044    8F   C1 00040                                                 ;
                                      ADDL3   #68, 24(R0), JNL_BUCKET
60                  65       04    A5   28 00049                                                 ; 0859
                                      MOVC3   4(BKT_ADDR), (BKT_ADDR), (JNL_BUCKET)
                        50             01   D0 0004E 1$:                                        ; 0862
                                      MOVL    #1, R0
                         007C   8F   BA 00051 2$:                                                ; 0864
                                      POPR    #^M<R2,R3,R4,R5,R6>
                                      05 00055                                                    ;
                                      RSB
```

; Routine Size: 86 bytes,   Routine Base: RM$RMS3 + 0248

; 808   0865 1

```
;  810     0866  1 %SBTTL  'RM$ALLOC_BKT'
;  811     0867  1 GLOBAL ROUTINE RM$ALLOC_BKT : RL$RABREG_7 =
;  812     0868  1
;  813     0869  1 !++
;  814     0870  1 !
;  815     0871  1 ! FUNCTIONAL DESCRIPTION:
;  816     0872  1 !       This routine allocates and formats a bucket for use by index sequential
;  817     0873  1 !       files.
;  818     0874  1 !
;  819     0875  1 ! CALLING SEQUENCE:
;  820     0876  1 !       RM$ALLOC_BKT()
;  821     0877  1 !
;  822     0878  1 ! INPUT PARAMETERS:
;  823     0879  1 !       None
;  824     0880  1 !
;  825     0881  1 ! IMPLICIT INPUTS:
;  826     0882  1 !       IDX_DFN         - address of index descriptor for current key of reference
;  827     0883  1 !       IRAB            - address of internal RAB
;  828     0884  1 !       CURBDB          - address of BDB describing bucket which precedes bucket
;  829     0885  1 !                         about to be allocated and formatted
;  830     0886  1 !       IFAB            - address of IFAB needed for I/O and prologue version
;  831     0887  1 !       IMPURE          - address of impure region ( needed for I/O)
;  832     0888  1 !       RAB             - address of user's RAB
;  833     0889  1 !
;  834     0890  1 ! OUTPUT PARAMETERS:
;  835     0891  1 !       None
;  836     0892  1 !
;  837     0893  1 ! IMPLICIT OUTPUTS:
;  838     0894  1 !       IRAB
;  839     0895  1 !               NXTBDB  - address of BDB describing newly allocated and formatted
;  840     0896  1 !                         bucket
;  841     0897  1 !
;  842     0898  1 ! ROUTINE VALUE:
;  843     0899  1 !       Various I/O and extend errors
;  844     0900  1 !
;  845     0901  1 ! SIDE EFFECTS:
;  846     0902  1 !       The new bucket is allocated from the area described by the index descriptor
;  847     0903  1 !       and the level of the bucket preceding it.  The new bucket has the same
;  848     0904  1 !       length as the previous one.  It's forward link becomes that of the previous
;  849     0905  1 !       one whereas the previous one's forward link is the newly allocated bucket.
;  850     0906  1 !       If the previous bucket was the last bucket on that level then the new
;  851     0907  1 !       bucket becomes the last.
;  852     0908  1 !
;  853     0909  1 !--
;  854     0910  1
;  855     0911  2     BEGIN
;  856     0912  2
;  857     0913  2     EXTERNAL REGISTER
;  858     0914  2         COMMON_RAB_STR,
;  859     0915  2         R_IDX_DFN_STR;
;  860     0916  2
;  861     0917  2     LOCAL
;  862     0918  2         BUCKET  : REF BBLOCK,
;  863     0919  2         PREV_BKT        : REF BBLOCK;
;  864     0920  2
;  865     0921  2     PREV_BKT = .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$L_ADDR];
;  866     0922  2
```

```
;  867          0923  2      ! The area number is either the data area number, the lower index area
;  868          0924  2      ! number or the index area number.  This is done in a tricky way taking
;  869          0925  2      ! advantage of the way the area numbers are allocated in memory.
;  870          0926  2      !
;  871          0927  2
;  872          0928  2      BEGIN
;  873          0929  2
;  874          0930  2      LOCAL
;  875          0931  2          AREA_NO;
;  876          0932  2
;  877          0933  2      AREA_NO = .PREV_BKT[BKT$B_LEVEL];
;  878          0934  2
;  879          0935  2      IF .AREA_NO GTRU 2
;  880          0936  2      THEN
;  881          0937  2          AREA_NO = 2;
;  882          0938  2
;  883          0939  2      AREA_NO = .(IDX_DFN[IDX$B_DANUM] - .AREA_NO)<0, 8>;
;  884          0940  2
;  885          0941  2      ! First try to reclaim a bucket from the AVAIL list, if that fails
;  886          0942  2      ! try the extent logic.
;  887          0943  2      !
;  888          0944  2      IF NOT RECLAIM_BKT(.AREA_NO,
;  889          0945  2              .BBLOCK[.IRAB[IRB$L_CURBDB],
;  890          0946  2                  BDB$W_NUMB])
;  891          0947  2      THEN
;  892        P 0948  2          RETURN_ON_ERROR (RM$AL_FRMT_BKT(.AREA_NO,
;  893        P 0949  2                  .BBLOCK[.IRAB[IRB$L_CURBDB],
;  894          0950  2                      BDB$W_NUMB]));
;  895          0951  2
;  896          0952  2      END;                                        ! end of LOCAL definition
;  897          0953  2
;  898          0954  2      BUCKET = .BBLOCK[.IRAB[IRB$L_NXTBDB], BDB$L_ADDR];
;  899          0955  2      BUCKET[BKT$B_LEVEL] = .PREV_BKT[BKT$B_LEVEL];
;  900          0956  2      BUCKET[BKT$L_NXTBKT] = .PREV_BKT[BKT$L_NXTBKT];
;  901          0957  2      PREV_BKT[BKT$L_NXTBKT] = .BBLOCK[.IRAB[IRB$L_NXTBDB], BDB$L_VBN];
;  902          0958  2
;  903          0959  2      IF .PREV_BKT[BKT$V_LASTBKT]
;  904          0960  2      THEN
;  905          0961  3          BEGIN
;  906          0962  3          BUCKET[BKT$V_LASTBKT] = 1;
;  907          0963  3          PREV_BKT[BKT$V_LASTBKT] = 0;
;  908          0964  2          END;
;  909          0965  2
;  910          0966  2      RETURN 1;
;  911          0967  2
;  912          0968  1      END;                                        ! end of routine


                                0C  BB 00000 RM$ALLOC_BKT::
                                                 PUSHR   #^M<R2,R3>                          ; 0867
                      51    20    A9  D0 00002         MOVL    32(IRAB), R1                ; 0921
                      52    18    A1  D0 00006         MOVL    24(R1), PREV_BKT            ;
                      53    0C    A2  9A 0000A         MOVZBL  12(PREV_BKT), AREA_NO       ; 0933
                      02          53  D1 0000E         CMPL    AREA_NO, #2                 ; 0935
```

```
                              03  1B 00011         BLEQU    1$                                    ;
                  53          02  D0 00013         MOVL     #2, AREA_NO                           ; 0937
        50        57          53  C3 00016 1$:     SUBL3    AREA_NO, IDX_DFN, R0                  ; 0939
                  53      14  A0  9A 0001A         MOVZBL   20(R0), AREA_NO                       ;
                  7E      14  A1  3C 0001E         MOVZWL   20(R1), -(SP)                         ; 0945
                              53  DD 00022         PUSHL    AREA_NO                               ; 0944
                            FE3C  30 00024         BSBW     RECLAIM_BKT                           ;
                  5E          08  C0 00027         ADDL2    #8, SP                                ;
                  13          50  E8 0002A         BLBS     R0, 2$                                ;
                  50      20  A9  D0 0002D         MOVL     32(IRAB), R0                          ; 0950
                  7E      14  A0  3C 00031         MOVZWL   20(R0), -(SP)                         ;
                              53  DD 00035         PUSHL    AREA_NO                               ;
                            FF70  30 00037         BSBW     RM$AL_FRMT_BKT                        ;
                  5E          08  C0 0003A         ADDL2    #8, SP                                ;
                  26          50  E9 0003D         BLBC     STATUS, 4$                            ;
                  51      3C  A9  D0 00040 2$:     MOVL     60(IRAB), R1                          ; 0954
                  50      18  A1  D0 00044         MOVL     24(R1), BUCKET                        ;
              0C  A0      0C  A2  90 00048         MOVB     12(PREV_BKT), 12(BUCKET)              ; 0955
              08  A0      08  A2  D0 0004D         MOVL     8(PREV_BKT), 8(BUCKET)                ; 0956
              08  A2      1C  A1  D0 00052         MOVL     28(R1), 8(PREV_BKT)                   ; 0957
                  08      0D  A2  E9 00057         BLBC     13(PREV_BKT), 3$                      ; 0959
              0D  A0          01  88 0005B         BISB2    #1, 13(BUCKET)                        ; 0962
              0D  A2          01  8A 0005F         BICB2    #1, 13(PREV_BKT)                      ; 0963
                  50          01  D0 00063 3$:     MOVL     #1, R0                                ; 0966
                              0C  BA 00066 4$:     POPR     #^M<R2,R3>                            ; 0968
                                  05 00068         RSB                                            ;

; Routine Size:  105 bytes,    Routine Base:  RM$RMS3 + 029E


;  913          0969  1
```

```
;  915          0970  1 %SBTTL  'RM$LOCK_AREA'
;  916          0971  1 GLOBAL ROUTINE RM$LOCK_AREA (AREA_NO) : RL$LOCK_AREA =
;  917          0972  1
;  918          0973  1 !++
;  919          0974  1 !
;  920          0975  1 ! FUNCTIONAL DESCRIPTION:
;  921          0976  1 !       This routine locks the area prologue block for this area descriptor
;  922          0977  1 !       and makes a few basic checks on its validity.
;  923          0978  1 !
;  924          0979  1 ! CALLING SEQUENCE:
;  925          0980  1 !       RM$LOCK_AREA(AREA_NO)
;  926          0981  1 !
;  927          0982  1 ! INPUT PARAMETERS:
;  928          0983  1 !       AREA_NO - area number to lock
;  929          0984  1 !
;  930          0985  1 ! IMPLICIT INPUTS:
;  931          0986  1 !       None
;  932          0987  1 !
;  933          0988  1 ! OUTPUT PARAMETERS:
;  934          0989  1 !       None
;  935          0990  1 !
;  936          0991  1 ! IMPLICIT OUTPUTS:
;  937          0992  1 !       BDB     - address of lock BDB
;  938          0993  1 !       AREA_DESC       - address within buffer of specified area
;  939          0994  1 !
;  940          0995  1 ! ROUTINE VALUE:
;  941          0996  1 !       1       - success
;  942          0997  1 !       AID     - bad area number
;  943          0998  1 !       PLG     - read error on prologue block
;  944          0999  1 !       various hardware errors
;  945          1000  1 !
;  946          1001  1 ! SIDE EFFECTS:
;  947          1002  1 !       None
;  948          1003  1 !
;  949          1004  1 !--
;  950          1005  1
;  951          1006  2     BEGIN
;  952          1007  2
;  953          1008  2     EXTERNAL REGISTER
;  954          1009  2         COMMON_RAB_STR,
;  955          1010  2         R_BDB_STR,
;  956          1011  2         AREA_DESC = 7   : REF BBLOCK;
;  957          1012  2
;  958          1013  2     LOCAL
;  959          1014  2         AREA_VBN;
;  960          1015  2
;  961          1016  2     IF .AREA_NO<0, 8> GEQU .IFAB[IFB$B_AMAX]    ! check range of input
;  962          1017  2     THEN
;  963          1018  2         RETURN RMSERR(AID);
;  964          1019  2
;  965          1020  2     ! Calculate the VBN in which this area descriptor is located
;  966          1021  2     !
;  967          1022  2     AREA_VBN = (.AREA_NO/8) + .IFAB[IFB$B_AVBN];
;  968          1023  2     AREA_DESC = .AREA_NO AND %X'00000007';
;  969          1024  2
;  970          1025  2     ! Lock VBN containing area descriptor.
;  971          1026  2     !
```

```
;  972          1027  3     BEGIN
;  973          1028  3
;  974          1029  3     GLOBAL REGISTER
;  975          1030  3         R_BKT_ADDR_STR;
;  976          1031  3
;  977          1032  3     RETURN_ON_ERROR (RM$CACHE(.AREA_VBN, 512, CSH$M_LOCK));
;  978          1033  3
;  979          1034  3     ! Set buffer invalid so that changes made to area descriptor need not be
;  980          1035  3     ! backed out.
;  981          1036  3     !
;  982          1037  3     BDB[BDB$V_VAL] = 0;
;  983          1038  3
;  984          1039  3     RETURN_ON_ERROR ( RM$CHKSUM(), RM$RELEASE(0) );
;  985          1040  3
;  986          1041  3     ! calc location in buffer of area desc
;  987          1042  3     !
;  988          1043  3     AREA_DESC = (.AREA_DESC*AREA$K_BLN) + .BKT_ADDR;
;  989          1044  2     END;
;  990          1045  2
;  991          1046  2     IF .AREA_DESC[AREA$B_AREAID] NEQ .AREA_NO<0, 8>
;  992          1047  2     THEN
;  993          1048  3         BEGIN
;  994          1049  3         RM$RELEASE(0);
;  995          1050  3         RETURN RMSERR(PLG);
;  996          1051  2         END;                                    ! end definition of BKT_ADDR
;  997          1052  2
;  998          1053  2     RETURN 1;
;  999          1054  2
; 1000          1055  1     END;


                                7E              55  7D 00000 RM$LOCK_AREA::
                                                            MOVQ    R5, -(SP)                                   ; 0971
                             00B1   CA      0C  AE  91 00003         CMPB    AREA_NO, 177(IFAB)                  ; 1016
                                                07  1F 00009         BLSSU   1$
                                    50  83F4    8F  3C 0000B         MOVZWL  #33780, R0                          ; 1018
                                                56  11 00010         BRB     4$
                          50        0C  AE          08  C7 00012 1$:  DIVL3   #8, AREA_NO, R0                    ; 1022
                                    51  00B0    CA  9A 00017         MOVZBL  176(IFAB), AREA_VBN
                                    51          50  C0 0001C         ADDL2   R0, AREA_VBN
                57        0C  AE              03  00  EF 0001F         EXTZV   #0, #3, AREA_NO, AREA_DESC          ; 1023
                                    53          01  D0 00025         MOVL    #1, R3                              ; 1032
                                    52  0200    8F  3C 00028         MOVZWL  #512, R2
                                              0000G 30 0002D         BSBW    RM$CACHE
                                    35          50  E9 00030         BLBC    STATUS, 4$
                             0A     A4          01  8A 00033         BICB2   #1, 10(BDB)                         ; 1037
                                              0000G 30 00037         BSBW    RM$CHKSUM                           ; 1039
                                    56          50  D0 0003A         MOVL    R0, STATUS
                                    0A          56  E8 0003D         BLBS    STATUS, 2$
                                                53  D4 00040         CLRL    R3
                                              0000G 30 00042         BSBW    RM$RELEASE
                                    50          56  D0 00045         MOVL    STATUS, R0
                                                1E  11 00048         BRB     4$
                          50                57          06  78 0004A 2$:  ASHL    #6, AREA_DESC, R0             ; 1043
```

```
                 57      50      55  C1 0004E        ADDL3   BKT_ADDR, R0, AREA_DESC                 ;
                    0C   AE   02   A7  91 00052        CMPB    2(AREA_DESC), AREA_NO                   ; 1046
                                   0C  13 00057        BEQL    3$                                      ;
                                   53  D4 00059        CLRL    R3                                      ; 1049
                              0000G 30  0005B        BSBW    RM$RELEASE                              ;
                      50    861C  8F  3C 0005E        MOVZWL  #34332, R0                              ; 1050
                                   03  11 00063        BRB     4$                                      ;
                      50           01  D0 00065 3$:    MOVL    #1, R0                                  ; 1053
                      55           8E  7D 00068 4$:    MOVQ    (SP)+, R5                               ; 1055
                                       05 0006B        RSB                                             ;

; Routine Size:  108 bytes,    Routine Base:  RM$RMS3 + 0307

; 1001          1056  1
; 1002          1057  1 END
; 1003          1058  1
; 1004          1059  0 ELUDOM



;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  RM$RMS3                       883  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        78         2         154        00:00.4



;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3ALLBKT/OBJ=OBJ$:RM3ALLBKT MSRC$:RM3ALLBKT/UPDATE=(ENH$:RM3ALLBKT)

; Size:         883 code + 0 data bytes
; Run Time:         00:22.0
; Elapsed Time:     00:45.2
; Lines/CPU Min:     2884
; Lexemes/CPU-Min: 18642
; Memory Used:  124 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