```
RRRRRRRRRRR      MMM          MMM      SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM      SSSSSSSSSSSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMMMMMM    MMMMMMM   SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   MMM   MMM   MMM SSS
RRR          RRR MMM   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               SSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
RRR          RRR MMM          MMM  SSSSSSSSSSS
```

```
RRRRRRRR    MM       MM    333333    UU       UU  PPPPPPPP    DDDDDDDD    DDDDDDDD    EEEEEEEEEE  LL
RRRRRRRR    MM       MM    333333    UU       UU  PPPPPPPP    DDDDDDDD    DDDDDDDD    EEEEEEEEEE  LL
RR      RR  MMMM   MMMM  33      33  UU       UU  PP      PP  DD      DD  DD      DD  EE          LL
RR      RR  MMMM   MMMM  33      33  UU       UU  PP      PP  DD      DD  DD      DD  EE          LL
RR      RR  MM MM MM MM        33  UU       UU  PP      PP  DD      DD  DD      DD  EE          LL
RR      RR  MM  MMM  MM        33  UU       UU  PP      PP  DD      DD  DD      DD  EE          LL
RRRRRRRR    MM       MM      33    UU       UU  PPPPPPPP    DD      DD  DD      DD  EEEEEEEE    LL
RRRRRRRR    MM       MM      33    UU       UU  PPPPPPPP    DD      DD  DD      DD  EEEEEEEE    LL
RR  RR      MM       MM        33  UU       UU  PP          DD      DD  DD      DD  EE          LL
RR  RR      MM       MM        33  UU       UU  PP          DD      DD  DD      DD  EE          LL
RR    RR    MM       MM  33    33  UU       UU  PP          DD      DD  DD      DD  EE          LL                ....
RR    RR    MM       MM  33    33  UU       UU  PP          DD      DD  DD      DD  EE          LL                ....
RR      RR  MM       MM    333333    UUUUUUUUUU  PP          DDDDDDDD    DDDDDDDD    EEEEEEEEEE  LLLLLLLLLL        ....
RR      RR  MM       MM    333333    UUUUUUUUUU  PP          DDDDDDDD    DDDDDDDD    EEEEEEEEEE  LLLLLLLLLL        ....

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 MODULE RM3UPDDEL (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 !**************************************************************************
;    28          0028  1
;    29          0029  1 !++
;    30          0030  1 !
;    31          0031  1 ! FACILITY:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !
;    35          0035  1 !       This module contains routines common to $UPDATE and $DELETE
;    36          0036  1 !
;    37          0037  1 ! ENVIRONMENT:
;    38          0038  1 !
;    39          0039  1 !               VAX/VMS OPERATING SYSTEM
;    40          0040  1 !
;    41          0041  1 !--
;    42          0042  1
;    43          0043  1 !
;    44          0044  1 ! AUTHOR:        Todd M. Katz          RE-CREATION DATE:       17-Jul-1982
;    45          0045  1 !
;    46          0046  1 !
;    47          0047  1 ! MODIFIED BY:
;    48          0048  1 !
;    49          0049  1 !       V03-009 JWT0183         Jim Teague              21-May-1984
;    50          0050  1 !               Fix cause of infrequent "invalid RRV" error.  If a
;    51          0051  1 !               record not in its original bucket is moved out of
;    52          0052  1 !               its current bucket as a result of a split, then any
;    53          0053  1 !               one who wants that record will get an "invalid RRV"
;    54          0054  1 !               error any time 1) after the split, and 2) before the
;    55          0055  1 !               RRV gets updated to point to the new location of
;    56          0056  1 !               the record.  The solution for now is just to retry
;    57          0057  1 !               the RRV a number of times.  Note that if anything
```

```
;    58          0058  1 !               keeps the process from updating the RRV (waiting on
;    59          0059  1 !               the CPU, for example), large numbers of retries
;    60          0060  1 !               may be necessary for those who are after the record.
;    61          0061  1 !
;    62          0062  1 !       V03-008 MCN0002         Maria del C. Nasr       15-Mar-1983
;    63          0063  1 !               More linkages reorganization
;    64          0064  1 !
;    65          0065  1 !       V03-007 MCN0001         Maria del C. Nasr       01-Mar-1983
;    66          0066  1 !               Reorganize linkages
;    67          0067  1 !
;    68          0068  1 !       V03-006 TMK0004         Todd M. Katz            03-Feb-1983
;    69          0069  1 !               Add support for Recovery Unit Journalling and RU ROLLBACK
;    70          0070  1 !               Recovery of ISAM files. If RMS is within a Recovery Unit and
;    71          0071  1 !               pseudo record locking is enabled, then make sure all required
;    72          0072  1 !               record locking is done.
;    73          0073  1 !
;    74          0074  1 !       V03-005 RAS0116         Ron Schaefer            18-Jan-1983
;    75          0075  1 !               Enhance TMK0003 to match the corresponding .BUG correction;
;    76          0076  1 !               by returning the status of RMS$_RRV rather than RMS$_BUG
;    77          0077  1 !               if the repositioning fails.  Also simplify the IF-THEN-ELSE
;    78          0078  1 !               logic during repositioning.
;    79          0079  1 !
;    80          0080  1 !       V03-004 TMK0003         Todd M. Katz            27-Dec-1982
;    81          0081  1 !               I have added another method of positioning to the primary data
;    82          0082  1 !               record to be deleted/updated. This method is used when the
;    83          0083  1 !               attempt to position to the primary data record by means of its
;    84          0084  1 !               RFA fails because the RFA is incorrect, or has been deleted, and
;    85          0085  1 !               RMS positioned orginally to the current primary data record by
;    86          0086  1 !               means of the primary index. In this case RMS knows that the
;    87          0087  1 !               record must be somewhere in the file, because it is the current
;    88          0088  1 !               record for this stream, and that the primary key of the record
;    89          0089  1 !               resides in keybuffer 2. This primary key together with the RFA
;    90          0090  1 !               is used to position to the current primary data record by
;    91          0091  1 !               searching the primary index by exact key value and comparing the
;    92          0092  1 !               RFA of the current primary data record with the RFA saved in
;    93          0093  1 !               each of the primary data records with this key value until a
;    94          0094  1 !               match is found.
;    95          0095  1 !
;    96          0096  1 !       V03-003 KBT0236         Keith B. Thompson       23-Aug-1982
;    97          0097  1 !               Reorganize psect
;    98          0098  1 !
;    99          0099  1 !       V03-002 TMK0002         Todd M. Katz            17-Jul-1982
;   100          0100  1 !               REFORMAT the routines within this module.
;   101          0101  1 !
;   102          0102  1 !*****
;   103          0103  1
;   104          0104  1 LIBRARY 'RMSLIB:RMS';
;   105          0105  1
;   106          0106  1 REQUIRE 'RMSSRC:RMSIDXDEF';
;   107          0171  1
;   108          0172  1 ! Define default PSECTS for code
;   109          0173  1 !
;   110          0174  1 PSECT
;   111          0175  1     CODE = RMS$RMS3(PSECT_ATTR),
;   112          0176  1     PLIT = RMS$RMS3(PSECT_ATTR);
;   113          0177  1
;   114          0178  1 ! Linkages
```

```
;  115          0179  1 !
;  116          0180  1 LINKAGE
;  117          0181  1     L_ERROR_LINK1,
;  118          0182  1     L_PRESERVE1,
;  119          0183  1     L_QUERY_AND_LOCK,
;  120          0184  1     L_RABREG_67,
;  121          0185  1     L_RABREG_7;
;  122          0186  1
;  123          0187  1 ! External Routines
;  124          0188  1 !
;  125          0189  1 EXTERNAL ROUTINE
;  126          0190  1     RM$CLEAN_BDB          : RL$ERROR_LINK1 NOVALUE,
;  127          0191  1     RM$CSEARCH_TREE       : RL$RABREG_67,
;  128          0192  1     RM$FIND_BY_RRV        : RL$RABREG_67,
;  129          0193  1     RM$GETNEXT_REC        : RL$RABREG_67,
;  130          0194  1     RM$KEY_DESC           : RL$RABREG_67,
;  131          0195  1     RM$QUERY_HARD         : RL$QUERY_AND_LOCK ADDRESSING_MODE (GENERAL),
;  132          0196  1     RM$RECORD_ID          : RL$RABREG_67,
;  133          0197  1     RM$RECORD_VBN         : RL$PRESERVE1,
;  134          0198  1     RM$UNLOCK_HARD        : RL$QUERY_AND_LOCK ADDRESSING_MODE (GENERAL);
```

```
;  136          0199  1 %SBTTL 'RM$UPDDELCOM'
;  137          0200  1 GLOBAL ROUTINE RM$UPDDELCOM : RL$RABREG_67 =
;  138          0201  1
;  139          0202  1 !++
;  140          0203  1 !
;  141          0204  1 ! FUNCTIONAL DESCRIPTION:
;  142          0205  1 !
;  143          0206  1 !       This routine performs common checks for $UPDATE and $DELETE.
;  144          0207  1 !
;  145          0208  1 !       1. As there must be a current primary data record to $DELETE or $UPDATE
;  146          0209  1 !          a check is made for one.
;  147          0210  1 !
;  148          0211  1 !       2. If this is a $UPDATE, the temporary IRAB locations IRB$L_PUTUP_VBN
;  149          0212  1 !          and IRB$W_PUTUP_ID are set up with the RFA address of the current
;  150          0213  1 !          primary data record. This is so code common to $UPDATE and $PUT maybe
;  151          0214  1 !          utilized.
;  152          0215  1 !
;  153          0216  1 !       3. The current primary data record is positioned to by means of its RFA
;  154          0217  1 !          address. During this positioning the primary data bucket is accessed,
;  155          0218  1 !          and if the opration is a $DELETE the RRV bucket (if there is one) is
;  156          0219  1 !          accessed too. If for some reason the primary data record can not be
;  157          0220  1 !          positioned to by its RFA, then it is positioned to by its key value
;  158          0221  1 !          which will be found in keybuffer 2. This alternate positioning can
;  159          0222  1 !          only be done if RMS originally positioned to the primary data record
;  160          0223  1 !          by means of the primary index.
;  161          0224  1 !
;  162          0225  1 !       4. Unlock the current primary data record or just query its lock status
;  163          0226  1 !          based upon whether record locking is necessary, automatic record
;  164          0227  1 !          locking is or isn't requested, and whether the operation is a $DELETE
;  165          0228  1 !          or an $UPDATE.
;  166          0229  1 !
;  167          0230  1 ! CALLING SEQUENCE:
;  168          0231  1 !
;  169          0232  1 !       RM$UPDDELCOM()
;  170          0233  1 !
;  171          0234  1 ! INPUT PARAMETERS:
;  172          0235  1 !       NONE
;  173          0236  1 !
;  174          0237  1 ! IMPLICIT INPUTS:
;  175          0238  1 !
;  176          0239  1 !       IFAB                    - address if IFAB
;  177          0240  1 !           IFB$W_KBUFSZ        - size of a keybuffer
;  178          0241  1 !           IFB$V_NORECLK       - if set, no record locking is necessary
;  179          0242  1 !           IFB$V_RUP           - if set, Recovery Unit is in progress
;  180          0243  1 !           IRB$V_RU_RLK        - if set, pseudo record locking is enabled
;  181          0244  1 !
;  182          0245  1 !       IRAB                    - address of IRAB
;  183          0246  1 !           IRB$L_KEYBUF        - address of the contigious keybuffers
;  184          0247  1 !           IRB$B_RP_KREF       - key of reference of last positioning
;  185          0248  1 !           IRB$W_UDR_ID        - ID of current record(RRV)
;  186          0249  1 !           IRB$L_UDR_VBN       - VBN of current record(RRV)
;  187          0250  1 !           IRB$V_UNLOCK_RP     - if set, there is a current record to unlock
;  188          0251  1 !           IRB$V_UPDATE        - if set, current operation is an $UPDATE
;  189          0252  1 !
;  190          0253  1 ! OUTPUT PARAMETERS:
;  191          0254  1 !       NONE
;  192          0255  1 !
```

```
;  193     0256  1 ! IMPLICIT OUTPUTS:
;  194     0257  1 !
;  195     0258  1 !       IRAB                    - address of the IRAB
;  196     0259  1 !           IRB$B_CACHEFLGS     - the bit CSH$V_LOCK is set
;  197     0260  1 !           IRB$L_CURBDB        - address of BDB for primary data bucket
;  198     0261  1 !           IRB$L_NXTBDB        - address of BDB for RRV data bucket (or 0)
;  199     0262  1 !           IRB$B_KEYSZ         - size of primary key (if alternate positioning)
;  200     0263  1 !           IRB$L_PUTUP_VBN     - RFA VBN of record to be updated
;  201     0264  1 !           IRB$W_PUTUP_ID      - RFA ID of record to be updated
;  202     0265  1 !           IRB$W_SRCHFLAGS     - 0 (if alternate positioning is used)
;  203     0266  1 !           IRB$V_UNLOCK_RP     - 0
;  204     0267  1 !
;  205     0268  1 !       REC_ADDR                - address of current primary data record
;  206     0269  1 !
;  207     0270  1 ! ROUTINE VALUE:
;  208     0271  1 !
;  209     0272  1 !       BUG             - unable to position to primary data record
;  210     0273  1 !       CUR             - illegal or no current primary data record
;  211     0274  1 !       RNL             - current primary data record is not locked
;  212     0275  1 !       SUC             - success
;  213     0276  1 !
;  214     0277  1 ! SIDE EFFECTS:
;  215     0278  1 !
;  216     0279  1 !       If RMS must perform a tree search to position to the current primary
;  217     0280  1 !           data record then AP will be trashed.
;  218     0281  1 !       If record locking is unnecessary then the lock status of the current
;  219     0282  1 !           primary data record is not checked.
;  220     0283  1 !       If automatic locking is specified, then the current primary data record
;  221     0284  1 !           is unlocked only if the operation is a $DELETE. In the case of an
;  222     0285  1 !           $UPDATE, just the existance of a lock on the current primary data
;  223     0286  1 !           record is verified.
;  224     0287  1 !       If automatic locking is not specified then the current primary data
;  225     0288  1 !           record is never unlocked, and the existance of a lock on the
;  226     0289  1 !           current primary data record is verified.
;  227     0290  1 !       If all goes successfully, RMS accesses the primary data bucket and
;  228     0291  1 !           positions to the current primary data record. If there is an RRV for
;  229     0292  1 !           the current primary data record and the operation is a $DELETE the
;  230     0293  1 !           bucket containing it is accessed too.
;  231     0294  1 !       Otherwise, all accessed buckets are released.
;  232     0295  1 !
;  233     0296  1 !--
;  234     0297  1
;  235     0298  2     BEGIN
;  236     0299  2
;  237     0300  2     EXTERNAL REGISTER
;  238     0301  2         COMMON_RAB_STR,
;  239     0302  2         R_REC_ADDR,
;  240     0303  2         R_IDX_DFN_STR;
;  241     0304  2
;  242     0305  2     LOCAL
;  243     0306  2         FLAGS,
;  244     0307  2         STATUS;
;  245     0308  2
;  246     0309  2     ! Initialize the IRB$L_NXTBDB and IRB$L_LOCK_BDB fields to 0.
;  247     0310  2     !
;  248     0311  2     IRAB[IRB$L_NXTBDB] = 0;
;  249     0312  2     IRAB[IRB$L_LOCK_BDB] = 0;
```

```
;  250          0313  2
;  251          0314  2      ! If this stream does not have a current primary data record then
;  252          0315  2      ! immediately return an error.
;  253          0316  2      !
;  254          0317  2      IF  .IRAB[IRB$L_UDR_VBN] EQLU 0
;  255          0318  2            OR
;  256          0319  2          .IRAB[IRB$W_UDR_ID] EQLU 0
;  257          0320  2      THEN
;  258          0321  2          RETURN (RMSERR(CUR));
;  259          0322  2
;  260          0323  2      ! If this operation is an $UPDATE then RMS saves the RFA address of the
;  261          0324  2      ! current primary data record in a temporary IRAB location (so that code
;  262          0325  2      ! common to both $PUT and $UPDATE maybe used), and sets up to signal the
;  263          0326  2      ! routine RM$FIND_BY_RRV that there is no need to keep a lock on the RRV
;  264          0327  2      ! bucket if the current primary data record is not in its original bucket.
;  265          0328  2      !
;  266          0329  2      IF .IRAB[IRB$V_UPDATE]
;  267          0330  2      THEN
;  268          0331  3          BEGIN
;  269          0332  3          IRAB[IRB$L_PUTUP_VBN] = .IRAB[IRB$L_UDR_VBN];
;  270          0333  3          IRAB[IRB$W_PUTUP_ID]  = .IRAB[IRB$W_UDR_ID];
;  271          0334  3          FLAGS = 0;
;  272          0335  3          END
;  273          0336  3
;  274          0337  3      ! If this operation is a $DELETE then RMS sets up to signal RM$FIND_BY_RRV
;  275          0338  3      ! that if the current primary data record is not in its original bucket
;  276          0339  3      ! it should keep a lock on the original bucket anyway since that bucket
;  277          0340  3      ! contains the RRV for the current primary data record which must also be
;  278          0341  3      ! deleted as part of the $DELETE.
;  279          0342  3      !
;  280          0343  2      ELSE
;  281          0344  2          FLAGS = 1;
;  282          0345  2
;  283          0346  2      ! Position to the current primary data record locking the primary data
;  284          0347  2      ! bucket and the RRV bucket (if it is required).
;  285          0348  2      !
;  286          0349  2      IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  287          0350  2
;  288          0351  2      STATUS = RM$FIND_BY_RRV (.IRAB[IRB$L_UDR_VBN],
;  289          0352  2                               .IRAB[IRB$W_UDR_ID],
;  290          0353  2                               .FLAGS);
;  291          0354  2 !
;  292          0355  2 ! Here's the scoop -- if someone has split the bucket containing
;  293          0356  2 !       the desired record but hasn't updated the RRV yet, then
;  294          0357  2 !       we're going to get "invalid RRV" errors.  If we just got
;  295          0358  2 !       one of those errors, then release our bucket locks and
;  296          0359  2 !       retry a finite number of times until we can hopefully get
;  297          0360  2 !       the new and improved RRV.
;  298          0361  2 !
;  299          0362  3          IF .STATUS<0,16> EQLU RMSERR(RRV)
;  300          0363  2          THEN
;  301          0364  2              INCRU I FROM 0 TO 25 DO
;  302          0365  3                  BEGIN
;  303          0366  3                  IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  304          0367  4                  IF (STATUS = RM$FIND_BY_RRV (.IRAB[IRB$L_UDR_VBN],
;  305          0368  4                                               .IRAB[IRB$W_UDR_ID],
;  306          0369  4                                               .FLAGS))
```

```
;  307     0370  3                THEN
;  308     0371  3                    EXITLOOP;
;  309     0372  2                END;
;  310     0373  2
;  311     0374  2      ! If RMS encounters some obscure I/O error during its positioning by
;  312     0375  2      ! means of the current record's RFA, return the obscure error.
;  313     0376  2      !
;  314     0377  2      IF NOT .STATUS
;  315     0378  2      THEN
;  316     0379  3          BEGIN
;  317     0380  3
;  318     0381  4          IF .STATUS<0,16> NEQU RMSERR(DEL)
;  319     0382  3          THEN
;  320     0383  3              RETURN .STATUS;
;  321     0384  3
;  322     0385  3          ! If RMS is unable to position to the current primary data record
;  323     0386  3          ! because its RFA has been deleted and is unable to use the alternate
;  324     0387  3          ! positioning method because the primary index was not used during the
;  325     0388  3          ! positioning to the current primary data record, then also return an
;  326     0389  3          ! error.
;  327     0390  3          !
;  328     0391  3          IF .IRAB[IRB$B_RP_KREF] NEQU 0
;  329     0392  3          THEN
;  330     0393  3              RETURN RMSERR(RRV);
;  331     0394  3
;  332     0395  3          ! If RMS is unable to position to the current primary data record
;  333     0396  3          ! because either the RFA has been deleted, or does not currently
;  334     0397  3          ! point to the current primary data record, then make use of the
;  335     0398  3          ! current record's primary key in keybuffer 2, and the current
;  336     0399  3          ! record's RFA to position to it.
;  337     0400  3          !
;  338     0401  3
;  339     0402  3
;  340     0403  3          ! Retrieve the primary key index descriptor and set up to
;  341     0404  3          ! position to the first record with this primary key value.
;  342     0405  3          !
;  343     0406  3          RM$KEY_DESC(0);
;  344     0407  3
;  345     0408  3          IRAB[IRB$B_KEYSZ] = .IDX_DFN[IDX$B_KEYSZ];
;  346     0409  3          IRAB[IRB$W_SRCHFLAGS] = 0;
;  347     0410  3
;  348     0411  3          ! Continue until either the current primary data record has
;  349     0412  3          ! been positioned to, or all records with this primary key
;  350     0413  3          ! have been exhausted.
;  351     0414  3          !
;  352     0415  3          WHILE 1
;  353     0416  3          DO
;  354     0417  4              BEGIN
;  355     0418  4
;  356     0419  4              GLOBAL REGISTER
;  357     0420  4                  R_BDB;
;  358     0421  4
;  359     0422  4              BUILTIN
;  360     0423  4                  AP;
;  361     0424  4
;  362     0425  4              ! If RMS is unable to position to the current primary data
;  363     0426  4              ! record by means of this alternate method, return a "bug"
```

```
;  364          0427  4              ! error.
;  365          0428  4              !
;  366          0429  4              IF NOT RM$CSEARCH_TREE()
;  367          0430  4              THEN
;  368          0431  4                  RETURN RMSERR(RRV);
;  369          0432  4
;  370          0433  4              ! If the record current positioned to is in fact the current
;  371          0434  4              ! primary data record then leave the positioning loop;
;  372          0435  4              ! otherwise, position immediately to the record that follows
;  373          0436  4              ! and continue.
;  374          0437  4              !
;  375          0438  4              AP = 3;
;  376          0439  4
;  377          0440  4              IF   .IRAB[IRB$W_UDR_ID] EQLU RM$RECORD_ID()
;  378          0441  4                   AND
;  379          0442  4                  .IRAB[IRB$L_UDR_VBN] EQLU RM$RECORD_VBN()
;  380          0443  4              THEN
;  381          0444  4                  EXITLOOP
;  382          0445  4              ELSE
;  383          0446  4                  RM$GETNEXT_REC();
;  384          0447  4
;  385          0448  3              END;
;  386          0449  2          END;
;  387          0450  2
;  388          0451  2      ! Unlocking of the current primary data record.
;  389          0452  2      !
;  390          0453  2      ! It will never be necessary to unlock the current primary data record
;  391          0454  2      ! when the file is opened for exclusive access unless the process is within
;  392          0455  2      ! a Recovery Unit and pseudo record locking is indicated.
;  393          0456  2      !
;  394          0457  2      ! Always call the _HARD lock entry points to map REA held locks to a RNL
;  395          0458  2      ! status.  REA locked records are not protected from concurrent update or
;  396          0459  2      ! delete except by this mechanism.
;  397          0460  2      !
;  398          0461  2      IF  NOT .IFAB[IFB$V_NORECLK]
;  399          0462  2           OR
;  400          0463  3          (.IFAB[IFB$V_RUP]
;  401          0464  3                  AND
;  402          0465  3                 .IFAB[IFB$V_RU_RLK])
;  403          0466  2      THEN
;  404          0467  3          BEGIN
;  405          0468  3
;  406          0469  3          ! If automatic record locking is specified and the operation is a
;  407          0470  3          ! $DELETE then the current primary data record must be unlocked. Return
;  408          0471  3          ! an error if the record was not locked in the first place.
;  409          0472  3          !
;  410          0473  3          IF .IRAB[IRB$V_UNLOCK_RP]
;  411          0474  3               AND
;  412          0475  3              NOT .IRAB[IRB$V_UPDATE]
;  413          0476  3          THEN
;  414          0477  4              BEGIN
;  415          0478  4              IRAB[IRB$V_UNLOCK_RP] = 0;
;  416          0479  4
;  417        P 0480  4              RETURN_ON_ERROR (RM$UNLOCK_HARD (.IRAB[IRB$L_UDR_VBN],
;  418        P 0481  4                                               .IRAB[IRB$W_UDR_ID]),
;  419        P 0482  4                  BEGIN
;  420        P 0483  4                  IRAB[IRB$V_UPDATE] = 0;
```

```
;  421        P 0484  4                RM$CLEAN_BDB();
;  422        P 0485  4                STATUS = RMSERR(RNL);
;  423          0486  5                END)
;  424          0487  4            END
;  425          0488  4
;  426          0489  4        ! If either automatic record locking is specified and the operation
;  427          0490  4        ! is an $UPDATE or automatic record locking is not specified, just
;  428          0491  4        ! check that the current primary data record is locked, and return
;  429          0492  4        ! an error if it is not. If this is an $UPDATE and automatic record
;  430          0493  4        ! locking is specified, then the current primary data record will be
;  431          0494  4        ! unlocked after the $UPDATE completes.
;  432          0495  4        !
;  433          0496  3        ELSE
;  434          0497  3            IF RM$QUERY_HARD (.IRAB[IRB$L_UDR_VBN],
;  435          0498  4                              .IRAB[IRB$W_UDR_ID]) NEQU RMSSUC(OK_ALK)
;  436          0499  3            THEN
;  437          0500  4                BEGIN
;  438          0501  4                IRAB[IRB$V_UPDATE] = 0;
;  439          0502  4                RM$CLEAN_BDB();
;  440          0503  4                RETURN RMSERR(RNL);
;  441          0504  3                END;
;  442          0505  3
;  443          0506  2            END;
;  444          0507  2
;  445          0508  2        RETURN RMSSUC();
;  446          0509  2
;  447          0510  1        END;                                    ! end of routine


                                                        .TITLE  RM3UPDDEL
                                                        .IDENT  \V04-000\

                                                        .EXTRN  RM$CLEAN_BDB. RM$CSEARCH_TREE
                                                        .EXTRN  RM$FIND_BY_RRV, RM$GETNEXT_REC
                                                        .EXTRN  RM$KEY_DESC, RM$QUERY_HARD
                                                        .EXTRN  RM$RECORD_ID, RM$RECORD_VBN
                                                        .EXTRN  RM$UNLOCK_HARD

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

                                    1C  BB 00000 RM$UPDDELCOM::
                                                        PUSHR   #^M<R2,R3,R4>                   ; 0200
                                3C      A9  D4 00002            CLRL    60(IRAB)                ; 0311
                              0084      C9  D4 00005            CLRL    132(IRAB)               ; 0312
                        50    00B0      C9  D0 00009            MOVL    176(IRAB), R0           ; 0317
                                          06  13 0000E            BEQL    1$                      ;
                              00BC      C9  B5 00010            TSTW    188(IRAB)               ; 0319
                                          08  12 00014            BNEQ    3$                      ;
                        50    8484      8F  3C 00016 1$:        MOVZWL  #33972, R0              ; 0321
                                    00FE  31  0001B 2$:         BRW     17$                     ;
OF                06      A9            03  E1 0001E 3$:        BBC     #3, 6(IRAB), 4$         ; 0329
                  78      A9            50  D0 00023            MOVL    R0, 120(IRAB)           ; 0332
              0080        C9      00BC  C9  B0 00027            MOVW    188(IRAB), 128(IRAB)    ; 0333
                                        53  D4 0002E            CLRL    FLAGS                   ; 0334
                                        03  11 00030            BRB     5$                      ; 0329
                        53              01  D0 00032 4$:        MOVL    #1, FLAGS               ; 0344
                  40    A9              01  90 00035 5$:        MOVB    #1, 64(IRAB)            ; 0349
```

```
                                        53  DD 00039          PUSHL   FLAGS                                 ; 0353
                      7E     00BC       C9  3C 0003B          MOVZWL  188(IRAB), -(SP)                      ; 0352
                             00B0       C9  DD 00040          PUSHL   176(IRAB)                             ; 0351
                                     0000G  30 00044          BSBW    RM$FIND_BY_RRV                        ;
                      5E                0C  C0 00047          ADDL2   #12, SP                               ;
             8684     8F                50  B1 0004A          CMPW    STATUS, #34436                        ; 0362
                                        21  12 0004F          BNEQ    7$                                    ;
                                        52  D4 00051          CLRL    I                                     ; 0364
               40     A9                01  90 00053  6$:     MOVB    #1, 64(IRAB)                          ; 0366
                                        53  DD 00057          PUSHL   FLAGS                                 ; 0369
                      7E     00BC       C9  3C 00059          MOVZWL  188(IRAB), -(SP)                      ; 0368
                             0090       C9  DD 0005E          PUSHL   176(IRAB)                             ; 0367
                                     0000G  30 00062          BSBW    RM$FIND_BY_RRV                        ;
                      5E                0C  C0 00065          ADDL2   #12, SP                               ;
                      53                50  E8 00068          BLBS    STATUS, 12$                           ;
                                        52  D6 0006B          INCL    I                                     ; 0364
                      19                52  D1 0006D          CMPL    I, #25                                ;
                                        E1  1B 00070          BLEQU   6$                                    ;
                      49                50  E8 00072  7$:     BLBS    STATUS, 12$                           ; 0377
             8262     8F                50  B1 00075          CMPW    STATUS, #33378                        ; 0381
                                        9F  12 0007A          BNEQ    2$                                    ;
                             00C2       C9  95 0007C          TSTB    194(IRAB)                             ; 0391
                                        17  12 00080          BNEQ    9$                                    ;
                                        7E  D4 00082          CLRL    -(SP)                                 ; 0406
                                     0000G  30 00084          BSBW    RM$KEY_DESC                           ;
                      5E                04  C0 00087          ADDL2   #4, SP                                ;
             00A6     C9       20       A7  90 0008A          MOVB    32(IDX_DFN), 166(IRAB)                ; 0408
                               42       A9  B4 00090          CLRW    66(IRAB)                              ; 0409
                                     0000G  30 00093  8$:     BSBW    RM$CSEARCH_TREE                       ; 0429
                      07                50  E8 00096          BLBS    R0, 10$                               ;
                      50     8684       8F  3C 00099  9$:     MOVZWL  #34436, R0                            ; 0431
                                        7C  11 0009E          BRB     17$                                   ;
                      5C                03  D0 000A0  10$:    MOVL    #3, AP                                ; 0438
                                     0000G  30 000A3          BSBW    RM$RECORD_ID                          ; 0440
 50     00BC     C9                10   00  ED 000A6          CMPZV   #0, #16, 188(IRAB), R0                ;
                                        0A  12 000AD          BNEQ    11$                                   ;
                                     0000G  30 000AF          BSBW    RM$RECORD_VBN                         ; 0442
                      50     00B0       C9  D1 000B2          CMPL    176(IRAB), R0                         ;
                                        05  13 000B7          BEQL    12$                                   ;
                                     0000G  30 000B9  11$:    BSBW    RM$GETNEXT_REC                        ; 0446
                                        D5  11 000BC          BRB     8$                                    ; 0415
         0C     06    AA                03  E1 000BE  12$:    BBC     #3, 6(IFAB), 13$                      ; 0461
         50     00A2  CA                02  E1 000C3          BBC     #2, 162(IFAB), 16$                    ; 0463
         4A     00A2  CA                03  E1 000C9          BBC     #3, 162(IFAB), 16$                    ; 0465
         1E     05    A9                05  E1 000CF  13$:    BBC     #5, 5(IRAB), 14$                      ; 0473
         19     06    A9                03  E0 000D4          BBS     #3, 6(IRAB), 14$                      ; 0475
                05    A9                20  8A 000D9          BICB2   #32, 5(IRAB)                          ; 0478
                      52     00BC       C9  3C 000DD          MOVZWL  188(IRAB), R2                         ; 0486
                      51     00B0       C9  D0 000E2          MOVL    176(IRAB), R1                         ;
                         00000000G      00  16 000E7          JSB     RM$UNLOCK_HARD                        ;
                      29                50  E8 000ED          BLBS    STATUS, 16$                           ;
                                        19  11 000F0          BRB     15$                                   ;
                      52     00BC       C9  3C 000F2  14$:    MOVZWL  188(IRAB), R2                         ; 0497
                      51     00B0       C9  D0 000F7          MOVL    176(IRAB), R1                         ;
                         00000000G      00  16 000FC          JSB     RM$QUERY_HARD                         ;
         00008039     8F                50  D1 00102          CMPL    R0, #32825                            ; 0498
                                        0E  13 00109          BEQL    16$                                   ;
```

```
                     06   A9              08  8A 0010B 15$:    BICB2   #8, 6(IRAB)                          ; 0501
                                    0000G 30  0010F            BSBW    RM$CLEAN_BDB                         ; 0502
                          50    81A0    8F  3C 00112          MOVZWL  #33184, R0                           ; 0503
                                        03  11 00117          BRB     17$                                  ;
                          5C            01  D0 00119 16$:     MOVL    #1, R0                               ; 0508
                                        1C  BA 0011C 17$:     POPR    #^M<R2,R3,R4>                        ; 0510
                                            05 0011E          RSB                                          ;

; Routine Size:  287 bytes,    Routine Base:  RM$RMS3 + 0000

;  448          0511  1
;  449          0512  1 END
;  450          0513  1
;  451          0514  0 ELUDOM



;                                  PSECT SUMMARY
;
;       Name                     Bytes                          Attributes
;
;  RM$RMS3                        287  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         46         1         154        00:00.4



;                                     COMMAND QUALIFIERS
;
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3UPDDEL/OBJ=OBJ$:RM3UPDDEL MSRC$:RM3UPDDEL/UPDATE=(ENH$:RM3UPDDEL)

; Size:          287 code + 0 data bytes
; Run Time:        00:08.6
; Elapsed Time:    00:30.3
; Lines/CPU Min:    3581
; Lexemes/CPU-Min: 15554
; Memory Used:  118 pages
; Compilation Complete
```

RM3SSIDR
LIS

RM3UPSIDX
LIS

RM3SRCHKY
LIS

RM3UPDDEL
LIS

RM3UPDATE
LIS