```
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    FFFFFFFFFF  NN      NN  DDDDDDDD    RRRRRRRR    RRRRRRRR    VV      VV
RRRRRRRR    MM      MM    333333    FFFFFFFFFF  NN      NN  DDDDDDDD    RRRRRRRR    RRRRRRRR    VV      VV
RR      RR  MMMM  MMMM  33      33  FF          NN      NN  DD      DD  RR      RR  RR      RR  VV      VV
RR      RR  MMMM  MMMM  33      33  FF          NN      NN  DD      DD  RR      RR  RR      RR  VV      VV
RR      RR  MM  MM  MM        33    FF          NNNN    NN  DD      DD  RR      RR  RR      RR  VV      VV
RR      RR  MM  MM  MM        33    FF          NNNN    NN  DD      DD  RR      RR  RR      RR  VV      VV
RRRRRRRR    MM      MM      33      FFFFFFFF    NN  NN  NN  DD      DD  RRRRRRRR    RRRRRRRR    VV      VV
RRRRRRRR    MM      MM      33      FFFFFFFF    NN  NN  NN  DD      DD  RRRRRRRR    RRRRRRRR    VV      VV
RR  RR      MM      MM        33    FF          NN    NNNN  DD      DD  RR  RR      RR  RR      VV      VV
RR  RR      MM      MM        33    FF          NN    NNNN  DD      DD  RR  RR      RR  RR      VV      VV
RR    RR    MM      MM  33    33    FF          NN      NN  DD      DD  RR    RR    RR    RR      VV  VV      ....
RR    RR    MM      MM  33    33    FF          NN      NN  DD      DD  RR    RR    RR    RR      VV  VV      ....
RR      RR  MM      MM    333333    FF          NN      NN  DDDDDDDD    RR      RR  RR      RR      VV      ....
RR      RR  MM      MM    333333    FF          NN      NN  DDDDDDDD    RR      RR  RR      RR      VV      ....

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 RM3FNDRRV (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 !                Find record by RRV, taking indirection if neccessary
;    35          0035  1 !
;    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:        Christian Saether     CREATION DATE:   28-APR-78  13:21
;    45          0045  1 !
;    46          0046  1 !
;    47          0047  1 ! MODIFIED BY:
;    48          0048  1 !
;    49          0049  1 !       V03-005 MCN0009         Maria del C. Nasr       31-Mar-1983
;    50          0050  1 !               More linkages reorganization
;    51          0051  1 !
;    52          0052  1 !       V03-004 MCN0008         Maria del C. Nasr       24-Feb-1983
;    53          0053  1 !               Reorganize linkages
;    54          0054  1 !
;    55          0055  1 !       V03-003 TMK0002         Todd M. Katz            30-Jan-1983
;    56          0056  1 !               Add support for Recovery Unit Journalling and RU ROLLBACK
;    57          0057  1 !               Recovery of ISAM files. This involves making a change to one                ; R
```

```
;    58          0058  1 !                of the error paths within RM$FIND_BY_RRV. When a RRV is deleted
;    59          0059  1 !                within a Recovery Unit, at some point in the history of the
;    60          0060  1 !                file (after the Recovery Unit has terminated successfully) the
;    61          0061  1 !                primary data record that this RU_DELETEd RRV points at might
;    62          0062  1 !                be deleted for good and its space reclaimed. If RM$FIND_BY_RRV
;    63          0063  1 !                were then to be called, a RRV error would be returned, and this
;    64          0064  1 !                is misleading. Therefore, when RMS isn't able to find the
;    65          0065  1 !                primary data record that an RRV points to, and after positioning
;    66          0066  1 !                back to the RRV finds that it is marked RU_DELETE, return an
;    67          0067  1 !                error of RMS$_DEL after reclaiming the space occupied by the RRV
;    68          0068  1 !                if the file has been opened for write access.
;    69          0069  1 !
;    70          0070  1 !        V03-002 KBT0291         Keith B. Thompson       23-Aug-1982
;    71          0071  1 !                Reorganize psects
;    72          0072  1 !
;    73          0073  1 !        V03-001 TMK0001         Todd M. Katz            17-Mar-1981
;    74          0074  1 !
;    75          0075  1 !        V03-001 TMK0001         Todd M. Katz            17-Mar-1981
;    76          0076  1 !                In order to solve the SIDR deadlock problem, the cache
;    77          0077  1 !                flag CSH$V_NOWAIT is set before this routine is called
;    78          0078  1 !                whenever we are attempting to access the primary data record
;    79          0079  1 !                by its RFA address from a SIDR. Therefore, the possibility
;    80          0080  1 !                exists that when we attempt to access the primary data bucket,
;    81          0081  1 !                we will recieve a record lock error in circumstances other than
;    82          0082  1 !                when the input parameter flag LOCK_ORIG caused the NOWAIT cache
;    83          0083  1 !                flag to be set when taking an indirection. The current behavior
;    84          0084  1 !                assumes that the LOCK_ORIG bit is set under such circumstances,
;    85          0085  1 !                that we should release our lock on the RRV bucket, try (with
;    86          0086  1 !                waiting) for the primary data bucket, and then reclaim the
;    87          0087  1 !                lock on the RRV bucket. This is not the desirable behavior when
;    88          0088  1 !                our original bucket was a SIDR. Instead we want to immediately
;    89          0089  1 !                return a RLK error when one is encountered. Therefore, add a
;    90          0090  1 !                further restriction so that after RM$GETBLK has been called
;    91          0091  1 !                and an error of RLK returned, that the original bucket
;    92          0092  1 !                represented by the BDB whose address is in IRAB[IRB$L_NEXTBDB]
;    93          0093  1 !                is not released, and another attempt is made to access the
;    94          0094  1 !                data record indirectly unless the input parameter flag
;    95          0095  1 !                LOCK_ORIG was also set.
;    96          0096  1 !
;    97          0097  1 !        V02-008 MCN0007         Maria del C. Nasr       27-Apr-1981
;    98          0098  1 !                Change input parameters to separate record identifier from
;    99          0099  1 !                lock flag.  Also define macro to detect prologue version and
;   100          0100  1 !                use correct record id.
;   101          0101  1 !
;   102          0102  1 !        V02-007 MCN0006         Maria del C. Nasr       16-Mar-1981
;   103          0103  1 !                Increase size of record identifier to a word in the local
;   104          0104  1 !                internal structures.
;   105          0105  1 !
;   106          0106  1 !        V02-006 REFORMAT        K. E. Kinnear           23-Jul-1980  9:58
;   107          0107  1 !
;   108          0108  1 ! REVISION HISTORY:
;   109          0109  1 !
;   110          0110  1 !        V01-005                 W. Koenig               24-Oct-1978  8:40
;   111          0111  1 !                Make changes caused by sharing conventions.
;   112          0112  1 !
;   113          0113  1 !        V01-004                 C. D. Saether           29-Sep-1978  13:57
;   114          0114  1 !                Complete Rewrite.
```

```
;  115          0115  1 !
;  116          0116  1 !       V01-003                 C. D. Saether           28-Aug-1978  15:23
;  117          0117  1 !               Fix logic on error pass.
;  118          0118  1 !
;  119          0119  1 !*****
;  120          0120  1
;  121          0121  1 LIBRARY 'RMSLIB:RMS';
;  122          0122  1
;  123          0123  1 REQUIRE 'RMSSRC:RMSIDXDEF';
;  124          0188  1
;  125          0189  1 ! Define default PSECTS for code.
;  126          0190  1 !
;  127          0191  1 PSECT
;  128          0192  1     CODE = RM$RMS3(PSECT_ATTR),
;  129          0193  1     PLIT = RM$RMS3(PSECT_ATTR);
;  130          0194  1
;  131          0195  1 ! Linkages.
;  132          0196  1 !
;  133          0197  1 LINKAGE
;  134          0198  1     L_RABREG_457,
;  135          0199  1     L_RABREG_567,
;  136          0200  1     L_RABREG_67,
;  137          0201  1     L_RABREG_7,
;  138          0202  1     L_PRESERVE1;
;  139          0203  1
;  140          0204  1 ! External Routines.
;  141          0205  1 !
;  142          0206  1 EXTERNAL ROUTINE
;  143          0207  1     RM$FIND_BY_ID       : RL$RABREG_567,
;  144          0208  1     RM$GETBKT           : RL$RABREG_457,
;  145          0209  1     RM$KEY_DESC         : RL$RABREG_7,
;  146          0210  1     RM$RECORD_VBN       : RL$PRESERVE1,
;  147          0211  1     RM$RLSBKT           : RL$PRESERVE1,
;  148          0212  1     RM$RU_RECLAIM       : RL$RABREG_67;
;  149          0213  1
;  150          0214  1 MACRO
;  151          0215  1     LOCK_ORIG    = FLAGS[0,0,1,0] %,
;  152          0216  1     ST           = TMP1[0,0,16,0] %,
;  153          0217  1     PTR_ID       = TMP1[2,0,16,0] %,
;  154          0218  1     LOOP_CONTROL = TMP1[4,0,8,0] %,
;  155          0219  1     INDIRECT     = TMP1[4,0,1,0] %,
;  156          0220  1     ERROR_PASS   = TMP1[4,1,1,0] %,
;  157          0221  1     IND_DELETED  = TMP1[4,2,1,0] %;
;  158          0222  1
```

```
;  160     0223  1 GLOBAL ROUTINE RM$FIND_BY_RRV (VBN, ID, FLAGS) : RL$RABREG_67 =
;  161     0224  1
;  162     0225  1 !++
;  163     0226  1 !
;  164     0227  1 ! FUNCTIONAL DESCRIPTION:
;  165     0228  1 !
;  166     0229  1 ! Using the VBN and ID passed as input parameters, search the
;  167     0230  1 ! bucket for the desired record.  If the record found is an RRV, take
;  168     0231  1 ! the indirection to the bucket pointed to after first releasing the
;  169     0232  1 ! original bucket. Return with bucket accessed and REC_ADDR pointing
;  170     0233  1 ! to the user data record.
;  171     0234  1 !
;  172     0235  1 ! If the low bit of the FLAGS parameter is set (LOCK_ORIG), then the
;  173     0236  1 ! original bucket is saved in NXTBDB and not released if the indirection
;  174     0237  1 ! is taken.
;  175     0238  1 !
;  176     0239  1 ! CALLING SEQUENCE:
;  177     0240  1 !
;  178     0241  1 !       RM$FIND_BY_RRV (VBN,ID,FLAGS)
;  179     0242  1 !
;  180     0243  1 ! INPUT PARAMETERS:
;  181     0244  1 !
;  182     0245  1 !       VBN - of record to search for
;  183     0246  1 !       ID - contains ID to search for
;  184     0247  1 !       FLAGS - low bit (LOCK_ORIG) set if original
;  185     0248  1 !               bucket is to be retained if indirection taken
;  186     0249  1 !
;  187     0250  1 ! IMPLICIT INPUTS:
;  188     0251  1 !
;  189     0252  1 !       COMMON_RABREG - registers used by GETBKT, GETNEXT_REC
;  190     0253  1 !
;  191     0254  1 ! OUTPUT PARAMETERS:
;  192     0255  1 !
;  193     0256  1 !       NONE
;  194     0257  1 !
;  195     0258  1 ! IMPLICIT OUTPUTS:
;  196     0259  1 !
;  197     0260  1 !       REC_ADDR - address of record found
;  198     0261  1 !       IFAB[IFB$B_PLG_VER] - prologue version
;  199     0262  1 !       IRAB[IRB$L_CURBDB] - address of current BDB, contains data record
;  200     0263  1 !       IRAB[IRB$L_NXTBDB] - address of BDB referencing original bucket, if
;  201     0264  1 !                            LOCK_ORIG specified and indirection taken
;  202     0265  1 !                            otherwise not modified
;  203     0266  1 !
;  204     0267  1 ! ROUTINE VALUE:
;  205     0268  1 !
;  206     0269  1 !       SUC - data record successfully found, REC_ADDR pointing
;  207     0270  1 !               to record in bucket referenced by CURBDB
;  208     0271  1 !
;  209     0272  1 !       ERRORS:
;  210     0273  1 !
;  211     0274  1 !               CURBDB and NXTBDB (if appropriate) are released and zeroed
;  212     0275  1 !
;  213     0276  1 !       RFA - bad bucket level (i.e, neq 0)
;  214     0277  1 !       RNF - record not found
;  215     0278  1 !       DEL - record deleted
;  216     0279  1 !       RRV - record pointer mismatchs
```

```
;  217          0280  1 !       plus assorted I/O error codes
;  218          0281  1 !
;  219          0282  1 ! SIDE EFFECTS:
;  220          0283  1 !
;  221          0284  1 !       On any error condition, CURBDB is zeroed and bucket is released.
;  222          0285  1 !       No check made for RLSBKT errors.
;  223          0286  1 !       AP is blown across this routine.
;  224          0287  1 !       IRAB[ PTR_VBN ] is used if indirection taken, otherwise not
;  225          0288  1 !
;  226          0289  1 !--
;  227          0290  1
;  228          0291  2     BEGIN
;  229          0292  2
;  230          0293  2     LABEL
;  231          0294  2         ALOOP,
;  232          0295  2         BLOOP;
;  233          0296  2
;  234          0297  2     BUILTIN
;  235          0298  2         AP;
;  236          0299  2
;  237          0300  2     MAP
;  238          0301  2         FLAGS : BBLOCK;
;  239          0302  2
;  240          0303  2     MACRO
;  241        M 0304  2         ERROR (CODE) =
;  242        M 0305  2                 BEGIN
;  243        M 0306  2                 ST = RMSERR(CODE);
;  244        M 0307  2                 EXITLOOP
;  245          0308  2                 END %,
;  246          0309  2
;  247        M 0310  2         EXIF_LOCK_ORIG =
;  248        M 0311  2
;  249        M 0312  2                 IF .LOCK_ORIG
;  250        M 0313  2                 THEN
;  251        M 0314  2
;  252        M 0315  2                     IF (BDB = .IRAB[IRB$L_NXTBDB]) NEQ 0
;  253        M 0316  2
;  254        M 0317  2                     ! release the original bucket (NXTBDB) if it is still accessed
;  255        M 0318  2                     !
;  256        M 0319  2                     THEN
;  257        M 0320  2                         BEGIN
;  258        M 0321  2                         RM$RLSBKT(0);
;  259        M 0322  2                         IRAB[IRB$L_NXTBDB] = 0;
;  260        M 0323  2                         BDB = .IRAB[IRB$L_CURBDB];
;  261        M  324  2                         EXITLOOP
;  262          0325  2                         END
;  263        M 0326  2                     ELSE
;  264        M 0327  2
;  265        M 0328  2                         ! If original bucket had been released, then put CURBDB
;  266        M 0329  2                         ! back into BDB and continue.
;  267        M 0330  2                         !
;  268          0331  2                         BDB = .IRAB[IRB$L_CURBDB]; %,
;  269          0332  2
;  270          0333  2         ! Determine if byte or word record identifier, depending on prologue
;  271          0334  2         ! version of file.
;  272          0335  2         !
;  273        M 0336  2         IRC$_RRV_ID =
```

```
;  274          M 0337  2                (IF .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  275          M 0338  2                THEN
;  276          M 0339  2                    .REC_ADDR[IRC$B_RRV_ID]
;  277          M 0340  2                ELSE
;  278            0341  2                    .REC_ADDR[IRC$W_RRV_ID]) %,
;  279            0342  2
;  280          M 0343  2            MAKE_ERR_PASS =
;  281          M 0344  2                BEGIN
;  282          M 0345  2
;  283          M 0346  2                ! 2 is the error pass flag
;  284          M 0347  2                !
;  285          M 0348  2                LOOP_CONTROL = .LOOP_CONTROL OR 2;
;  286          M 0349  2                LEAVE BLOOP
;  287            0350  2                END %,
;  288            0351  2
;  289          M 0352  2            GO_INDIRECT =
;  290          M 0353  2                BEGIN
;  291          M 0354  2                LOOP_CONTROL = 1;                ! set control flags to INDIRECT
;  292          M 0355  2                PTR_ID = IRC$_RRV_ID;
;  293          M 0356  2                PTR_VBN = .REC_PTR_VBN;
;  294          M 0357  2
;  295          M 0358  2                IF .LOCK_ORIG
;  296          M 0359  2                THEN
;  297          M 0360  2                    BEGIN
;  298          M 0361  2                    IRAB[IRB$L_NXTBDB] = .BDB;
;  299          M 0362  2                    IRAB[IRB$B_CACHEFLGS] = CSH$M_NOWAIT;
;  300          M 0363  2                    LEAVE ALOOP
;  301          M 0364  2                    END
;  302          M 0365  2                ELSE
;  303          M 0366  2                    LEAVE BLOOP
;  304            0367  2                END %,
;  305            0368  2
;  306          M 0369  2            PTR_VBN =
;  307            0370  2                    IRAB[IRB$L_PTR_VBN] %;
;  308            0371  2
;  309            0372  2            EXTERNAL REGISTER
;  310            0373  2                COMMON_RAB_STR,
;  311            0374  2                R_REC_ADDR_STR;
;  312            0375  2
;  313            0376  2            GLOBAL REGISTER
;  314            0377  2                R_IDX_DFN_STR,
;  315            0378  2                COMMON_IO_STR;
;  316            0379  2
;  317            0380  2            LOCAL
;  318            0381  2                SAV_CFLAGS      : BYTE,
;  319            0382  2                TMP1            : BBLOCK [5];
;  320            0383  2
;  321            0384  2            CH$FILL(0,5,TMP1);                              ! initialize LOOP_CONTROL
;  322            0385  2
;  323            0386  2            ! Save cacheflags for future passes and set up index descriptor for key 0.
;  324            0387  2            !
;  325            0388  2            SAV_CFLAGS = .IRAB[IRB$B_CACHEFLGS];
;  326            0389  2            IRAB[IRB$B_CACHEFLGS] = 0;
;  327            0390  2            RM$KEY_DESC(0);
;  328            0391  2
;  329            0392  2            DO
;  330            0393  2
```

```
;  331     0394  2      ! Leaving ALOOP causes another pass to be made with no additional action.
;  332     0395  2      !
;  333     0396  2 ALOOP :
;  334     0397  3          BEGIN
;  335     0398  3          IRAB[IRB$B_CACHEFLGS] = .IPAB[IRB$B_CACHEFLGS] OR .SAV_CFLAGS;
;  336     0399  3
;  337     0400  3          ! Leaving BLOOP releases the current BDB, resets cacheflags and makes
;  338     0401  3          ! another pass.
;  339     0402  3          !
;  340     0403  3 BLOOP :
;  341     0404  4          BEGIN
;  342     0405  4          ST =
;  343     0406  5              BEGIN
;  344     0407  5
;  345     0408  5              LOCAL
;  346     0409  5                  SIZE;
;  347     0410  5
;  348     0411  5              SIZE = .IDX_DFN[IDX$B_DATBKTSZ]*512;
;  349     0412  5
;  350     0413  5              IF .INDIRECT
;  351     0414  5              THEN
;  352     0415  5                  RM$GETBKT(.PTR_VBN, .SIZE)
;  353     0416  5              ELSE
;  354     0417  5                  RM$GETBKT(.VBN, .SIZE)
;  355     0418  5
;  356     0419  4              END;
;  357     0420  4
;  358     0421  4          IF NOT .ST
;  359     0422  4          THEN
;  360     0423  4
;  361     0424  4              IF    .LOCK_ORIG
;  362     0425  4                AND
;  363     0426  5              (.ST EQL RMSERR(RLK))
;  364     0427  4          THEN
;  365     0428  4
;  366     0429  4              ! One way to get an RLK error is when LOCK_ORIG caused the
;  367     0430  4              ! NOWAIT cacheflag to be set when taking an indirection.  In
;  368     0431  4              ! that case go indirect again without the NOWAIT set after
;  369     0432  4              ! releasing the original bucket first.  It will be picked up
;  370     0433  4              ! again later if the indirect pass is successful.
;  371     0434  4              !
;  372     0435  5              BEGIN
;  373     0436  5              BDB = .IRAB[IRB$L_NXTBDB];
;  374     0437  5              IRAB[IRB$L_NXTBDB] = 0;
;  375     0438  5              LEAVE BLOOP;
;  376     0439  5
;  377     0440  5              END
;  378     0441  4          ELSE
;  379     0442  4
;  380     0443  4              ! This is most likely a hardware failure so get out and make
;  381     0444  4              ! sure everything is released.
;  382     0445  4              !
;  383     0446  4              ! Another possibility is that we encountered a RLK error when
;  384     0447  4              ! attempting to access a primary data bucket from a SIDR.  In
;  385     0448  4              ! this case we want to just return the error status regardless
;  386     0449  4              ! if we were going indirect or direct at the time.
;  387     0450  4              !
```

```
;  388     0451  5                BEGIN
;  389     0452  5
;  390     0453  5                IF .LOCK_ORIG AND (BDB = .IRAB[IRB$L_NXTBDB]) NEQ 0
;  391     0454  5                THEN
;  392     0455  5                    EXITLOOP;           ! causing NXTBDB to be released
;  393     0456  5
;  394     0457  5                IRAB[IRB$L_CURBDB] = 0;
;  395     0458  5                RETURN .ST;
;  396     0459  5
;  397     0460  4                END;
;  398     0461  4
;  399     0462  4            ! The bucket is accessed successfully.  Save the current BDB and
;  400     0463  4            ! continue.
;  401     0464  4            !
;  402     0465  4            IRAB[IRB$L_CURBDB] = .BDB;
;  403     0466  4
;  404     0467  4            !                   *** NOTE ***
;  405     0468  4            ! that with reallocation of buckets to different levels,
;  406     0469  4            ! it may be possible to get this condition on an indirect pass.  This
;  407     0470  4            ! is not currently implemented and therefore not currently checked for.
;  408     0471  4            !
;  409     0472  4
;  410     0473  4            IF .BKT_ADDR[BKT$B_LEVEL] NEQ 0
;  411     0474  4            THEN
;  412     0475  4
;  413     0476  4                ! Exit with RFA error, making sure that original bucket is released
;  414     0477  4                ! if indirection taken with LOCK_ORIG.
;  415     0478  4                !
;  416     0479  5                BEGIN
;  417     0480  5                ST = RMSERR(RFA);
;  418     0481  5
;  419     0482  5                IF .INDIRECT
;  420     0483  5                THEN
;  421     0484  5                    EXIF_LOCK_ORIG;
;  422     0485  5
;  423     0486  5                EXITLOOP;
;  424     0487  5
;  425     0488  4                END;
;  426     0489  4
;  427     0490  4            ! Load AP with the appropriate ID for the FIND_BY_ID search of this
;  428     0491  4            ! bucket.
;  429     0492  4            !
;  430     0493  4
;  431     0494  4            IF .INDIRECT
;  432     0495  4            THEN
;  433     0496  4                AP = .PTR_ID
;  434     0497  4            ELSE
;  435     0498  4                AP = .ID;
;  436     0499  4
;  437     0500  4            ST = RMSFIND_BY_ID();
;  438     0501  4            AP = 3;             ! initialize for subsequent calls to RECORD_VBN
;  439     0502  4
;  440     0503  4            IF .INDIRECT
;  441     0504  4            THEN
;  442     0505  4
;  443     0506  4                ! This code is executed on the indirect pass.  In LOCK_ORIG mode,
;  444     0507  4                ! an error condition will cause an exit if the original bucket is
```

```
;  445     0508  4          ! still accessed. Otherwise, an error pass back to the original
;  446     0509  4          ! bucket is made to confirm that the pointers to this bucket have
;  447     0510  4          ! not changed.
;  448     0511  4          !
;  449     0512  5          BEGIN
;  450     0513  5          LOOP_CONTROL = 0;
;  451     0514  5
;  452     0515  5          IF NOT .ST
;  453     0516  5          THEN
;  454     0517  6              BEGIN
;  455     0518  6              EXIF_LOCK_ORIG;
;  456     0519  6              MAKE_ERR_PASS;
;  457     0520  5              END;
;  458     0521  5
;  459     0522  5          IF .REC_ADDR[IRC$V_DELETED]
;  460     0523  5          THEN
;  461     0524  6              BEGIN
;  462     0525  6              IND_DELETED = 1;
;  463     0526  6              ST = RMSERR(DEL);
;  464     0527  6              EXIF_LOCK_ORIG;
;  465     0528  6              MAKE_ERR_PASS;
;  466     0529  5              END;
;  467     0530  5
;  468     0531  5          IF .REC_ADDR[IRC$V_RRV]
;  469     0532  5          THEN
;  470     0533  6              BEGIN
;  471     0534  6              ST = RMSERR(RRV);
;  472     0535  6              EXIF_LOCK_ORIG;
;  473     0536  6              MAKE_ERR_PASS;
;  474     0537  5              END;
;  475     0538  5
;  476     0539  6          IF (IRC$_RRV_ID NEQ .ID)
;  477     0540  5              OR
;  478     0541  6              (RM$RECORD_VBN() NEQ .VBN)
;  479     0542  5          THEN
;  480     0543  6              BEGIN
;  481     0544  6              ST = RMSERR(RRV);
;  482     0545  6              EXIF_LOCK_ORIG;
;  483     0546  6              MAKE_ERR_PASS;
;  484     0547  5              END;
;  485     0548  5
;  486     0549  5          ! If we have gotten this far, we have successfully found the
;  487     0550  5          ! correct record taking the indirection.  If LOCK_ORIG mode, we
;  488     0551  5          ! must get back the original bucket if we had to release it to get
;  489     0552  5          ! this one, otherwise just exit (STATUS contains success).
;  490     0553  5          !
;  491     0554  5
;  492     0555  5          IF .LOCK_ORIG AND .IRAB[IRB$L_NXTBDB] EQL 0
;  493     0556  5          THEN
;  494     0557  6              BEGIN
;  495     0558  6              IRAB[IRB$B_CACHEFLGS] = .SAV_CFLAGS;
;  496     0559  6              ST = RM$GETBKT(.VBN, .IDX_DFN[IDX$B_DATBKTSZ]*512);
;  497     0560  6
;  498     0561  6              IF .ST
;  499     0562  6              THEN
;  500     0563  7                  BEGIN
;  501     0564  7                  IRAB[IRB$L_NXTBDB] = .BDB;
```

```
;  502     0565  7                    RETURN .ST;
;  503     0566  7
;  504     0567  7                    END
;  505     0568  6                ELSE
;  506     0569  7                    BEGIN
;  507     0570  7                    BDB = .IRAB[IRB$L_CURBDB];
;  508     0571  7                    EXITLOOP;
;  509     0572  7
;  510     0573  6                    END;
;  511     0574  6
;  512     0575  6                END
;  513     0576  5            ELSE
;  514     0577  5                RETURN .ST;
;  515     0578  5
;  516     0579  4            END;                                ! of INDIRECT pass code
;  517     0580  4
;  518     0581  4        ! This is a direct pass, i.e., we are looking at the bucket described
;  519     0582  4        ! by the input VBN and ID.  Note that not finding the original record
;  520     0583  4        ! if on an error pass is a bug for prologue version 1, but may not be
;  521     0584  4        ! one if RRV's can be purged when the record is deleted.  At any rate,
;  522     0585  4        ! I'm not checking for it.
;  523     0586  4        !
;  524     0587  4
;  525     0588  4        IF NOT .ST
;  526     0589  4        THEN
;  527     0590  4            EXITLOOP;
;  528     0591  4
;  529     0592  4        IF .REC_ADDR[IRC$V_DELETED]
;  530     0593  4        THEN
;  531     0594  4            ERROR(DEL);
;  532     0595  4
;  533     0596  5        BEGIN
;  534     0597  5
;  535     0598  5        LOCAL
;  536     0599  5            REC_PTR_VBN;
;  537     0600  5
;  538     0601  5        REC_PTR_VBN = RM$RECORD_VBN();
;  539     0602  5
;  540     0603  5        ! If this is an error pass and the indirect pointers are still the
;  541     0604  5        ! same, then we have a real error, otherwise just go indirect and try
;  542     0605  5        ! again.
;  543     0606  5        !
;  544     0607  5
;  545     0608  5        IF .ERROR_PASS
;  546     0609  5        THEN
;  547     0610  5
;  548     0611  6            IF (.PTR_ID EQL IRC$_RRV_ID)
;  549     0612  5                AND
;  550     0613  6                (.PTR_VBN EQL .REC_PTR_VBN)
;  551     0614  5            THEN
;  552     0615  5
;  553     0616  5                IF .IND_DELETED
;  554     0617  5                THEN
;  555     0618  6                    ERROR(DEL)
;  556     0619  5                ELSE
;  557     0620  5
;  558     0621  5                    ! If the RRV was deleted within a Recovery Unit (which has
```

```
;  559     0622  5                ! since succesfully completed, or the primary data record
;  560     0623  5                ! this RRV pointed at would not have been reclaimed), then
;  561     0624  5                ! reclaim the space it occupies (if the file has been open
;  562     0625  5                ! for write access) befor returning an error of RMS$_DEL.
;  563     0626  5                !
;  564     0627  5                IF .REC_ADDR[IRC$V_RU_DELETE]
;  565     0628  5                THEN
;  566     0629  6                    BEGIN
;  567     0630  6
;  568     0631  6                    IF .IFAB[IFB$V_WRTACC]
;  569     0632  6                    THEN
;  570     0633  6                        RM$RU_RECLAIM();
;  571     0634  6
;  572     0635  6                    ERROR(DEL);
;  573     0636  6                    END
;  574     0637  5                ELSE
;  575     0638  6                    ERROR(RRV)
;  576     0639  5            ELSE
;  577     0640  5                GO_INDIRECT;
;  578     0641  5
;  579     0642  5        ! This is not an error pass so check if the record is an RRV.
;  580     0643  5        !
;  581     0644  5
;  582     0645  5        IF .REC_ADDR[IRC$V_RRV]
;  583     0646  5        THEN
;  584     0647  5            GO_INDIRECT;
;  585     0648  5
;  586     0649  5        ! Record is not an RRV, so if the back pointers match, this is the
;  587     0650  5        ! record we want, otherwise return an RRV error.
;  588     0651  5        !
;  589     0652  5
;  590     0653  6            IF (IRC$_RRV_ID EQL .ID)
;  591     0654  5                AND
;  592     0655  6                (.REC_PTR_VBN EQL .VBN)
;  593     0656  5            THEN
;  594     0657  6                BEGIN
;  595     0658  6                RETURN .ST;
;  596     0659  6
;  597     0660  6                END
;  598     0661  5            ELSE
;  599     0662  5                ERROR(RRV);
;  600     0663  5
;  601     0664  4            END;                                ! of block defining REC_PTR_VBN
;  602     0665  3            END;                                    ! of BLOOP
;  603     0666  3
;  604     0667  3            ! We have left BLOOP so release the bucket, reset cacheflags and go
;  605     0668  3            ! again.
;  606     0669  3            !
;  607     0670  3            RM$RLSBKT(0);
;  608     0671  3            END                                         ! of ALOOP
;  609     0672  2        UNTIL 0;                        ! an EXITLOOP or RETURN is the only way out
;  610     0673  2
;  611     0674  2        ! This code executed on an EXITLOOP
;  612     0675  2        !
;  613     0676  2        RM$RLSBKT(0);
;  614     0677  2        IRAB[IRB$L_CURBDB] = 0;
;  615     0678  2        RETURN .ST
```

```
;  616         0679  2
;  617         0680  1     END;


                                                  .TITLE  RM3FNDRRV
                                                  .IDENT  \V04-000\

                                                  .EXTRN  RM$FIND_BY_ID, RM$GETBKT
                                                  .EXTRN  RM$KEY_DESC, RM$RECORD_VBN
                                                  .EXTRN  RM$RLSBKT, RM$RU_RECLAIM

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

                         00BC   8F  BB 00000 RM$FIND_BY_RRV::
                                                  PUSHR   #^M<R2,R3,R4,R5,R7>                ; 0223
                    5E          08  C2 00004      SUBL2   #8, SP
     05        00   6E          00  2C 00007      MOVC5   #0, (SP), #0, #5, TMP1              ; 0384
                                6E     0000C
                    53     40   A9  90 0000D      MOVB    64(IRAB), SAV_CFLAGS                ; 0388
                           40   A9  94 00011      CLRB    64(IRAB)                            ; 0389
                                7E  D4 00014      CLRL    -(SP)                               ; 0390
                             0000G  30 00016      BSBW    RM$KEY_DESC
                    5E          04  C0 00019 1$:  ADDL2   #4, SP
               40   A9          53  88 0001C 2$:  BISB2   SAV_CFLAGS, 64(IRAB)                ; 0398
                    50     17   A7  9A 00020      MOVZBL  23(IDX_DFN), SIZE                   ; 0411
          50        50          09  78 00024      ASHL    #9, SIZE, SIZE
                    07     04   AE  E9 00028      BLBC    TMP1+4, 3$                          ; 0413
                                50  DD 0002C      PUSHL   SIZE                                ; 0415
                           4C   A9  DD 0002E      PUSHL   76(IRAB)
                                05  11 00031      BRB     4$
                                50  DD 00033 3$:  PUSHL   SIZE                                ; 0417
                           24   AE  DD 00035      PUSHL   VBN
                             0000G  30 00038 4$:  BSBW    RM$GETBKT
                    5E          08  C0 0003B      ADDL2   #8, SP
                    6E          50  B0 0003E      MOVW    R0, TMP1                            ; 0413
                    25          6E  E8 00041      BLBS    TMP1, 8$                            ; 0421
                    18     28   AE  E9 00044      BLBC    FLAGS, 6$                           ; 0424
             82AA   8F          6E  B1 00048      CMPW    TMP1, #33450                        ; 0426
                                0A  12 0004D      BNEQ    5$
                    54     3C   A9  D0 0004F      MOVL    60(IRAB), BDB                       ; 0436
                           3C   A9  D4 00053      CLRL    60(IRAB)                            ; 0437
                              0184  31 00056      BRW     39$                                 ; 0438
                    06     28   AE  E9 00059 5$:  BLBC    FLAGS, 6$                           ; 0453
                    54     3C   A9  D0 0005D      MOVL    60(IRAB), BDB
                                03  12 00061      BNEQ    7$
                              0187  31 00063 6$:  BRW     41$
                              017C  31 00066 7$:  BRW     40$
               20   A9          54  D0 00069 8$:  MOVL    BDB, 32(IRAB)                       ; 0465
                           0C   A5  95 0006D      TSTB    12(BKT_ADDR)                        ; 0473
                                15  13 00070      BEQL    9$
                    6E   865C   8F  B0 00072      MOVW    #-31140, TMP1                       ; 0480
                    EB     04   AE  E9 00077      BLBC    TMP1+4, 7$                          ; 0482
                    E7     28   AE  E9 0007B      BLBC    FLAGS, 7$                           ; 0483
                    54     3C   A9  D0 0007F      MOVL    60(IRAB), BDB
                                73  13 00083      BEQL    19$
                                66  11 00085      BRB     18$
                    06     04   AE  E9 00087 9$:  BLBC    TMP1+4, 10$                         ; 0494
```

```
                5C      02    AE   3C 0008B          MOVZWL  TMP1+2, AP                    ; 0496
                              04   11 0008F          BRB     11$
                5C      24    AE   D0 00091 10$:     MOVL    ID, AP                      ; 0498
                        0000G       30 00095 11$:     BSBW    RM$FIND_BY_ID               ; 0500
                6E              50   B0 00098          MOVW    R0, TMP1
                5C              03   D0 0009B          MOVL    #3, AP                      ; 0501
                03      04    AE   E8 0009E          BLBS    TMP1+4, 12$                 ; 0503
                        0091        31 000A2          BRW     25$
                        04    AE    94 000A5 12$:     CLRB    TMP1+4                      ; 0513
                38              6E   E9 000A8          BLBC    TMP1, 17$                   ; 0515
          0B    66              02   E1 000AB          BBC     #2, (REC_ADDR), 13$         ; 0522
          04    AE              04   88 000AF          BISB2   #4, TMP1+4                  ; 0525
                6E      8262    8F   B0 000B3          MOVW    #-32158, TMP1               ; 0526
                              29   11 000B8          BRB     17$
          20    66              03   E0 000BA 13$:     BBS     #3, (REC_ADDR), 16$         ; 0531
                03      00B7    CA   91 000BE          CMPB    183(IFAB), #3               ; 0539
                              06   1E 000C3          BGEQU   14$
                50      02    A6   9A 000C5          MOVZBL  2(REC_ADDR), R0
                              04   11 000C9          BRB     15$
                50      03    A6   3C 000CB 14$:     MOVZWL  3(REC_ADDR), R0
          24    AE              50   D1 000CF 15$:     CMPL    R0, ID
                              09   12 000D3          BNEQ    16$
                        0000G       30 000D5          BSBW    RM$RECORD_VBN               ; 0541
          20    AE              50   D1 000D8          CMPL    R0, VBN
                              27   13 000DC          BEQL    22$
                6E      8684    8F   B0 000DE 16$:     MOVW    #-31100, TMP1               ; 0544
                17      28    AE   E9 000E3 17$:     BLBC    FLAGS, 21$
                54      3C    A9   D0 000E7          MOVL    60(IRAB), BDB
                              0D   13 000EB          BEQL    20$
                              7E   D4 000ED 18$:     CLRL    -(SP)
                        0000G       30 000EF          BSBW    RM$RLSBKT
                5E              04   C0 000F2          ADDL2   #4, SP
                        3C    A9    D4 000F5          CLRL    60(IRAB)
                              36   11 000F8 19$:     BRB     24$
                54      20    A9   D0 000FA 20$:     MOVL    32(IRAB), BDB
          04    AE              02   88 000FE 21$:     BISB2   #2, TMP1+4                  ; 0545
                        00D8        31 00102          BRW     39$
                24      28    AE   E9 00105 22$:     BLBC    FLAGS, 23$                  ; 0555
                        3C    A9    D5 00109          TSTL    60(IRAB)
                              1F   12 0010C          BNEQ    23$
          40    A9              53   90 0010E          MOVB    SAV_CFLAGS, 64(IRAB)        ; 0558
                50      17    A7   9A 00112          MOVZBL  23(IDX_DFN), R0             ; 0559
    7E          50              09   78 00116          ASHL    #9, R0, -(SP)
                        24    AE    DD 0011A          PUSHL   VBN
                        0000G       30 0011D          BSBW    RM$GETBKT
                5E              08   C0 00120          ADDL2   #8, SP
                6E              50   B0 00123          MOVW    R0, TMP1
                07              6E   E9 00126          BLBC    TMP1, 24$                   ; 0561
          3C    A9              54   D0 00129          MOVL    BDB, 60(IRAB)               ; 0564
                        00C0        31 0012D 23$:     BRW     42$                         ; 0565
                54      20    A9   D0 00130 24$:     MOVL    32(IRAB), BDB               ; 0570
                              48   11 00134          BRB     29$                         ; 0569
                45              6E   E9 00136 25$:     BLBC    TMP1, 29$                   ; 0588
    3C          66              02   E0 00139          BBS     #2, (REC_ADDR), 28$         ; 0592
                        0000G       30 0013D          BSBW    RM$RECORD_VBN               ; 0601
                52              50   D0 00140          MOVL    R0, REC_PTR_VBN
    41    04    AE              01   E1 00143          BBC     #1, TMP1+4, 31$             ; 0608
```

```
                        51  D4 00148          CLRL    R1                              ; 0611
                03      00B7  CA 91 0014A     CMPB    183(IFAB), #3
                        08  1E 0014F         BGEQU   26$
                        51  D6 00151         INCL    R1
                50        02  A6 9A 00153     MOVZBL  2(REC_ADDR), R0
                        04  11 00157         BRB     27$
                50        03  A6 3C 00159 26$:  MOVZWL  3(REC_ADDR), R0
                50        02  AE B1 0015D 27$:  CMPW    TMP1+2, R0
                        1D  12 00161         BNEQ    30$
                52        4C  A9 D1 00163     CMPL    76(IRAB), REC_PTR_VBN           ; 0613
                        17  12 00167         BNEQ    30$
        0B      04      AE      02  E0 00169     BBS     #2, TMP1+4, 28$             ; 0616
        64              66      05  E1 0016E     BBC     #5, (REC_ADDR), 38$         ; 0627
                03        06  AA E9 00172     BLBC    6(IFAB), 28$                ; 0631
                      0000G 30 00176         BSBW    RM$RU_RECLAIM                   ; 0633
                6E      8262  8F B0 00179 28$:  MOVW    #-32158, TMP1               ; 0635
                        65  11 0017E 29$:  BRB     40$                             ; 0629
                04      AE      01  90 00180 30$:  MOVB    #1, TMP1+4                  ; 0639
                        17      51  E9 00184     BLBC    R1, 33$
                        0F  11 00187         BRB     32$
        2C              66      03  E1 00189 31$:  BBC     #3, (REC_ADDR), 35$         ; 0645
                04      AE      01  90 0018D     MOVB    #1, TMP1+4                  ; 0646
                03        00B7  CA 91 00191     CMPB    183(IFAB), #3
                        06  1E 00196         BGEQU   33$
                50        02  A6 9A 00198 32$:  MOVZBL  2(REC_ADDR), R0
                        04  11 0019C         BRB     34$
                50        03  A6 3C 0019E 33$:  MOVZWL  3(REC_ADDR), R0
        02      AE              50  B0 001A2 34$:  MOVW    R0, TMP1+2
        4C      A9              52  D0 001A6     MOVL    REC_PTR_VBN, 76(IRAB)
                2F        28    AE E9 001AA     BLBC    FLAGS, 39$
        3C      A9              54  D0 001AE     MOVL    BDB, 60(IRAB)
        40      A9              02  90 001B2     MOVB    #2, 64(IRAB)
                      FE63  31 001B6         BRW     2$
                03      00B7  CA 91 001B9 35$:  CMPB    183(IFAB), #3               ; 0653
                        06  1E 001BE         BGEQU   36$
                50        02  A6 9A 001C0     MOVZBL  2(REC_ADDR), R0
                        04  11 001C4         BRB     37$
                50        03  A6 3C 001C6 36$:  MOVZWL  3(REC_ADDR), R0
        24      AE              50  D1 001CA 37$:  CMPL    R0, ID
                        06  12 001CE         BNEQ    38$
        20      AE              52  D1 001D0     CMPL    REC_PTR_VBN, VBN            ; 0655
                        1A  13 001D4         BEQL    42$
                6E      8684  8F B0 001D6 38$:  MOVW    #-31100, TMP1               ; 0662
                        08  11 001DB         BRB     40$
                        7E  D4 001DD 39$:  CLRL    -(SP)                           ; 0670
                      0000G 30 001DF         BSBW    RM$RLSBKT
                      FE34  31 001E2         BRW     1$
                        7E  D4 001E5 40$:  CLRL    -(SP)                           ; 0676
                      0000G 30 001E7         BSBW    RM$RLSBKT
                5E              04  C0 001EA     ADDL2   #4, SP
                        20    A9 D4 001ED 41$:  CLRL    32(IRAB)                    ; 0677
                50              8E  3C 001F0 42$:  MOVZWL  TMP1, R0                    ; 0678
                5E              06  C0 001F3     ADDL2   #6, SP                      ; 0680
                      00BC  8F BA 001F6     POPR    #^M<R2,R3,R4,R5,R7>
                            05 001FA         RSB

; Routine Size:  507 bytes,    Routine Base:  RM$RMS3 + 0000
```

```
;  618          0681  1
;  619          0682  1 END
;  620          0683  1
;  621          0684  0 ELUDOM
```

PSECT SUMMARY

Name	Bytes	Attributes
RM$RMS3	507	NOVEC,NOWRT, RD , EXE,NOSHP, GBL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	Symbols Total	Symbols Loaded	Symbols Percent	Pages Mapped	Processing Time
_255DUA28:[RMS.OBJ]RMS.L32;1	3109	47	1	154	00:00.4

COMMAND QUALIFIERS

```
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3FNDRRV/OBJ=OBJ$:RM3FNDRRV MSRC$:RM3FNDRRV/UPDATE=(ENH$:RM3FNDRRV)

; Size:          507 code + 0 data bytes
; Run Time:        00:16.8
; Elapsed Time:    00:45.2
; Lines/CPU Min:    2447
; Lexemes/CPU-Min: 21935
; Memory Used:  247 pages
; Compilation Complete
```

RM3MAKIDX
LIS

RM3FNDRRV
LIS

RM3FNDRFA
LIS

RM3GET
LIS

RM3IUDR
LIS

RM3JOURNL
LIS

RM3KEYDSC
LIS

RM3MISC
LIS

RM3MISPUT
LIS