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

FILEID**RM3POSSEQ

```
RRRRRRR     MM       MM    333333    PPPPPPPP      000000      SSSSSSSS    SSSSSSSS  EEEEEEEEEE    QQQQQQ
RRRRRRRR    MM       MM   333333     PPPPPPPP     000000      SSSSSSSS    SSSSSSSS  EEEEEEEEEE    QQQQQQ
RR      RR  MMMM   MMMM  33      33   PP      PP  00      00  SS          SS        EE            QQ      QQ
RR      RR  MMMM   MMMM  33      33   PP      PP  00      00  SS          SS        EE            QQ      QQ
RR      RR  MM MM MM MM        33     PP      PP  00      00  SS          SS        EE            QQ      QQ
RR      RR  MM  MM  MM         33     PP      PP  00      00  SS          SS        EE            QQ      QQ
RRRRRRRR    MM      MM       33       PPPPPPPP    00      00   SSSSSS      SSSSSS   EEEEEEEE      QQ      QQ
RRRRRRRR    MM      MM       33       PPPPPPPP    00      00   SSSSSS      SSSSSS   EEEEEEEE      QQ      QQ
RR  RR      MM      MM         33     PP          00      00         SS          SS  EE            QQ  QQ  QQ
RR  RR      MM      MM         33     PP          00      00         SS          SS  EE            QQ  QQ  QQ
RR    RR    MM      MM  33     33     PP          00      00         SS          SS  EE            QQ    QQ
RR    RR    MM      MM  33     33     PP          00      00         SS          SS  EE            QQ    QQ
RR      RR  MM      MM    333333      PP            000000    SSSSSSSS    SSSSSSSS  EEEEEEEEEE    QQQQ  QQ
RR      RR  MM      MM    333333      PP            000000    SSSSSSSS    SSSSSSSS  EEEEEEEEEE    QQQQ  QQ

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 RM3POSSEQ (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 !               Routines to position sequentially to the next record
;    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:       Todd M. Katz    CREATION DATE:          23-Jun-1982
;    45          0045  1 !
;    46          0046  1 !
;    47          0047  1 ! MODIFIED BY:
;    48          0048  1 !
;    49          0049  1 !       V03-016 TSK0001         Tamar Krichevsky        15-Jun-1983
;    50          0050  1 !               Change addressing mode of RM$RU_RECLAIM to long relative.
;    51          0051  1 !
;    52          0052  1 !       V03-015 MCN0002         Maria del C. Nasr       24-Mar-1983
;    53          0053  1 !               More linkages reorganization.
;    54          0054  1 !
;    55          0055  1 !       V03-014 TMK0009         Todd M. Katz            11-Mar-1983
;    56          0056  1 !               Make sure that all bucket locks are taken out as exclusive
;    57          0057  1 !               whenever the possibility exists that some reclamation maybe
```

```
;    58          0058  1 !                 done. Reclamation is possible whenever the positioning is by
;    59          0059  1 !                 alternate key of reference, or whenever the file has been
;    60          0060  1 !                 write accessed and is Recovery Unit Journallable.
;    61          0061  1 !
;    62          0062  1 !       V03-013 MCN0001         Maria del C. Nasr       28-Feb-1983
;    63          0063  1 !                 Reorganize linkages
;    64          0064  1 !
;    65          0065  1 !       V03-012 TMK0008         Todd M. Katz            24-Feb-1983
;    66          0066  1 !                 Fix a bug in and add several performance enhancements to
;    67          0067  1 !                 RM$FIND_NONDEL. The bug was that when this routine exhausts the
;    68          0068  1 !                 current bucket in its search for a non-deleted primary data
;    69          0069  1 !                 record and continues with the next bucket it should be resetting
;    70          0070  1 !                 IRB$L_LST_NCMP to point to the first record in the next bucket
;    71          0071  1 !                 when primary key compression is enabled. Because it was not,
;    72          0072  1 !                 when the time came to re-expand the key of the non-deleted
;    73          0073  1 !                 primary data record positioned to, this key expansion was done
;    74          0074  1 !                 incorrectly. The enhancements consist of making sure as RMS
;    75          0075  1 !                 scans the current bucket for a non-deleted primary data record
;    76          0076  1
;    77          0077  1 !
;    78          0078  1 !       V03-011 TMK0007         Todd M. Katz            17-Jan-1983
;    79          0079  1 !                 Add support for Recovery Unit Journalling and RU ROLLBACK
;    80          0080  1 !                 Recovery of ISAM files. Support involves modifications to
;    81          0081  1 !                 RM$POS_SEQ and RM$FIND_NONDEL.
;    82          0082  1 !
;    83          0083  1 !                 The purpose of the routines within this module is to find the
;    84          0084  1 !                 next non-deleted primary data record sequentially. If during
;    85          0085  1 !                 its search RMS encounters records that are marked RU_DELETE, it
;    86          0086  1 !                 will try and delete them for good at this time provided it has
;    87          0087  1 !                 write access to the file and the Recovery Unit in which they
;    88          0088  1 !                 were deleted has completed successfully.
;    89          0089  1 !
;    90          0090  1 !                 If RMS is able to delete a primary data record marked RU_DELETE,
;    91          0091  1 !                 then RMS proceeds to continue looking for a non-deleted primary
;    92          0092  1 !                 data record just as if it had encountered a deleted record in
;    93          0093  1 !                 the first place. Likewise, if RMS is unable to delete a record
;    94          0094  1 !                 that is marked RU_DELETE because it does not have write access
;    95          0095  1 !                 to the file, it merely continues its search with the next
;    96          0096  1 !                 record. However, if RMS is unable to delete the record for good
;    97          0097  1 !                 because the Recovery Unit in which it was marked RU_DELETE has
;    98          0098  1 !                 not successfully terminated, then RMS returns this record as if
;    99          0099  1 !                 it was the next non-deleted primary data record, and lets a
;   100          0100  1 !                 higher level routine decide whether or not to wait for the
;   101          0101  1 !                 Recovery Unit in which the record was deleted to complete, or to
;   102          0102  1 !                 return an error to the user.
;   103          0103  1 !
;   104          0104  1 !                 RMS will also re-format any records that are marked RU_UPDATE
;   105          0105  1 !                 and are in a special format provided the stream has write access
;   106          0106  1 !                 to the file, and the Recovery Unit in which the record was
;   107          0107  1 !                 updated has terminated.
;   108          0108  1 !
;   109          0109  1 !       V03-010 TMK0006         Todd M. Katz            09-Nov-1982
;   110          0110  1 !                 The routine RM$FIND_SIDR never returns an end-of-file error so
;   111          0111  1 !                 remove all references to this particular error.
;   112          0112  1 !
;   113          0113  1 !       V03-009 TMK0005         Todd M. Katz            20-Oct-1982
;   114          0114  1 !                 An infinite loop existed in RM$POS_SEQ. When a search must be
```

```
;  115     0115  1 !                made for the current primary data record by calling
;  116     0116  1 !                RM$CSEARCH_TREE and duplicates are allowed, then the duplicate
;  117     0117  1 !                chain is searched for the current record. If the first duplicate
;  118     0118  1 !                in the chain was not the current record, then RMS was assuming
;  119     0119  1 !                that if it called RM$CSEARCH_TREE, this routine would position
;  120     0120  1 !                to the next record, and return status indicating whether it
;  121     0121  1 !                too was a duplicate. This assumption is incorrect. The routine
;  122     0122  1 !                RM$CSEARCH_TREE works as follows: If the routine
;  123     0123  1 !                RM$CSEARCH_TREE is called with REC_ADDR pointing to a place in
;  124     0124  1 !                a bucket, whose BDB's address is saved in IRB$L_CURBDB, then
;  125     0125  1 !                RM$CSEARCH_TREE determines whether or not the key of this
;  126     0126  1 !                record matches the key in keybuffer 2 according to the search
;  127     0127  1 !                characteristics. Thus, this routine does not position RMS to the
;  128     0128  1 !                very next record, which what was being incorrectly assumed, but
;  129     0129  1 !                just checks out the key of the current record. The fix is to
;  130     0130  1 !                explicitely position to the next primary data record if the
;  131     0131  1 !                record currently positioned to is not the current record, before
;  132     0132  1 !                calling RM$CSEARCH_TREE to see if this record is a member of the
;  133     0133  1 !                duplicate chain.
;  134     0134  1 !
;  135     0135  1 !        V03-008 TMK0004         Todd M. Katz            19-Oct-1982
;  136     0136  1 !                I have made two changes to the routine RM$POS_SEQ. First, if
;  137     0137  1 !                RMS was unable to position to any record using the current NRP
;  138     0138  1 !                information, then RMS should not be checking whether the record
;  139     0139  1 !                it has positioned to is a non-deleted RRV (because of course,
;  140     0140  1 !                RMS has not positioned to any record). At present, this check
;  141     0141  1 !                is made in RM$POS_SEQ. Second, RMS should not be releasing
;  142     0142  1 !                buckets, and then referencing addresses within them. This also
;  143     0143  1 !                is currently taking place within RM$POS_SEQ.
;  144     0144  1 !
;  145     0145  1 !        V03-007 TMK0003         Todd M. Katz            09-Sep-1982
;  146     0146  1 !                The field IRB$B_SRCHFLAGS is now a word in size. Fix all
;  147     0147  1 !                references to it.
;  148     0148  1 !
;  149     0149  1 !                Add support for prologue 3 SIDRs. A change is required to
;  150     0150  1 !                RM$POS_SEQ. When positioning sequentially for the first time
;  151     0151  1 !                following a $CONNECT or $REWIND. RMS wants to position to
;  152     0152  1 !                the first non-empty bucket. Because prolgoue 3 SIDRs do not have
;  153     0153  1 !                control bytes, RMS can't just check for a RRV record as its
;  154     0154  1 !                indicator of an empty bucket. It can only make this check for
;  155     0155  1 !                primary data buckets. This change has been made. Furthermore,
;  156     0156  1 !                RMS will also now check for an empty SIDR bucket (this was not
;  157     0157  1 !                being done) by checking the freespace offset pointer.
;  158     0158  1 !
;  159     0159  1 !        V03-006 KBT0296         Keith B. Thompson       24-Aug-1982
;  160     0160  1 !                Reorganize psects
;  161     0161  1 !
;  162     0162  1 !        V03-005 TMK0002         Todd M. Katz            28-Jul-1982
;  163     0163  1 !                When positioning sequentially by alternate key, the routine
;  164     0164  1 !                RM$FIND_SIDR is called to find the current SIDR, or to position
;  165     0165  1 !                to the next SIDR if the current SIDR has been deleted. If a
;  166     0166  1 !                stream's current SIDR is the last SIDR in the index, and it is
;  167     0167  1 !                deleted by another stream, RM$FIND_SIDR returns the error
;  168     0168  1 !                RMS$_RNF because it is unable to find any SIDR in the
;  169     0169  1 !                greater-than-or-equal search of the file it makes (using as the
;  170     0170  1 !                search key the key of the current record) when it decides that
;  171     0171  1 !                the stream's current SIDR has been deleted. In this case
```

```
;  172          0172  1 !               RM$POS_SEQ should convert this error into RM$_EOF and return
;  173          0173  1 !               it to indicate that the end of the file has been reached.
;  174          0174  1 !
;  175          0175  1 !       V03-004 TMK0001         Todd M. Katz            22-Jun-1982
;  176          0176  1 !               Revised entire module.
;  177          0177  1 !
;  178          0178  1 !*****
;  179          0179  1
;  180          0180  1 LIBRARY 'RMSLIB:RMS';
;  181          0181  1
;  182          0182  1 REQUIRE 'RMSSRC:RMSIDXDEF';
;  183          0247  1
;  184          0248  1 ! Define default PSECTS for code
;  185          0249  1 !
;  186          0250  1 PSECT
;  187          0251  1     CODE = RM$RMS3(PSECT_ATTR),
;  188          0252  1     PLIT = RM$RMS3(PSECT_ATTR);
;  189          0253  1
;  190          0254  1 ! Linkages
;  191          0255  1 !
;  192          0256  1 LINKAGE
;  193          0257  1     L_PRESERVE1,
;  194          0258  1     L_REC_OVHD,
;  195          0259  1     L_SIDR_FIRST,
;  196          0260  1     L_RABREG_457,
;  197          0261  1     L_RABREG_567,
;  198          0262  1     L_RABREG_67,
;  199          0263  1
;  200          0264  1     ! Local Linkages
;  201          0265  1     !
;  202          0266  1     RL$LINKAGE          = JSB ()
;  203          0267  1                         : GLOBAL (R_IDX_DFN, R_REC_ADDR, COMMON_RABREG);
;  204          0268  1
;  205          0269  1 ! External Routines
;  206          0270  1 !
;  207          0271  1 EXTERNAL ROUTINE
;  208          0272  1     RM$COMPARE_REC      : RL$RABREG_67,
;  209          0273  1     RM$CSEARCH_TREE     : RL$RABREG_67,
;  210          0274  1     RM$FIND_BY_RFA      : RL$RABREG_67,
;  211          0275  1     RM$FIND_BY_RRV      : RL$RABREG_67,
;  212          0276  1     RM$GETBKT           : RL$RABREG_457,
;  213          0277  1     RM$GETNEXT_REC      : RL$RABREG_67,
;  214          0278  1     RM$RECORD_ID        : RL$RABREG_67,
;  215          0279  1     RM$RECORD_VBN       : RL$PRESERVE1,
;  216          0280  1     RM$REC_OVHD         : RL$REC_OVHD,
;  217          0281  1     RM$RLSBKT           : RL$PRESERVE1,
;  218          0282  1     RM$RU_RECLAIM       : RL$RABREG_67 ADDRESSING_MODE( LONG_RELATIVE ),
;  219          0283  1     RM$SEARCH_SIDR      : RL$RABREG_67,
;  220          0284  1     RM$SIDR_FIRST       : RL$SIDR_FIRST,
;  221          0285  1     RM$SRCH_BY_KEY      : RL$RABREG_567;
;  222          0286  1
;  223          0287  1 ! Define a local macro to return on error and
;  224          0288  1 ! to change a 'RNF' error to 'EOF' ( this is seq. proccessing )
;  225          0289  1 !
;  226          0290  1 MACRO
;  227        M 0291  1     RET_ON_ERR_EOF (X) =
;  228        M 0292  1         (RETURN_ON_ERROR(X,
```

```
;  229          0293  1          IF .STATUS<0,16> EQL RMSERR(RNF) THEN STATUS = RMSERR(EOF))) %;
;  230          0294  1
```

```
;  232     0295  1 %SBTTL 'RM$CHECK_RRV'
;  233     0296  1 ROUTINE RM$CHECK_RRV : RL$LINKAGE =
;  234     0297  1 !++
;  235     0298  1 !
;  236     0299  1 ! FUNCTIONAL DESCRIPTION:
;  237     0300  1 !
;  238     0301  1 !       This routine compares the RRV pointer of the record RMS is currently
;  239     0302  1 !       positioned to with the RFA address of the current record, and returns
;  240     0303  1 !       the result of that match verifying whether or not the two are one and
;  241     0304  1 !       the same - ie the record RMS is currently positioned to is the current
;  242     0305  1 !       record.
;  243     0306  1 !
;  244     0307  1 ! CALLING SEQUENCE:
;  245     0308  1 !
;  246     0309  1 !       BSBW RM$CHECK_RRV()
;  247     0310  1 !
;  248     0311  1 ! INPUT PARAMETERS:
;  249     0312  1 !       NONE
;  250     0313  1 !
;  251     0314  1 ! IMPLICIT INPUTS:
;  252     0315  1 !
;  253     0316  1 !       IRAB                    - address of the IRAB
;  254     0317  1 !           IRB$W_POS_ID        - ID of user data record for NRP positioning
;  255     0318  1 !           IRB$L_POS_VBN       - VBN of user data record for NRP positioning
;  256     0319  1 !
;  257     0320  1 !       REC_ADDR                - address of primary data record
;  258     0321  1 !
;  259     0322  1 ! OUTPUT PARAMETERS:
;  260     0323  1 !       NONE
;  261     0324  1 !
;  262     0325  1 ! IMPLICIT OUTPUTS:
;  263     0326  1 !       NONE
;  264     0327  1 !
;  265     0328  1 ! ROUTINE VALUE:
;  266     0329  1 !
;  267     0330  1 !       SUC - the RRV of the current primary data record matches the current
;  268     0331  1 !             records RFA address.
;  269     0332  1 !
;  270     0333  1 !       0   - the RRV of the current primary data record does not match the
;  271     0334  1 !             current records RFA address.
;  272     0335  1 !
;  273     0336  1 ! SIDE EFFECTS:
;  274     0337  1 !
;  275     0338  1 !       AP is trashed.
;  276     0339  1 !
;  277     0340  1 !--
;  278     0341  1
;  279     0342  2     BEGIN
;  280     0343  2
;  281     0344  2     BUILTIN
;  282     0345  2         AP;
;  283     0346  2
;  284     0347  2     EXTERNAL REGISTER
;  285     0348  2         COMMON_RAB_STR,
;  286     0349  2         R_IDX_DFN_STR,
;  287     0350  2         R_REC_ADDR;
;  288     0351  2
```

```
;  289          0352  2     GLOBAL REGISTER
;  290          0353  2         R_BDB;
;  291          0354  2
;  292          0355  2     AP = 3;
;  293          0356  2
;  294          0357  2     ! If the two addresses match, return success. The record RMS is positioned
;  295          0358  2     ! to is in fact the current record.
;  296          0359  2     !
;  297          0360  3     IF  (.IRAB[IRB$W_POS_ID] EQL RM$RECORD_ID())
;  298          0361  2          AND
;  299          0362  3         (.IRAB[IRB$L_POS_VBN] EQL RM$RECORD_VBN())
;  300          0363  2     THEN
;  301          0364  3         RETURN RMSSUC()
;  302          0365  3
;  303          0366  3     ! Otherwise, the record RMS is currently positioned to is not the current
;  304          0367  3     ! record
;  305          0368  3     !
;  306          0369  2     ELSE
;  307          0370  2         RETURN 0;
;  308          0371  2
;  309          0372  1     END;                                    ! {  of routine }

                                                .TITLE  RM3POSSEQ
                                                .IDENT  \V04-000\

                                                .EXTRN  RM$COMPARE_REC, RM$CSEARCH_TREE
                                                .EXTRN  RM$FIND_BY_RFA, RM$FIND_BY_RRV
                                                .EXTRN  RM$GETBKT, RM$GETNEXT_REC
                                                .EXTRN  RM$RECORD_ID, RM$RECORD_VBN
                                                .EXTRN  RM$REC_OVHD, RM$RLSBKT
                                                .EXTRN  RM$RU_RECLAIM, RM$SEARCH_SIDR
                                                .EXTRN  RM$SIDR_FIRST, RM$SRCH_BY_KEY

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

                                  54  DD 00000 RM$CHECK_RRV:
                                                PUSHL   R4                              ; 0296
                              5C          03  D0 00002          MOVL    #3, AP          ; 0355
                                      0000G 30 00005          BSBW    RM$RECORD_ID    ; 0360
                50      00BA  C9      10      00  ED 00008          CMPZV   #0, #16, T86(IRAB), R0  ;
                                              0F  12 0000F          BNEQ    1$              ;
                                      0000G 30 00011          BSBW    RM$RECORD_VBN   ; 0362
                              50      00AC    C9  D1 00014          CMPL    172(IRAB), R0   ;
                                              05  12 00019          BNEQ    1$              ;
                              50              01  D0 0001B          MOVL    #1, R0          ; 0370
                                              02  11 0001E          BRB     2$              ;
                                              50  D4 00020 1$:      CLRL    R0              ;
                                              10  BA 00022 2$:      POPR    #^M<R4>         ; 0372
                                                  05 00024          RSB                     ;

; Routine Size:  37 bytes,    Routine Base:  RM$RMS3 + 0000
```

```
;  311     0373  1 %SBTTL 'RM$FIND_NONDEL'
;  312     0374  1 ROUTINE RM$FIND_NONDEL (FLAG) : RL$LINKAGE =
;  313     0375  1 !++
;  314     0376  1 !
;  315     0377  1 ! FUNCTIONAL DESCRIPTION:
;  316     0378  1 !
;  317     0379  1 !       This routine searches for the next next non-deleted primary data record
;  318     0380  1 !       in the file. If the input parameter is 1, the search starts with the
;  319     0381  1 !       record RMS is currently positioned to, but if its 0, the search will
;  320     0382  1 !       start with the very next bucket.
;  321     0383  1 !
;  322     0384  1 !       If RMS encounters a record that is marked RU_DELETE and the Recovery
;  323     0385  1 !       Unit in which the record was deleted it still active, then RMS returns
;  324     0386  1 !       this record as the next non-deleted primary data record and lets a
;  325     0387  1 !       higher level routine decide what to do. If the Recovery Unit in which
;  326     0388  1 !       the record was deleted has successfully terminated, then RMS will
;  327     0389  1 !       continue its search for a non-deleted primary data record after deleting
;  328     0390  1 !       this RU_DELETEd record (the latter if it has write access to the file).
;  329     0391  1 !
;  330     0392  1 !       If RMS encounters a record that is marked RU_UPDATE and is in a special
;  331     0393  1 !       format then RMS will return this record as the next non-deleted primary
;  332     0394  1 !       data record after reformatting it. The reformatting is done if RMS has
;  333     0395  1 !       write access to the file, and the Recovery Unit in which it was updated
;  334     0396  1 !       has successfully terminated.
;  335     0397  1 !
;  336     0398  1 ! CALLING SEQUENCE:
;  337     0399  1 !
;  338     0400  1 !       BSBW RM$FIND_NONDEL()
;  339     0401  1 !
;  340     0402  1 ! INPUT PARAMETERS:
;  341     0403  1 !
;  342     0404  1 !       FLAG            - if 1, start search with the record RMS is currently
;  343     0405  1 !                         positioned to
;  344     0406  1 !
;  345     0407  1 ! IMPLICIT INPUTS:
;  346     0408  1 !
;  347     0409  1 !       IDX_DFN                 - address of index descriptor
;  348     0410  1 !           IDX$B_DATBKTSZ      - data bucket size
;  349     0411  1 !           IDX$V_KEY_COMPR     - if set, primary key compression is enabled
;  350     0412  1 !
;  351     0413  1 !       IFAB                    - address of IFAB
;  352     0414  1 !           IFB$V_RU            - if set, file is Recovery Unit Journallable
;  353     0415  1 !           IFB$V_WRTACC        - if set, stream has write access to the file
;  354     0416  1 !
;  355     0417  1 !       IRAB                    - address of IRAB
;  356     0418  1 !           IRB$L_CURBDB        - address of BDB for accessed data bucket
;  357     0419  1 !
;  358     0420  1 !       REC_ADDR                - address of record RMS is positioned to
;  359     0421  1 !
;  360     0422  1 ! OUTPUT PARAMETERS:
;  361     0423  1 !       NONE
;  362     0424  1 !
;  363     0425  1 ! IMPLICIT OUTPUTS:
;  364     0426  1 !
;  365     0427  1 !       IRB$L_CURBDB    - address of BDB describing primary data bucket
;  366     0428  1 !       IRB$V_DEL_SEEN  - set if deleted records encountered
;  367     0429  1 !       IRB$L_LST_NCMP  - updated to last record with a 0 front compressed key
```

```
;  368          0430  1 !       REC_ADDR        - address of next non-deleted primary data record
;  369          0431  1 !
;  370          0432  1 ! ROUTINE VALUE:
;  371          0433  1 !
;  372          0434  1 !       SUC     - found next non-deleted primary data record.
;  373          0435  1 !       EOF     - end-of-file encountered before record found.
;  374          0436  1 !       various I/O errors
;  375          0437  1 !
;  376          0438  1 ! SIDE EFFECTS:
;  377          0439  1 !
;  378          0440  1 !       SUC - REC_ADDR points to next non-deleted primary data record, and
;  379          0441  1 !             IRB$L_CURBDB contains the address of the BDB describing the
;  380          0442  1 !             primary data bucket.
;  381          0443  1 !       On all errors, no primary data bucket is accessed.
;  382          0444  1 !       If deleted reords are encountered the flag IRB$V_DEL_SEEN is set.
;  383          0445  1 !       IF RU_DELETEd records are encountered, they might have been deleted.
;  384          0446  1 !       If RU_UPDATEd records are encountered, they might have been reformatted.
;  385          0447  1 !
;  386          0448  1 !--
;  387          0449  1
;  388          0450  2     BEGIN
;  389          0451  2
;  390          0452  2     BUILTIN
;  391          0453  2         AP;
;  392          0454  2
;  393          0455  2     EXTERNAL REGISTER
;  394          0456  2         COMMON_RAB_STR,
;  395          0457  2         R_IDX_DFN_STR,
;  396          0458  2         R_REC_ADDR_STR;
;  397          0459  2
;  398          0460  2     GLOBAL REGISTER
;  399          0461  2         COMMON_IO_STR;
;  400          0462  2
;  401          0463  2     LOCAL
;  402          0464  2         VBN;
;  403          0465  2
;  404          0466  2     ! Retrieve the address of the start of the primary data bucket.
;  405          0467  2     !
;  406          0468  2     BDB = .IRAB[IRB$L_CURBDB];
;  407          0469  2     BKT_ADDR = .BDB[BDB$L_ADDR];
;  408          0470  2
;  409          0471  2     ! If RMS is to start the positioning to the next non-deleted primary data
;  410          0472  2     ! record with the next record, and not the record it is currently
;  411          0473  2     ! positioned to, position to the next record.
;  412          0474  2     !
;  413          0475  2     IF NOT .FLAG
;  414          0476  2     THEN
;  415          0477  3         BEGIN
;  416          0478  3
;  417          0479  3         ! If primary key compression is enabled and the key of the record is
;  418          0480  3         ! zero-front compressed, then save the address of the record as the last
;  419          0481  3         ! record encountered with a zero-front compressed key.
;  420          0482  3         !
;  421          0483  3         IF .IDX_DFN[IDX$V_KEY_COMPR]
;  422          0484  3             AND
;  423          0485  3             .(.REC_ADDR + RM$REC_OVHD(0))<8,8> EQLU 0
;  424          0486  3         THEN
```

```
;  425     0487  3              IRAB[IRB$L_LST_NCMP] = .REC_ADDR;
;  426     0488  3
;  427     0489  3          RM$GETNEXT_REC();
;  428     0490  2          END;
;  429     0491  2
;  430     0492  2      ! Continue to look for the next non-deleted record until either one is
;  431     0493  2      ! found, the end-of-file is encountered or some random I/O error occurs.
;  432     0494  2      !
;  433     0495  2      WHILE 1
;  434     0496  2      DO
;  435     0497  3          BEGIN
;  436     0498  3
;  437     0499  3          ! Continue the search for a non-deleted primary data record in the
;  438     0500  3          ! current bucket until either all the record are exhausted, or the
;  439     0501  3          ! first RRV is encountered.
;  440     0502  3          !
;  441     0503  4          WHILE .REC_ADDR LSSA (.BKT_ADDR + .BKT_ADDR[BKT$W_FREESPACE])
;  442     0504  3                AND
;  443     0505  3               NOT .REC_ADDR[IRC$V_RRV]
;  444     0506  3          DO
;  445     0507  3
;  446     0508  3              ! If RMS finds that the current record has been deleted within a
;  447     0509  3              ! Recovery Unit, then it subjects this record to further processing
;  448     0510  3              ! before deciding whether to return this record as the next
;  449     0511  3              ! non-deleted primary data record, or to continue with the search.
;  450     0512  3              !
;  451     0513  3              IF  .REC_ADDR[IRC$V_RU_DELETE]
;  452     0514  3              THEN
;  453     0515  4                  BEGIN
;  454     0516  4
;  455     0517  4                  LOCAL
;  456     0518  4                      RECORD_ID : WORD,
;  457     0519  4                      STATUS;
;  458     0520  4
;  459     0521  4                  RECORD_ID = .REC_ADDR[IRC$W_ID];
;  460     0522  4
;  461     0523  4                  ! If the Recovery Unit in which the record was deleted is
;  462     0524  4                  ! still active, or the file is not open for write access then
;  463     0525  4                  ! RMS will not be able to recover any space from the current
;  464     0526  4                  ! record; however, RMS may still want to return this record as
;  465     0527  4                  ! the next non-deleted primary data record.
;  466     0528  4                  !
;  467     0529  5                  IF  NOT (STATUS = RM$RU_RECLAIM())
;  468     0530  4                  THEN
;  469     0531  4
;  470     0532  4                      ! RMS will return the current record as the next non-deleted
;  471     0533  4                      ! primary data record if this record was deleted by another
;  472     0534  4                      ! process whose Recovery Unit has not yet completed.
;  473     0535  4                      !
;  474     0536  5                      IF .STATUS<0,16> EQLU RMSERR(RLK)
;  475     0537  4                      THEN
;  476     0538  5                          RETURN RMSSUC()
;  477     0539  5
;  478     0540  5                      ! If the current record was deleted in a Recovery Unit by
;  479     0541  5                      ! the current process and this process is still within a
;  480     0542  5                      ! Recovery Unit; or, if RMS is able to lock this record but
;  481     0543  5                      ! has not write accessed the file, then RMS treats the
```

```
;  482     0544  5                    ! current record as if it actually is deleted, and
;  483     0545  5                    ! positions to the next record so as to continue the search.
;  484     0546  5                    !
;  485     0547  4                    ELSE
;  486     0548  5                        BEGIN
;  487     0549  5
;  488     0550  5                        ! If index compression is enabled and the key of the
;  489     0551  5                        ! record is zero-front compressed, then save the
;  490     0552  5                        ! address of the record as the last record encountered
;  491     0553  5                        ! with a zero-front compressed key.
;  492     0554  5                        !
;  493     0555  5                        IF .IDX_DFN[IDX$V_KEY_COMPR]
;  494     0556  5                            AND
;  495     0557  5                            .(.REC_ADDR + RM$REC_OVHD(0))<8,8> EQLU 0
;  496     0558  5                        THEN
;  497     0559  5                            IRAB[IRB$L_LST_NCMP] = .REC_ADDR;
;  498     0560  5
;  499     0561  5                        IRAB[IRB$V_DEL_SEEN] = 1;
;  500     0562  5                        RM$GETNEXT_REC();
;  501     0563  5                        END
;  502     0564  5
;  503     0565  5                ! RMS was able to reclaim some space from the current
;  504     0566  5                ! record because the Recovery Unit in which the record had
;  505     0567  5                ! been deleted has successfully terminated. RMS will
;  506     0568  5                ! continue the search for a non-deleted primary data record
;  507     0569  5                ! with the next record in the file.
;  508     0570  5                !
;  509     0571  4                ELSE
;  510     0572  4                    IF .RECORD_ID NEQU .REC_ADDR[IRC$W_ID]
;  511     0573  4                    THEN
;  512     0574  4                        IRAB[IRB$V_DEL_SEEN] = 1;
;  513     0575  4                    END
;  514     0576  4
;  515     0577  4                ! If the record RMS has positioned to is not deleted, then RMS has
;  516     0578  4                ! found its next non-deleted primary data record, and can return
;  517     0579  4                ! success. Otherwise, set the deleted record seen bit, and position
;  518     0580  4                ! to the very next record in the primary data bucket.
;  519     0581  4                !
;  520     0582  3                ELSE
;  521     0583  3                    IF NOT .REC_ADDR[IRC$V_DELETED]
;  522     0584  3                    THEN
;  523     0585  4                        BEGIN
;  524     0586  4
;  525     0587  4                        ! If the current non-deleted primary data record has been
;  526     0588  4                        ! updated within a Recovery Unit then it maybe re-formated
;  527     0589  4                        ! before it is returned provided the file has been open
;  528     0590  4                        ! for write access, and the Recovery Unit in which it was
;  529     0591  4                        ! eas updated has completed successfully.
;  530     0592  4                        !
;  531     0593  4                        IF  .REC_ADDR[IRC$V_RU_UPDATE]
;  532     0594  4                            AND
;  533     0595  4                            .IFAB[IFB$V_WRTACC]
;  534     0596  4                        THEN
;  535     0597  4                            RM$RU_RECLAIM();
;  536     0598  4
;  537     0599  4                        RETURN RMSSUC();
;  538     0600  4                        END
```

```
;  539     0601  3                ELSE
;  540     0602  4                    BEGIN
;  541     0603  4
;  542     0604  4                    ! If index compression is enabled and the key of the record
;  543     0605  4                    ! is zero-front compressed, then save the address of the
;  544     0606  4                    ! record as the last record encountered with a zero-front
;  545     0607  4                    ! compressed key.
;  546     0608  4                    !
;  547     0609  4                    IF  .IDX_DFN[IDX$V_KEY_COMPR]
;  548     0610  4                        AND
;  549     0611  4                        .(.REC_ADDR + RM$REC_OVHD(0))<8,8> EQLU 0
;  550     0612  4                    THEN
;  551     0613  4                        IRAB[IRB$L_LST_NCMP] = .REC_ADDR;
;  552     0614  4
;  553     0615  4                    IRAB[IRB$V_DEL_SEEN] = 1;
;  554     0616  4                    RM$GETNEXT_REC();
;  555     0617  3                    END;
;  556     0618  3
;  557     0619  3            ! If all records in the current record have been exhausted without
;  558     0620  3            ! find a non-deleted primary data record, and this is the very last
;  559     0621  3            ! bucket in the horizontal chain, then return an error of end-of-file.
;  560     0622  3            !
;  561     0623  3            IF .BKT_ADDR[BKT$V_LASTBKT]
;  562     0624  3            THEN
;  563     0625  4                BEGIN
;  564     0626  4                RELEASE (IRAB[IRB$L_CURBDB]);
;  565     0627  4                RETURN RMSERR(EOF);
;  566     0628  3                END;
;  567     0629  3
;  568     0630  3            ! If this is not the last bucket in the horizontal chain, get the
;  569     0631  3            ! very next bucket and continue the search for the next non-deleted
;  570     0632  3            ! primary data bucket in it.
;  571     0633  3            !
;  572     0634  3            VBN = .BKT_ADDR[BKT$L_NXTBKT];
;  573     0635  3            RELEASE (IRAB[IRB$L_CURBDB]);
;  574     0636  3
;  575     0637  3            ! If the file is Recovery Unit Journallable and write accessed, then
;  576     0638  3            ! make sure the next primary data bucket in the horizontal chain is
;  577     0639  3            ! exclusively accessed.
;  578     0640  3            !
;  579     0641  3            IF .IFAB[IFB$V_WRTACC]
;  580     0642  3                AND
;  581     0643  3                .IFAB[IFB$V_RU]
;  582     0644  3            THEN
;  583     0645  3                IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  584     0646  3
;  585     0647  3            RETURN_ON_ERROR (RM$GETBKT (.VBN, .IDX_DFN[IDX$B_DATBKTSZ] * 512));
;  586     0648  3
;  587     0649  3            IRAB[IRB$L_CURBDB] = .BDB;
;  588     0650  3
;  589     0651  3            REC_ADDR = .BKT_ADDR + BKT$C_OVERHDSZ;
;  590     0652  3
;  591     0653  3            ! Make sure the address of the first record in this bucket is placed
;  592     0654  3            ! in IRB$L_LST_NCMP if primary key compression is enabled.
;  593     0655  3            !
;  594     0656  3            IF .IDX_DFN[IDX$V_KEY_COMPR]
;  595     0657  3            THEN
```

```
;  596          0658  3          IRAB[IRB$L_LST_NCMP] = .REC_ADDR;
;  597          0659  2      END;
;  598          0660  2
;  599          0661  1  END;                                   ! {  of routine }
; INFO#212              L1:0495
; Null expression appears in value-required context


                                    3C  BB 00000 RM$FIND_NONDEL:
                                                 PUSHR   #^M<R2,R3,R4,R5>                ; 0374
                              54    20  A9 D0 00002            MOVL    32(IRAB), BDB      ; 0468
                              55    18  A4 D0 00006            MOVL    24(BDB), BKT_ADDR  ; 0469
                              18    14  AE E8 0000A            BLBS    FLAG, 2$           ; 0475
                    10        1C  A7    06 E1 0000E            BBC     #6, 28(IDX_DFN), 1$ ; 0483
                                        51 D4 00013            CLRL    R1                 ; 0485
                                  0000G 30 00015            BSBW    RM$REC_OVHD
                                     01 A046 95 00018            TSTB    1(R0)[REC_ADDR]
                                        05 12 0001C            BNEQ    1$
                              0098  C9    56 D0 0001E            MOVL    REC_ADDR, 152(IRAB) ; 0487
                                  0000G 30 00023 1$:          BSBW    RM$GETNEXT_REC     ; 0489
                              50    04  A5 3C 00026 2$:          MOVZWL  4(BKT_ADDR), R0    ; 0503
                              50        55 C0 0002A            ADDL2   BKT_ADDR, R0
                              50        56 D1 0002D            CMPL    REC_ADDR, R0
                                        6C 1E 00030            BGEQU   9$
                    68        66        03 E0 00032            BBS     #3, (REC_ADDR), 9$ ; 0505
                    3B        66        05 E1 00036            BBC     #5, (REC_ADDR), 6$ ; 0513
                              52    01  A6 B0 0003A            MOVW    1(REC_ADDR), RECORD_ID ; 0521
                                  00000000G EF 16 0003E            JSB     RM$RU_RECLAIM      ; 0529
                              22        50 E8 00044            BLBS    STATUS, 5$
                          82AA  8F        50 B1 00047            CMPW    STATUS, #33450     ; 0536
                                        39 13 0004C            BEQL    7$
                    10        1C  A7    06 E1 0004E            BBC     #6, 28(IDX_DFN), 4$ ; 0555
                                        51 D4 00053            CLRL    R1                 ; 0557
                                  0000G 30 00055            BSBW    RM$REC_OVHD
                                     01 A046 95 00058            TSTB    1(R0)[REC_ADDR]
                                        05 12 0005C            BNEQ    4$
                          0098  C9        56 D0 0005E 3$:          MOVL    REC_ADDR, 152(IRAB) ; 0559
                            43  A9        02 88 00063 4$:          BISB2   #2, 67(IRAB)       ; 0561
                                        BA 11 00067            BRB     1$                 ; 0562
                          01    A6        52 B1 00069 5$:          CMPW    RECORD_ID, 1(REC_ADDR) ; 0572
                                        B7 13 0006D            BEQL    2$
                            43  A9        02 88 0006F            BISB2   #2, 67(IRAB)       ; 0574
                                        B1 11 00073            BRB     2$                 ; 0513
                    13        66        02 E0 00075 6$:          BBS     #2, (REC_ADDR), 8$ ; 0583
                    0A        66        06 E1 00079            BBC     #6, (REC_ADDR), 7$ ; 0593
                              06    06  AA E9 0007D            BLBC    6(IFAB), 7$        ; 0595
                                  00000000G EF 16 00081            JSB     RM$RU_RECLAIM      ; 0597
                              50        01 D0 00087 7$:          MOVL    #1, R0             ; 0599
                                        75 11 0008A            BRB     13$
                    D2        1C  A7    06 E1 0008C 8$:          BBC     #6, 28(IDX_DFN), 4$ ; 0609
                                        51 D4 00091            CLRL    R1                 ; 0611
                                  0000G 30 00093            BSBW    RM$REC_OVHD
                                     01 A046 95 00096            TSTB    1(R0)[REC_ADDR]
                                        C2 13 0009A            BEQL    3$
```

```
                                    C5  11 0009C          BRB     4$                                      ; 0615
                              16    0D  A5  E9 0009E 9$:  BLBC    13(BKT_ADDR), 10$                       ; 0623
                              54    20  A9  D0 000A2       MOVL    32(IRAB), BDB                           ; 0626
                                    20  A9  D4 000A6       CLRL    32(IRAB)                                ;
                                        7E  D4 000A9       CLRL    -(SP)                                   ;
                                  0000G 30 000AB          BSBW    RM$RLSBKT                               ;
                              5E        04  C0 000AE       ADDL2   #4, SP                                  ;
                              50  827A  8F  3C 000B1       MOVZWL  #33402, R0                              ; 0627
                                        49  11 000B6       BRB     13$                                     ;
                              53    08  A5  D0 000B8 10$:  MOVL    8(BKT_ADDR), VBN                        ; 0634
                              54    20  A9  D0 000BC       MOVL    32(IRAB), BDB                           ; 0635
                                    20  A9  D4 000C0       CLRL    32(IRAB)                                ;
                                        7E  D4 000C3       CLRL    -(SP)                                   ;
                                  0000G 30 000C5          BSBW    RM$RLSBKT                               ;
                              5E        04  C0 000C8       ADDL2   #4, SP                                  ;
                              0A    06  AA  E9 000CB       BLBC    6(IFAB), 11$                            ; 0641
                        04    00A0  CA      01  E1 000CF  BBC     #1, 160(IFAB), 11$                      ; 0643
                              40    A9      01  90 000D5  MOVB    #1, 64(IRAB)                            ; 0645
                              50    17  A7  9A 000D9 11$:  MOVZBL  23(IDX_DFN), R0                         ; 0647
                        7E          50      09  78 000DD  ASHL    #9, R0, -(SP)                           ;
                                            53  DD 000E1  PUSHL   VBN                                     ;
                                      0000G 30 000E3       BSBW    RM$GETBKT                               ;
                              5E            08  C0 000E6  ADDL2   #8, SP                                  ;
                              15            50  E9 000E9  BLBC    STATUS, 13$                             ;
                                    20  A9      54  D0 000EC  MOVL    BDB, 32(IRAB)                       ; 0649
                              56        0E  A5  9E 000F0  MOVAB   14(R5), REC_ADDR                        ; 0651
                        05    1C  A7          06  E1 000F4  BBC     #6, 28(IDX_DFN), 12$                    ; 0656
                            0098  C9          56  D0 000F9  MOVL    REC_ADDR, 152(IRAB)                     ; 0658
                                        FF25  31 000FE 12$:  BRW     2$                                      ; 0495
                                          3C  BA 00101 13$:  POPR    #^M<R2,R3,R4,R5>                        ; 0661
                                              05 00103       RSB                                           ;

; Routine Size:  260 bytes,    Routine Base:  RM$RMS3 + 0025
```

```
;  601          0662  1 %SBTTL 'RM$FIND_SIDR'
;  602          0663  1 ROUTINE RM$FIND_SIDR : RL$LINKAGE =
;  603          0664  1 !++
;  604          0665  1 !
;  605          0666  1 ! FUNCTIONAL DESCRIPTION:
;  606          0667  1 !
;  607          0668  1 !       The function of this routine is to position to a particular SIDR.
;  608          0669  1 !
;  609          0670  1 !       If the NRP key of reference coincides with the key of reference of the
;  610          0671  1 !       current index descriptor, then this SIDR is the current record. In such
;  611          0672  1 !       a case the NRP information is used to attempt to position to the
;  612          0673  1 !       current SIDR, and if that fails an attempt is made to position to the
;  613          0674  1 !       current SIDR by means of a random search of the alternate index. If RMS
;  614          0675  1 !       is able to locate the current record, the number of array elements
;  615          0676  1 !       preceeding the current element is placed in IRB$W_SAVE_POS, and success
;  616          0677  1 !       is returned. If RMS is unable to position to the current record because
;  617          0678  1 !       it has been deleted, then IRB$W_SAVE_POS is zeroed, and an RMS error
;  618          0679  1 !       of record deleted is returned. This status may not represent an actual
;  619          0680  1 !       error, and so the SIDR positioned to is also returned. If RMS is unable
;  620          0681  1 !       to locate any record, than an error of RNF is returned.
;  621          0682  1 !
;  622          0683  1 !       If the NRP key of reference is not equal to the key of reference of the
;  623          0684  1 !       current index descriptor, then RMS positions to the first SIDR with a
;  624          0685  1 !       key greater than or equal to to the key found in keybuffer 2, or to the
;  625          0686  1 !       first SIDR with a key equal to the key in keybuffer 2, depending upon
;  626          0687  1 !       the IRB$B_SRCHKEY bit settings. If RMS is successful at positioning to
;  627          0688  1 !       a SIDR, IRB$W_SAVE_POS will always be zero and a status of success is
;  628          0689  1 !       returned; otherwise, a RMS error of record not found is returned.
;  629          0690  1 !
;  630          0691  1 ! CALLING SEQUENCE:
;  631          0692  1 !
;  632          0693  1 !       BSBW RM$FIND_SIDR()
;  633          0694  1 !
;  634          0695  1 ! INPUT PARAMETERS:
;  635          0696  1 !       NONE
;  636          0697  1 !
;  637          0698  1 ! IMPLICIT INPUTS:
;  638          0699  1 !
;  639          0700  1 !       IDX_DFN                 - index descriptor of index to search
;  640          0701  1 !           IDX$B_DATBKTSZ      - size of SIDR bucket in VBNs
;  641          0702  1 !           IDX$B_KEYREF        - key of reference of index descriptor
;  642          0703  1 !           IDX$B_KEYSZ         - size of SIDR key
;  643          0704  1 !
;  644          0705  1 !       IFAB                    - address of IFAB
;  645          0706  1 !           IFB$W_KBUFSZ        - size of a keybuffer
;  646          0707  1 !           IFB$V_RU            - if set, file is Recovery Unit Journallable
;  647          0708  1 !           IFB$V_WRTACC        - if set, stream has write access to the file
;  648          0709  1 !
;  649          0710  1 !       IRAB                    - address of IRAB
;  650          0711  1 !           IRB$L_CURBDB        - must be zero
;  651          0712  1 !           IRB$W_CUR_COUNT     - number of elements preceeding current element
;  652          0713  1 !           IRB$B_CUR_KREF      - key of reference of current SIDR record
;  653          0714  1 !           IRB$L_KEYBUF        - address of stream's five keybuffers
;  654          0715  1 !           IRB$W_SIDR_ID       - ID of current SIDR's first array element
;  655          0716  1 !           IRB$L_SIDR_VBN      - VBN of current SIDR's first array element
;  656          0717  1 !           IRB$V_SRCHGE        - if set, search for GE match
;  657          0718  1 !           IRB$V_SRCHGT        - if set, search for GT match
```

```
;  658     0719  1 !           IRB$B_STOPLEVEL     - level at which to stop search (must be 0)
;  659     0720  1 !
;  660     0721  1 ! OUTPUT PARAMETERS:
;  661     0722  1 !       NONE
;  662     0723  1 !
;  663     0724  1 ! IMPLICIT OUTPUTS:
;  664     0725  1 !
;  665     0726  1 !       IRB$L_CURBDB    - address of SIDR bucket's BDB
;  666     0727  1 !       IRB$W_SAVE_POS  - number of elements preceeding the current one
;  667     0728  1 !       REC_ADDR        - address of SIDR record
;  668     0729  1 !
;  669     0730  1 ! ROUTINE VALUE:
;  670     0731  1 !
;  671     0732  1 !       DEL     - if RMS was to position to the current SIDR and found it
;  672     0733  1 !                 deleted.
;  673     0734  1 !       RNF     - if RMS was unable to find any SIDR to position to and EOF
;  674     0735  1 !                 was not encountered.
;  675     0736  1 !       SUC     - if the desired SIDR was positioned to.
;  676     0737  1 !       various I/O errors.
;  677     0738  1 !
;  678     0739  1 ! SIDE EFFECTS:
;  679     0740  1 !
;  680     0741  1 !       If RMS is successful at positioning to the current SIDR by means of the
;  681     0742  1 !           NRP information, IRB$W_SAVE_POS, REC_ADDR, and IRB$L_CURBDB all
;  682     0743  1 !           have their stated values and a status of SUC is returned.
;  683     0744  1 !       If RMS is unable to successfully position to the current SIDR by means
;  684     0745  1 !           of the NRP information, but is able to position to a record, then
;  685     0746  1 !           REC_ADDR and IRB$L_CURBDB have their assigned values,
;  686     0747  1 !           IRB$W_SAVE_POS is set to 0, and a status of DEL is returned.
;  687     0748  1 !       If RMS is not positioning to the current SIDR and is able to position
;  688     0749  1 !           to some record, then IRB$W_SAVE_POS is set to 0, REC_ADDR and
;  689     0750  1 !           IRB$L_CURBDB have their assigned values, and a status of SUC is
;  690     0751  1 !           returned.
;  691     0752  1 !       If RMS is not positioning to the current SIDR and is unable to position
;  692     0753  1 !           to some record, then IRB$W_SAVE_POS is set to 0, and a status of
;  693     0754  1 !           RNF is returned.
;  694     0755  1 !--
;  695     0756  1
;  696     0757  2     BEGIN
;  697     0758  2
;  698     0759  2     EXTERNAL REGISTER
;  699     0760  2         COMMON_RAB_STR,
;  700     0761  2         R_IDX_DFN_STR,
;  701     0762  2         R_REC_ADDR;
;  702     0763  2
;  703     0764  2     LOCAL
;  704     0765  2         FLAGS   : BLOCK[1];
;  705     0766  2
;  706     0767  2     MACRO
;  707     0768  2         FIRST_TIME = 0,0,1,0 %,
;  708     0769  2         NRP        = 0,1,1,0 %;
;  709     0770  2
;  710     0771  2     ! Initialize some variables.
;  711     0772  2     !
;  712     0773  2     FLAGS = 0;
;  713     0774  2     FLAGS[FIRST_TIME] = 1;
;  714     0775  2
```

```
;  715          0776  2      IRAB[IRB$W_SAVE_POS] = 0;
;  716          0777  2
;  717          0778  2      ! If the index descriptor is for the current key of reference, then the
;  718          0779  2      ! NRP information will be used in the positioning.
;  719          0780  2      !
;  720          0781  3      IF (.IDX_DFN[IDX$B_KEYREF] EQLU .IRAB[IRB$B_CUR_KREF])
;  721          0782  2      THEN
;  722          0783  2          FLAGS[NRP] = 1;
;  723          0784  2
;  724          0785  2      ! We may have to make two circuits through roughly the same code;
;  725          0786  2      ! therefore, setup the circumstances for doing so - ie simulate a GOTO.
;  726          0787  2      ! This loop can only be exited by returning to the calling routine.
;  727          0788  2      !
;  728          0789  2      WHILE 1
;  729          0790  2      DO
;  730          0791  3          BEGIN
;  731          0792  3
;  732          0793  3          GLOBAL REGISTER
;  733          0794  3              COMMON_IO_STR;
;  734          0795  3
;  735          0796  3          LOCAL
;  736          0797  3              SEARCH_RESULT;
;  737          0798  3
;  738          0799  3          ! If the next record positioning formation is to be used, and this is
;  739          0800  3          ! the first time through the loop, then use the next record context in
;  740          0801  3          ! the attempt to positioning to the desired SIDR.
;  741          0802  3          !
;  742          0803  3          IF  .FLAGS[NRP] AND .FLAGS[FIRST_TIME]
;  743          0804  3          THEN
;  744          0805  4              BEGIN
;  745          0806  4
;  746          0807  4              ! All SIDR buckets are exclusively accessed in case space
;  747          0808  4              ! reclamation will be required.
;  748          0809  4              !
;  749          0810  4              IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  750          0811  4
;  751          0812  4              ! First, access the bucket hopefully containing the current record
;  752          0813  4              ! using the next record context information.
;  753          0814  4              !
;  754        P 0815  4              RETURN_ON_ERROR (RM$GETBKT (.IRAB[IRB$L_CUR_VBN],
;  755          0816  4                                          .IDX_DFN[IDX$B_DATBKTSZ]*512));
;  756          0817  4              IRAB[IRB$L_CURBDB] = .BDB;
;  757          0818  4
;  758          0819  4              ! Next, do a search through this bucket looking for a SIDR with a
;  759          0820  4              ! key value the same as that of the current record, the key of
;  760          0821  4              ! which maybe found in keybuffer 2.
;  761          0822  4              !
;  762          0823  4              BKT_ADDR = .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$L_ADDR];
;  763          0824  4              REC_ADDR = .BKT_ADDR + BKT$C_OVERHDSZ;
;  764          0825  4
;  765          0826  4              SEARCH_RESULT = RM$SRCH_BY_KEY();
;  766          0827  4              END
;  767          0828  4
;  768          0829  4          ! If the next record positioning information can not be used to
;  769          0830  4          ! position to the desired SIDR either because it is for the wrong key
;  770          0831  4          ! of reference, or because this is not the first time through this
;  771          0832  4          ! loop, then perform a random search of the alternate index using as
```

```
;  772     0833  4          ! the search key the key in keybuffer 2.
;  773     0834  4          !
;  774     0835  3          ELSE
;  775     0836  4              BEGIN
;  776     0837  4
;  777     0838  4              RETURN_ON_ERROR (RM$CSEARCH_TREE());
;  778     0839  4
;  779     0840  4              ! Determine whether the SIDR positioned to has a key greater
;  780     0841  4              ! than or equal to or less than the key in keybuffer 2.
;  781     0842  4              !
;  782     0843  4              SEARCH_RESULT = RM$COMPARE_REC (KEYBUF_ADDR(2),
;  783     0844  4                                              .IDX_DFN[IDX$B_KEYSZ],
;  784     0845  4                                              0);
;  785     0846  3              END;
;  786     0847  3
;  787     0848  3          ! The SEARCH_RESULT can be greater-than only if RMS is currently
;  788     0849  3          ! positioning using the NRP context information, and the key in
;  789     0850  3          ! keybuffer 2 is greater than the key of every SIDR in the current
;  790     0851  3          ! bucket. In such a situation, RMS knows that the SIDR it was looking
;  791     0852  3          ! for has moved since it was accessed by this process. Therefore, this
;  792     0853  3          ! SIDR could not have been in a continuation bucket, and if its still
;  793     0854  3          ! in the ISAM file it can be positioned to by performing a random
;  794     0855  3          ! search of the alternate key index using the record's key value after
;  795     0856  3          ! releasing the current SIDR bucket.
;  796     0857  3          !
;  797     0858  3          IF .SEARCH_RESULT GTR 0
;  798     0859  3          THEN
;  799     0860  4              BEGIN
;  800     0861  4              FLAGS[FIRST_TIME] = 0;
;  801     0862  4              RELEASE (IRAB[IRB$L_CURBDB]);
;  802     0863  4              END
;  803     0864  4
;  804     0865  4          ! The SEARCH_RESULT can be less-than only if RMS was unable to find
;  805     0866  4          ! a SIDR record with a key equal to the key in keybuffer 2 either in
;  806     0867  4          ! the current bucket (if the NRP context information was being
;  807     0868  4          ! utilized) or anywhere in the alternate index (if it was not). Under
;  808     0869  4          ! such circumstances, the SIDR RMS has positioned to, which has a key
;  809     0870  4          ! greater than the key in keybuffer 2, is the desired SIDR.
;  810     0871  4          !
;  811     0872  3          ELSE
;  812     0873  3              IF .SEARCH_RESULT LSS 0
;  813     0874  3              THEN
;  814     0875  3
;  815     0876  3                  ! If RMS had been using the NRP context information to position
;  816     0877  3                  ! to the current SIDR then this means that the current SIDR has
;  817     0878  3                  ! been deleted. To indicate this IRB$W_SAVE_POS is left at
;  818     0879  3                  ! zero and an RMS error of record delete is returned although
;  819     0880  3                  ! whether this is an error or not depends upon the context in
;  820     0881  3                  ! which this routine is called.
;  821     0882  3                  !
;  822     0883  3                  IF .FLAGS[NRP]
;  823     0884  3                  THEN
;  824     0885  4                      RETURN RMSERR(DEL)
;  825     0886  4
;  826     0887  4                  ! If the NRP context information had never been used, then this
;  827     0888  4                  ! SIDR represents the desired SIDR, and a successful status is
;  828     0889  4                  ! returned.
```

```
;  829     0890  4                !
;  830     0891  3            ELSE
;  831     0892  4                RETURN RMSSUC()
;  832     0893  4
;  833     0894  4            ! RMS has successfully positioned to a SIDR with a key exactly
;  834     0895  4            ! matching the key to be found in keybuffer 2.
;  835     0896  4            !
;  836     0897  3            ELSE
;  837     0898  3
;  838     0899  3                ! If the SIDR RMS is to retrieve is in fact the current record,
;  839     0900  3                ! then even though RMS has been able to position to a SIDR
;  840     0901  3                ! with the same key as the current record, this SIDR may not be
;  841     0902  3                ! the current record. To determine this, the RFA pointer of
;  842     0903  3                ! the found SIDR's first array element must be compared with
;  843     0904  3                ! the RFA pointer of the first element of the current record
;  844     0905  3                ! which is part of the next record context found in the IRAB.
;  845     0906  3                !
;  846     0907  3                IF .FLAGS[NRP]
;  847     0908  3                THEN
;  848     0909  4                    BEGIN
;  849     0910  4
;  850     0911  4                    LOCAL
;  851     0912  4                        FIRST_VBN,
;  852     0913  4                        FIRST_ID;
;  853     0914  4
;  854     0915  4                    ! Retrieve the RFA pointer of the SIDR's first element.
;  855     0916  4                    !
;  856     0917  4                    RMS$SIDR_FIRST (1;FIRST_VBN, FIRST_ID);
;  857     0918  4
;  858     0919  4                    ! If the RFA pointer of the SIDR's first element matches
;  859     0920  4                    ! the RFA pointer of the current SIDR's first element
;  860     0921  4                    ! saved as part of the next record positioning context,
;  861     0922  4                    ! then RMS has been able to successfully position to the
;  862     0923  4                    ! current record. Load the number of array elements
;  863     0924  4                    ! preceeding the current element in to IRB$W_SAVE_POS and
;  864     0925  4                    ! return success.
;  865     0926  4                    !
;  866     0927  5                    IF  (.FIRST_VBN EQLU .IRAB[IRB$L_SIDR_VBN])
;  867     0928  4                     AND
;  868     0929  5                    (.FIRST_ID EQLU .IRAB[IRB$W_SIDR_ID])
;  869     0930  4                    THEN
;  870     0931  5                        BEGIN
;  871     0932  5                        IRAB[IRB$W_SAVE_POS] = .IRAB[IRB$W_CUR_COUNT];
;  872     0933  5                        RETURN RMSSUC();
;  873     0934  5                        END
;  874     0935  5
;  875     0936  5                    ! If either the VBN or the ID of the first element of the
;  876     0937  5                    ! current record and the first element of the SIDR RMS has
;  877     0938  5                    ! positioned to do not match, then the current SIDR has
;  878     0939  5                    ! been deleted, IRB$W_SAVE_POS is left at 0, and a RMS
;  879     0940  5                    ! error of record deleted is returned (although whether
;  880     0941  5                    ! this is an error or not depends upon the context in
;  881     0942  5                    ! which this routine is called.
;  882     0943  5                    !
;  883     0944  4                    ELSE
;  884     0945  4                        RETURN RMSERR(DEL);
;  885     0946  4                    END
```

```
;  886          0947  4
;  887          0948  4                    ! If RMS is not asked to specifically retrieve the current
;  888          0949  4                    ! record, then this SIDR with a key equal to the key in
;  889          0950  4                    ! keybuffer 2 is the desired SIDR, and a status of success
;  890          0951  4                    ! is returned.
;  891          0952  4                    !
;  892          0953  3                    ELSE
;  893          0954  3                        RETURN RMSSUC();
;  894          0955  3
;  895          0956  2                END;
;  896          0957  2
;  897          0958  2
;  898          0959  1            END;                                    ! {  of routine }
; INFO#212                    L1:0789
; Null expression appears in value-required context
```

```
                                    3C   BB 00000 RM$FIND_SIDR:
                                                  PUSHR   #^M<R2,R3,R4,R5>                    ; 0663
                                    53   D4 00002          CLRL    FLAGS                       ; 0773
                               53   01   88 00004          BISB2   #1, FLAGS                   ; 0774
                          76   A9   B4 00007          CLRW    118(IRAB)                   ; 0776
                    00C3  C9   21   A7 91 0000A          CMPB    33(IDX_DFN), 195(IRAB)      ; 0781
                                    03   12 00010          BNEQ    1$                          ;
                               53   02   88 00012          BISB2   #2, FLAGS                   ; 0783
                          31   53   01   E1 00015 1$:      BBC     #1, FLAGS, 2$               ; 0803
                               2E   53   E9 00019          BLBC    FLAGS, 2$                   ;
                          40   A9   01   90 0001C          MOVB    #1, 64(IRAB)                ; 0810
                               50   17   A7 9A 00020          MOVZBL  23(IDX_DFN), R0             ; 0816
                          7E   50   09   78 00024          ASHL    #9, R0, -(SP)               ;
                                    00A8 C9   DD 00028          PUSHL   168(IRAB)           ;
                                         0000G 30 0002C          BSBW    RM$GETBKT           ;
                               5E   08   C0 0002F          ADDL2   #8, SP                  ;
                               7E   50   E9 00032          BLBC    STATUS, 8$              ;
                          20   A9   54   D0 00035          MOVL    BDB, 32(IRAB)               ; 0817
                               50   20   A9 D0 00039          MOVL    32(IRAB), R0                ; 0823
                               55   18   A0 D0 0003D          MOVL    24(R0), BKT_ADDR            ;
                               56   0E   A5 9E 00041          MOVAB   14(R5), REC_ADDR            ; 0824
                                         0000G 30 00045          BSBW    RM$SRCH_BY_KEY          ; 0826
                                    1B   11 00048          BRB     3$                          ; 0803
                                         0000G 30 0004A 2$:      BSBW    RM$CSEARCH_TREE         ; 0838
                               63   50   E9 0004D          BLBC    STATUS, 8$                  ;
                                    7E   D4 00050          CLRL    -(SP)                       ; 0843
                               7E   20   A7 9A 00052          MOVZBL  32(IDX_DFN), -(SP)          ; 0844
                               50   00B4 CA 3C 00056          MOVZWL  180(IFAB), R0               ; 0843
                                    60 B940 9F 0005B          PUSHAB  @96(IRAB)[R0]           ;
                                         0000G 30 0005F          BSBW    RM$COMPARE_REC          ;
                               5E   0C   C0 00062          ADDL2   #12, SP                 ;
                                    50   D5 00065 3$:       TSTL    SEARCH_RESULT               ; 0858
                                    14   15 00067          BLEQ    4$                          ;
                               53   01   8A 00069          BICB2   #1, FLAGS                   ; 0861
                               54   20   A9 D0 0006C          MOVL    32(IRAB), BDB               ; 0862
                                    20   A9 D4 00070          CLRL    32(IRAB)                    ;
                                    7E   D4 00073          CLRL    -(SP)                       ;
```

```
                                                0000G 30 00075          BSBW    RM$RLSBKT                                       ;
                                          5E      04 C0 00078           ADDL2   #4, SP                                          ;
                                                  98 11 0007B           BRB     1$                                              ; 0858
                                                  06 18 0007D 4$:       BGEQ    5$                                              ; 0873
                                       2D     53  01 E1 0007F           BBC     #1, FLAGS, 7$                                   ; 0883
                                                  24 11 00083           BRB     6$                                              ; 0885
                                       27     53  01 E1 00085 5$:       BBC     #1, FLAGS, 7$                                   ; 09C7
                                                  01 DD 00089           PUSHL   #1                                              ; 0917
                                                0000G 30 0008B          BSBW    RM$SIDR_FIRST                                   ;
                                          5E      04 C0 0008E           ADDL2   #4, SP                                          ;
                                    00B4  C9      51 D1 00091           CMPL    FIRST_VBN, 180(IRAB)                            ; 0927
                                                  11 12 00096           BNEQ    6$                                              ;
                                52  00BE  C9          10 ED 00098       CMPZV   #0, #16, 190(IRAB), FIRST_ID                    ; 0929
                                                  08 12 0009F           BNEQ    6$                                              ;
                                       76     A9  00C0  C9 B0 000A1     MOVW    192(IRAB), 118(IRAB)                            ; 0932
                                                  07 11 000A7           BRB     7$                                              ; 0945
                                          50  8262  8F 3C 000A9 6$:     MOVZWL  #33378, R0                                      ;
                                                  03 11 000AE           BRB     8$                                              ;
                                          50      01 D0 000B0 7$:       MOVL    #1, R0                                          ; 0954
                                                  3C BA 000B3 8$:       POPR    #^M<R2,R3,R4,R5>                                ; 0959
                                                  05 000B5              RSB                                                     ;

; Routine Size:  182 bytes,    Routine Base:  RM$RMS3 + 0129
```

```
;  900     0960  1 %SBTTL 'RM$POS_SEQ'
;  901     0961  1 GLOBAL ROUTINE RM$POS_SEQ : RL$RABREG_67 =
;  902     0962  1
;  903     0963  1 !++
;  904     0964  1 !
;  905     0965  1 ! FUNCTIONAL DESCRIPTION:
;  906     0966  1 !
;  907     0967  1 !       This routines responsibility is to direct RMS's positioning to the next
;  908     0968  1 !       primary data record to be returned. In some cases the next primary data
;  909     0969  1 !       record will be the current primary data record (if IRB$V_SKIP_NEXT is
;  910     0970  1 !       not set and the current record is not deleted), but in most cases the
;  911     0971  1 !       next primary data record will in fact be the first non-deleted primary
;  912     0972  1 !       data record RMS can position to after positioning to the current
;  913     0973  1 !       primary data record or SIDR array element.
;  914     0974  1 !
;  915     0975  1 !       This routine handles the positioning itself if the primary key is the
;  916     0976  1 !       current key of reference, and it also directs the positioning to the
;  917     0977  1 !       very first SIDR if the current index is an alternate index, and this is
;  918     0978  1 !       the first positioning attempt after a $CONNECT or $REWIND. However,
;  919     0979  1 !       this routine handles all subsequent SIDR positionings by calling
;  920     0980  1 !       RM$FIND_SIDR to position to the current SIDR and RM$SEARCH_SIDR to
;  921     0981  1 !       position to the current array within the SIDR.
;  922     0982  1 !
;  923     0983  1 !       RMS's first task is to position to the current record (to the current
;  924     0984  1 !       SIDR array element if the current record is a SIDR). One basic
;  925     0985  1 !       assumption is made by this routine, and that is that if RMS is
;  926     0986  1 !       positioning by the primary key of reference and duplicate primary keys
;  927     0987  1 !       ARE allowed then this process's current record MUST be present in
;  928     0988  1 !       the file. This is because the only time the space held by a record can
;  929     0989  1 !       be reclaimed on that record's deletion is if duplicate primary keys are
;  930     0990  1 !       not allowed in the file. The same can not be said to be true for
;  931     0991  1 !       SIDRs. If every element in a SIDR array is deleted it will be possible
;  932     0992  1 !       to delete the entire SIDR regardless of whether it is any process's
;  933     0993  1 !       current record or not.
;  934     0994  1 !
;  935     0995  1 !       Once RMS has positioned to the current record, it can position to the
;  936     0996  1 !       primary data record to be returned (which may in fact be the current
;  937     0997  1 !       primary data record). This positioning is accomplished by this routine,
;  938     0998  1 !       if the primary key is the current key of reference, or by
;  939     0999  1 !       RM$SEARCH_SIDR if it is not.
;  940     1000  1 !
;  941     1001  1 !       The algorithm used by this routine in positioning to the current record
;  942     1002  1 !       when the primary key is the key of reference is as follows:
;  943     1003  1 !
;  944     1004  1 !       1) If this is the first time a positioning is attempted after a
;  945     1005  1 !          $CONNECT or $REWIND there is no current record so position to the
;  946     1006  1 !          very first non-deleted primary data record in the file.
;  947     1007  1 !
;  948     1008  1 !       2) Otherwise, the current record information is used to access the
;  949     1009  1 !          current bucket and search for the record in there.
;  950     1010  1 !
;  951     1011  1 !       3) If the current record is not found in the current bucket and the
;  952     1012  1 !          VBN of the current primary data record and current bucket differ,
;  953     1013  1 !          then attempt to find the current primary data record utilizing its
;  954     1014  1 !          RFA address.
;  955     1015  1 !
;  956     1016  1 !       4) If RMS is still unable to find the current record it must have been
```

```
;  957          1017  1 ;          deleted; therefore RMS performs a random search of the index either
;  958          1018  1 ;          to locate the deleted record (just marked deleted) or to find the
;  959          1019  1 ;          next record (prologue 3 files not allowing duplicate primary keys
;  960          1020  1 ;          only).
;  961          1021  1 ;
;  962          1022  1 ;          a) If duplicates are allowed the search is for the first record with
;  963          1023  1 ;             an identical key. A search is then made of all primary data
;  964          1024  1 ;             records with this key value until the current record, which is
;  965          1025  1 ;             guarenteed to be in the file, is found.
;  966          1026  1 ;
;  967          1027  1 ;          b) If duplicates are not allowed then the search is for the first
;  968          1028  1 ;             record with a key greater than or equal to the key of the
;  969          1029  1 ;             current record. This is because if duplicates are not allowed
;  970          1030  1 ;             a greater-than or equal search will position RMS to the first
;  971          1031  1 ;             record of identical or greater key value, and the next
;  972          1032  1 ;             non-deleted record, (including thwe one directly positioned to)
;  973          1033  1 ;             is the record RMS wants to return.
;  974          1034  1 ;
;  975          1035  1 ; CALLING SEQUENCE:
;  976          1036  1 ;       BSBW RM$POS_SEQ()
;  977          1037  1 ;
;  978          1038  1 ; INPUT PARAMETERS:
;  979          1039  1 ;       NONE
;  980          1040  1 ;
;  981          1041  1 ; IMPLICIT INPUTS:
;  982          1042  1 ;
;  983          1043  1 ;       IDX_DFN                 - address of index descriptor
;  984          1044  1 ;           IDX$B_DATBKTSZ      - size of data bucket
;  985          1045  1 ;           IDX$B_KEYREF        - key of reference
;  986          1046  1 ;
;  987          1047  1 ;       IFAB                    - address of IFAB
;  988          1048  1 ;           IFB$W_KBUFSZ        - keybuffer size
;  989          1049  1 ;           IFB$V_RU            - if set, file is Recovery Unit Journallable
;  990          1050  1 ;           IFB$V_WRTACC        - if set, stream has write access to the file
;  991          1051  1 ;
;  992          1052  1 ;       IRAB                    - address of IRAB
;  993          1053  1 ;           IRB$W_CUR_ID        - ID of current record
;  994          1054  1 ;           IRB$L_CUR_VBN       - VBN of current record
;  995          1055  1 ;           IRB$L_KEYBUF        - address of keybuffers
;  996          1056  1 ;           IRB$V_SKIP_NEXT     - if set, skip current record and return next
;  997          1057  1 ;           IRB$W_POS_ID        - ID of user data record for NRP positioning
;  998          1058  1 ;           IRB$L_POS_VBN       - VBN of user data record for NRP positioning
;  999          1059  1 ;
; 1000          1060  1 ; OUTPUT PARAMETERS:
; 1001          1061  1 ;       NONE
; 1002          1062  1 ;
; 1003          1063  1 ; IMPLICIT OUTPUTS:
; 1004          1064  1 ;
; 1005          1065  1 ;       IRB$L_CURBDB    - address of the BDB describing the primary data bucket
; 1006          1066  1 ;       IRB$V_DEL_SEEN  - deleted records seen between current and next record
; 1007          1067  1 ;       IRB$W_FIRST_ID  - ID of the next SIDR's first element's RRV pointer
; 1008          1068  1 ;       IRB$L_FIRST_VBN - VBN of the next SIDR's first element's RRV pointer
; 1009          1069  1 ;       IRB$W_RFA_ID    - ID of the next record (primary data only)
; 1010          1070  1 ;       IRB$L_RFA_VBN   - VBN of the next record (SIDR or primary data)
; 1011          1071  1 ;       IRB$W_SAVE_POS  - number of elements preceeding next SIDR array element
; 1012          1072  1 ;       REC_ADDR        - address of next primary data record
; 1013          1073  1 ;
```

```
;  1014          1074  1 ! ROUTINE VALUE:
;  1015          1075  1 !
;  1016          1076  1 !       EOF - end-of-file encountered before record to be returned is found.
;  1017          1077  1 !       SUC - found non-deleted primary data record to return.
;  1018          1078  1 !       BUG - serious internal error precluded positioning to a record.
;  1019          1079  1 !       various I/O errors
;  1020          1080  1 !
;  1021          1081  1 ! SIDE EFFECTS:
;  1022          1082  1 !
;  1023          1083  1 !       SUC - if the primary key is the key of reference, IRB$L_RFA_VBN and
;  1024          1084  1 !             IRB$W_RFA_ID have their assigned values, and if not,
;  1025          1085  1 !             IRB$L_RFA_VBN, IRB$L_FIRST_VBN, IRB$W_FIRST_ID, and
;  1026          1086  1 !             IRB$W_SAVE_POS have their assign values. In both cases, REC_ADDR,
;  1027          1087  1 !             IRB$V_DEL_SEEN, and IRB$L_CURBDB have their assigned values.
;  1028          1088  1 !       In all other cases, these fields have indeterminate values and all
;  1029          1089  1 !           bucket locks have been released.
;  1030          1090  1 !       AP IS TRASHED
;  1031          1091  1 !       IRB$B_STOPLEVEL is 0
;  1032          1092  1 !       IRB$W_SRCHFLGS is of indeterminate status
;  1033          1093  1 !
;  1034          1094  1 !--
;  1035          1095  1
;  1036          1096  2     BEGIN
;  1037          1097  2
;  1038          1098  2     EXTERNAL REGISTER
;  1039          1099  2         COMMON_RAB_STR,
;  1040          1100  2         R_IDX_DFN_STR,
;  1041          1101  2         R_REC_ADDR_STR;
;  1042          1102  2
;  1043          1103  2
;  1044          1104  2     ! Initialize some search control fields.
;  1045          1105  2     !
;  1046          1106  2     IRAB[IRB$B_STOPLEVEL] = 0;
;  1047          1107  2     IRAB[IRB$W_SRCHFLAGS] = 0;
;  1048          1108  2
;  1049          1109  2     ! If this is the very first time RMS is positioning after a $CONNECT or
;  1050          1110  2     ! $REWIND then position to the very first non-RRV record in the very first
;  1051          1111  2     ! SIDR/primary data bucket.
;  1052          1112  2     !
;  1053          1113  3     IF  (.IRAB[IRB$L_CUR_VBN] EQLU 0)
;  1054          1114  2          AND
;  1055          1115  3         (.IRAB[IRB$L_POS_VBN] EQLU 0)
;  1056          1116  2     THEN
;  1057          1117  3         BEGIN
;  1058          1118  3
;  1059          1119  3         IRAB[IRB$V_FIRST_TIM] = 1;
;  1060          1120  3         RET_ON_ERR_EOF(RM$CSEARCH_TREE(), IRAB[IRB$V_FIRST_TIM] = 0);
;  1061          1121  3
;  1062          1122  3         ! If RM$CSEARCH_TREE positioned RMS to an empty bucket, position to the
;  1063          1123  3         ! first non-empty bucket or until the end-of-file is encountered.
;  1064          1124  3         !
;  1065          1125  3         WHILE 1
;  1066          1126  3         DO
;  1067          1127  4             BEGIN
;  1068          1128  4
;  1069          1129  4             GLOBAL REGISTER
;  1070          1130  4                 COMMON_IO_STR;
```

```
;  1071     1131  4
;  1072     1132  4          LOCAL
;  1073     1133  4              VBN;
;  1074     1134  4
;  1075     1135  4          BKT_ADDR = .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$L_ADDR];
;  1076     1136  4
;  1077     1137  4          ! If RMS has positioned to a RRV in a primary data bucket, or if
;  1078     1138  4          ! the bucket is empty, then RMS nust release this bucket, and access
;  1079     1139  4          ! the next one in the horizontal chain. Otherwise, RMS starts its
;  1080     1140  4          ! search for the first non-deleted primary data record with the
;  1081     1141  4          ! first record in this bucket.
;  1082     1142  4          !
;  1083     1143  6          IF  NOT ((.IDX_DFN[IDX$B_KEYREF] EQLU 0
;  1084     1144  6                              AND
;  1085     1145  6                             .REC_ADDR[IRC$V_RRV])
;  1086     1146  5                    OR
;  1087     1147  5                   (.BKT_ADDR[BKT$W_FREESPACE] LEQU BKT$C_OVERHDSZ))
;  1088     1148  4          THEN
;  1089     1149  4              EXITLOOP;
;  1090     1150  4
;  1091     1151  4          ! If this is the last data bucket in the horizontal chain then
;  1092     1152  4          ! every single bucket is empty. End the search for the first
;  1093     1153  4          ! non-empty bucket and return end-of-file.
;  1094     1154  4          !
;  1095     1155  4          IF .BKT_ADDR[BKT$V_LASTBKT]
;  1096     1156  4          THEN
;  1097     1157  5              BEGIN
;  1098     1158  5              RELEASE(IRAB[IRB$L_CURBDB]);
;  1099     1159  5              RETURN RMSERR(EOF);
;  1100     1160  4              END;
;  1101     1161  4
;  1102     1162  4          ! Save the VBN of the next bucket in the horizontal chain, release
;  1103     1163  4          ! the current bucket, and then access the next bucket by its VBN.
;  1104     1164  4          !
;  1105     1165  4          VBN = .BKT_ADDR[BKT$L_NXTBKT];
;  1106     1166  4          RELEASE(IRAB[IRB$L_CURBDB]);
;  1107     1167  4
;  1108     1168  4          ! If the file is Recovery Unit Journallable and write accessed or
;  1109     1169  4          ! the positioning is being done by alternate key of reference, then
;  1110     1170  4          ! make sure the next data bucket in the horizontal chain is
;  1111     1171  4          ! exclusively accessed.
;  1112     1172  4          !
;  1113     1173  5          IF  (.IFAB[IFB$V_WRTACC]
;  1114     1174  5                      AND
;  1115     1175  5                     .IFAB[IFB$V_RU])
;  1116     1176  4               OR
;  1117     1177  4              .IDX_DFN[IDX$B_KEYREF] NEQU 0
;  1118     1178  4          THEN
;  1119     1179  4              IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  1120     1180  4
;  1121     1181  4          RETURN_ON_ERROR (RM$GETBKT(.VBN, .IDX_DFN[IDX$B_DATBKTSZ]*512));
;  1122     1182  4
;  1123     1183  4          IRAB[IRB$L_CURBDB] = .BDB;
;  1124     1184  4          REC_ADDR = .BKT_ADDR + BKT$C_OVERHDSZ;
;  1125     1185  3          END;
;  1126     1186  3
;  1127     1187  3      ! When the first non-empty bucket has been encountered and REC_ADDR
```

```
;  1128     1188  3          ! is setup to point to the first record in that bucket, then clear the
;  1129     1189  3          ! first time flag.
;  1130     1190  3          !
;  1131     1191  3          IRAB[IRB$V_FIRST_TIM] = 0;
;  1132     1192  3
;  1133     1193  3          ! If the key of reference is the primary key, then RMS must find the
;  1134     1194  3          ! first non-deleted primary data record in the file. If this is an
;  1135     1195  3          ! alternate index, RMS does not need to do this positioning because
;  1136     1196  3          ! that will be done when RMS positions to the first non-deleted
;  1137     1197  3          ! primary data record.
;  1138     1198  3          !
;  1139     1199  3          IF .IDX_DFN[IDX$B_KEYREF] EQL 0
;  1140     1200  3          THEN
;  1141     1201  4              RETURN_ON_ERROR (RM$FIND_NONDEL (1))
;  1142     1202  4
;  1143     1203  4          ! If the key of reference is not the primary key, then RMS is to
;  1144     1204  4          ! start its search for the first non-deleted primary data record in
;  1145     1205  4          ! the file with the very first array element of the SIDR it has
;  1146     1206  4          ! positioned to.
;  1147     1207  4          !
;  1148     1208  3          ELSE
;  1149     1209  3              IRAB[IRB$W_SAVE_POS] = 0;
;  1150     1210  3          END
;  1151     1211  3
;  1152     1212  3      ! This is not the first time RMS has sequentially accessed the file since
;  1153     1213  3      ! a $CONNECT, or $REWIND was done. If RMS is positioning by means of an
;  1154     1214  3      ! alternate key of reference, then all of the positioning to the current
;  1155     1215  3      ! SIDR and the current array element within that SIDR is done by the
;  1156     1216  3      ! routine RM$FIND_SIDR.
;  1157     1217  3      !
;  1158     1218  2      ELSE
;  1159     1219  2          IF .IDX_DFN[IDX$B_KEYREF] NEQ 0
;  1160     1220  2          THEN
;  1161     1221  3              BEGIN
;  1162     1222  3
;  1163     1223  3              LOCAL
;  1164     1224  3                  STATUS;
;  1165     1225  3
;  1166     1226  3              IRAB[IRB$V_SRCHGE] = 1;
;  1167     1227  3
;  1168     1228  3              ! If some serious error occurred during RMS's positioning to the
;  1169     1229  3              ! current SIDR, then immediately return the error. However, if
;  1170     1230  3              ! RMS has successfully positioned to the current SIDR then
;  1171     1231  3              ! IRB$W_SAVE_POS contains the number of array elements preceeding
;  1172     1232  3              ! the current SIDR array element.
;  1173     1233  3              !
;  1174     1234  5              IF NOT ((STATUS = RM$FIND_SIDR())
;  1175     1235  4                      OR
;  1176     1236  4                     (.STATUS<0,16> EQL RMSERR(DEL)))
;  1177     1237  3              THEN
;  1178     1238  3
;  1179     1239  3                  ! If the current SIDR was the last SIDR in the file, and it
;  1180     1240  3                  ! has been deleted, return EOF indicating that the end of the
;  1181     1241  3                  ! file has been encountered; otherwise, return the actual error.
;  1182     1242  3                  !
;  1183     1243  4                  IF .STATUS<0,16> EQLU RMSERR(RNF)
;  1184     1244  3                  THEN
```

```
;  1185     1245  4                RETURN RMSERR(EOF)
;  1186     1246  4            ELSE
;  1187     1247  4                RETURN .STATUS;
;  1188     1248  4
;  1189     1249  4            ! If RMS is successful at positioning to the current SIDR then
;  1190     1250  4            ! setup to return the next record provided RMS is to return the
;  1191     1251  4            ! next record (IRB$V_SKIP_NEXT is set).
;  1192     1252  4            !
;  1193     1253  4            IF .STATUS AND .IRAB[IRB$V_SKIP_NEXT]
;  1194     1254  4            THEN
;  1195     1255  4                IRAB[IRB$W_SAVE_POS] = .IRAB[IRB$W_SAVE_POS] + 1;
;  1196     1256  4            END
;  1197     1257  4
;  1198     1258  4        ! If this is not the first time sequentially accessing a record after
;  1199     1259  4        ! a $REWIND or $CONNECT, and RMS is to position by the primary key of
;  1200     1260  4        ! reference, then RMS's strategy is to first position to the current
;  1201     1261  4        ! record which must be present somewhere in the file if it allows
;  1202     1262  4        ! duplicate primary keys. RMS will return either its position or that
;  1203     1263  4        ! of the next non-deleted primary data record which follows it based
;  1204     1264  4        ! upon whether or not it found the current record to be deleted, and
;  1205     1265  4        ! whether it was to return the current or the next record.
;  1206     1266  4        !
;  1207     1267  4        ELSE
;  1208     1268  4            BEGIN
;  1209     1269  4
;  1210     1270  4            FIELD
;  1211     1271  4                POS_FLAGS =
;  1212     1272  4                    SET
;  1213     1273  4                    FND_BY_RRV  = [0,0,1,0],
;  1214     1274  4                    FND_BY_SRCH = [0,1,1,0]
;  1215     1275  4                    TES;
;  1216     1276  4
;  1217     1277  3            LOCAL
;  1218     1278  3                FLAG    : BLOCK[1,BYTE]
;  1219     1279  3                          FIELD (POS_FLAGS);
;  1220     1280  3
;  1221     1281  3            FLAG = 0;
;  1222     1282  3
;  1223     1283  3            ! RMS will use the VBN and ID of the current record positioning
;  1224     1284  3            ! information to attempt to position directly to the current
;  1225     1285  3            ! primary data record.
;  1226     1286  3            !
;  1227     1287  4            BEGIN
;  1228     1288  4
;  1229     1289  4            BUILTIN
;  1230     1290  4                AP;
;  1231     1291  4
;  1232     1292  4            LOCAL
;  1233     1293  4                STATUS;
;  1234     1294  4
;  1235     1295  4            ! If the file is Recovery Unit Journallable and write accessed, then
;  1236     1296  4            ! make sure the current primary data bucket exclusively accessed.
;  1237     1297  4            !
;  1238     1298  4            IF  .IFAB[!FB$V_WRTACC]
;  1239     1299  4                 AND
;  1240     1300  4                .IFAB[IFB$V_RU]
;  1241     1301  4            THEN
```

```
;  1242          1302  4                    IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  1243          1303  4
;  1244          1304  4                AP = .IRAB[IRB$W_CUR_ID];
;  1245          1305  4                STATUS = RM$FIND_BY_RFA (.IRAB[IRB$L_CUR_VBN]);
;  1246          1306  4
;  1247          1307  4                ! On any errors other than record not found or record deleted
;  1248          1308  4                ! return immediately. These will be I/O errors.
;  1249          1309  4                !
;  1250          1310  5                IF NOT (.STATUS
;  1251          1311  5                        OR
;  1252          1312  6                       (.STATUS<0,16> EQL RMSERR(RNF))
;  1253          1313  5                        OR
;  1254          1314  5                       (.STATUS<0,16> EQL RMSERR(DEL)))
;  1255          1315  4                THEN
;  1256          1316  4                    RETURN .STATUS;
;  1257          1317  4
;  1258          1318  4                ! If RMS is unable to find any record through use of the
;  1259          1319  4                ! current record information, or if it finds a RRV then this
;  1260          1320  4                ! indicates that the current record must have moved, or may have
;  1261          1321  4                ! been deleted.
;  1262          1322  4                !
;  1263          1323  4                IF  NOT .STATUS
;  1264          1324  4                     OR
;  1265          1325  4                    .REC_ADDR[IRC$V_RRV]
;  1266          1326  4                THEN
;  1267          1327  5                    BEGIN
;  1268          1328  5
;  1269          1329  5                    GLOBAL REGISTER
;  1270          1330  5                        COMMON_IO_STR;
;  1271          1331  5
;  1272          1332  5                    ! If the VBN of the current record is not the same as the
;  1273          1333  5                    ! VBN of the current record's RFA address, then RMS has no
;  1274          1334  5                    ! way of knowing what has happened to the record. RMS has
;  1275          1335  5                    ! no way of knowing whether it has not been deleted and
;  1276          1336  5                    ! can still be access through its RFA, or whether it has
;  1277          1337  5                    ! been deleted and can not. In this case, RMS must search
;  1278          1338  5                    ! for the record by its RFA address, saved as part of the
;  1279          1339  5                    ! next record positioning context information, to confirm
;  1280          1340  5                    ! its status.
;  1281          1341  5                    !
;  1282          1342  5                    IF .IRAB[IRB$L_CUR_VBN] NEQ .IRAB[IRB$L_POS_VBN]
;  1283          1343  5                    THEN
;  1284          1344  5                        FLAG[FND_BY_RRV] = 1
;  1285          1345  5
;  1286          1346  5                    ! If RMS has positioned to a RRV through the use of the
;  1287          1347  5                    ! current record information, and the VBN of that bucket is
;  1288          1348  5                    ! the same as the VBN of the current record's RFA address,
;  1289          1349  5                    ! then RMS has positioned to the RRV for the current
;  1290          1350  5                    ! record. Provided, the RRV is not marked deleted, RMS will
;  1291          1351  5                    ! be able to position to the current record by means of its
;  1292          1352  5                    ! RFA address.
;  1293          1353  5                    !
;  1294          1354  5                    ELSE
;  1295          1355  5                        IF  .STATUS
;  1296          1356  5                             AND
;  1297          1357  5                            .REC_ADDR[IRC$V_RRV]
;  1298          1358  5                             AND
```

```
;  1299          1359  5                    NOT .REC_ADDR[IRC$V_DELETED]
;  1300          1360  5                THEN
;  1301          1361  5                    FLAG[FND_BY_RRV] = 1
;  1302          1362  5
;  1303          1363  5                ! If the VBN of the current record is the same as the
;  1304          1364  5                ! VBN of the current record's RFA address, and RMS
;  1305          1365  5                ! was unable to position to a record (Prologue 3 - RRVs
;  1306          1366  5                ! are deleted entirely although the record is just
;  1307          1367  5                ! marked), or it was able to position to a deleted RRV
;  1308          1368  5                ! (Prologue 1), then this indicates that the current
;  1309          1369  5                ! record has been deleted. In such a situation a random
;  1310          1370  5                ! search of the primary index will have to be done
;  1311          1371  5                ! either to locate the next record itself (primary key
;  1312          1372  5                ! duplicates NOT allowed), or to locate the current record.
;  1313          1373  5                !
;  1314          1374  5                ELSE
;  1315          1375  5                    FLAG[FND_BY_SRCH] = 1;
;  1316          1376  5
;  1317          1377  5            ! It will be necessary to release the primary data bucket
;  1318          1378  5            ! described by the current record information, if RMS has
;  1319          1379  5            ! positioned to an RRV, and the bucket is still locked.
;  1320          1380  5            !
;  1321          1381  5            IF .STATUS
;  1322          1382  5            THEN
;  1323          1383  5                RELEASE (IRAB[IRB$L_CURBDB]);
;  1324          1384  5
;  1325          1385  5            END
;  1326          1386  5
;  1327          1387  5        ! RMS has successfully positioned to a non-RRV record by means
;  1328          1388  5        ! of the next record positioning context information in the
;  1329          1389  5        ! IRAB. This does not mean that RMS has positioned to the
;  1330          1390  5        ! current record because it would be possible in a prologue 1 file
;  1331          1391  5        ! for the current record to move, and its ID re-used by another
;  1332          1392  5        ! record (ID recycling is allowed in one special circumstance).
;  1333          1393  5        ! Therefore, that the record RMS has positioned to is in fact the
;  1334          1394  5        ! current record must be verified by comparing the RRV in the
;  1335          1395  5        ! record positioned to with the RFA address of the current record.
;  1336          1396  5        !
;  1337          1397  4        ELSE
;  1338          1398  4
;  1339          1399  4            ! If the RRV of the record RMS has positioned to is not the
;  1340          1400  4            ! same as the RFA address of the current record, then RMS
;  1341          1401  4            ! has failed to position to the current record. If this is
;  1342          1402  4            ! a prologue 3 file, then this represents a serious error
;  1343          1403  4            ! because IDs within a bucket may not be recycled. If this
;  1344          1404  4            ! is a prologue 1 file this does not represent a serious
;  1345          1405  4            ! error because IDs maybe recycled; therefore, attempt to
;  1346          1406  4            ! position to the current record by means of its RFA
;  1347          1407  4            ! address.
;  1348          1408  4            !
;  1349          1409  4            IF NOT RM$CHECK_RRV()
;  1350          1410  4            THEN
;  1351          1411  5                BEGIN
;  1352          1412  5
;  1353          1413  5                GLOBAL REGISTER
;  1354          1414  5                    COMMON_IO_STR;
;  1355          1415  5
```

```
;  1356          1416  5                RELEASE (IRAB[IRB$L_CURBDB]);
;  1357          1417  5
;  1358          1418  5                IF .IFAB[IFB$B_PLG_VER] GEQU PLG$C_VER_3
;  1359          1419  5                THEN
;  1360          1420  6                    RETURN RMSERR(BUG)
;  1361          1421  5                ELSE
;  1362          1422  5                    FLAG[FND_BY_RRV] = 1
;  1363          1423  5                END
;  1364          1424  5
;  1365          1425  5            ! If the RRV of the record RMS has positioned to is the
;  1366          1426  5            ! same as the RFA address of the current record, then RMS
;  1367          1427  5            ! has successfully positioned to the current record.
;  1368          1428  5            !
;  1369          1429  5            ! Look for the next non-deleted primary data record. Begin
;  1370          1430  5            ! the search with the current record if and only if the
;  1371          1431  5            ! state bit IRB$V_SKIP_NEXT is clear.
;  1372          1432  5            !
;  1373          1433  4            ELSE
;  1374        P 1434  4                RETURN_ON_ERROR
;  1375          1435  4                    (RM$FIND_NONDEL (NOT .IRAB[IRB$V_SKIP_NEXT]));
;  1376          1436  3        END;
;  1377          1437  3
;  1378          1438  3        ! If RMS was unable to position to the current record by means
;  1379          1439  3        ! of the current record information, then it may try to position
;  1380          1440  3        ! to the current record by means of its RFA address. It will try
;  1381          1441  3        ! to do so as long as it knows that the current record has not
;  1382          1442  3        ! been deleted.
;  1383          1443  3        !
;  1384          1444  3        IF .FLAG[FND_BY_RRV]
;  1385          1445  3        THEN
;  1386          1446  4            BEGIN
;  1387          1447  4
;  1388          1448  4            LOCAL
;  1389          1449  4                STATUS;
;  1390          1450  4
;  1391          1451  4            ! If the file is Recovery Unit Journallable and write accessed,
;  1392          1452  4            ! then make sure the primary data bucket bucket containing the
;  1393          1453  4            ! current primary data record is exclusively accessed.
;  1394          1454  4            !
;  1395          1455  4            IF .IFAB[IFB$V_WRTACC]
;  1396          1456  4                AND
;  1397          1457  4               .IFAB[IFB$V_RU]
;  1398          1458  4            THEN
;  1399          1459  4                IRAB[IRB$B_CACHEFLGS] = CSH$M_LOCK;
;  1400          1460  4
;  1401          1461  4            ! Position to the current record by means of its RFA address.
;  1402          1462  4            !
;  1403          1463  4            STATUS = RM$FIND_BY_RRV (.IRAB[IRB$L_POS_VBN],
;  1404          1464  4                                     .IRAB[IRB$W_POS_ID],
;  1405          1465  4                                     0);
;  1406          1466  4
;  1407          1467  4            ! If RMS is unable to position to the current record by means
;  1408          1468  4            ! of its RFA address, and the status is other than record
;  1409          1469  4            ! deleted, then save the real reason for the failure in the
;  1410          1470  4            ! RAB's STV field and return an invalid internal condition.
;  1411          1471  4            !
;  1412          1472  4            IF NOT .STATUS
```

```
;  1413          1473  4                    AND
;  1414          1474  5                   (.STATUS<0,16> NEQ RMSERR(DEL))
;  1415          1475  4                THEN
;  1416          1476  5                    BEGIN
;  1417          1477  5                    RAB[RAB$L_STV] = .STATUS;
;  1418          1478  5                    RETURN RMSERR(BUG);
;  1419          1479  5                    END
;  1420          1480  5
;  1421          1481  5                ! If RMS is unable to position to the current record because
;  1422          1482  5                ! it has been deleted, then indicate that a random search of
;  1423          1483  5                ! the primary index to either locate the current record or
;  1424          1484  5                ! to locate the next primary data record (primary key
;  1425          1485  5                ! duplicates not allowed) must be done.
;  1426          1486  5                !
;  1427          1487  4                ELSE
;  1428          1488  4                    IF NOT .STATUS
;  1429          1489  4                    THEN
;  1430          1490  4                        FLAG[FND_BY_SRCH] = 1
;  1431          1491  4
;  1432          1492  4                    ! RMS has successfully positioned to a primary data record
;  1433          1493  4                    ! through the use of the RFA address of the current record
;  1434          1494  4                    ! saved as part of the IRAB's next record positioning
;  1435          1495  4                    ! context. Before continuing, verify that the record RMS
;  1436          1496  4                    ! has positioned to is in fact the current record.
;  1437          1497  4                    !
;  1438          1498  4                    ELSE
;  1439          1499  4
;  1440          1500  4                        ! If the RRV of the record RMS has positioned to is
;  1441          1501  4                        ! not the same as the RFA address of the current record
;  1442          1502  4                        ! then RMS has failed to position to the current record
;  1443          1503  4                        ! by means of the next record context information. This
;  1444          1504  4                        ! represents a serious error, regardless of the
;  1445          1505  4                        ! prologue. Release the primary data bucket and return
;  1446          1506  4                        ! a status of invalid internal condition or bug error.
;  1447          1507  4                        !
;  1448          1508  4                    IF NOT RM$CHECK_RRV()
;  1449          1509  4                    THEN
;  1450          1510  5                        BEGIN
;  1451          1511  5
;  1452          1512  5                        GLOBAL REGISTER
;  1453          1513  5                            COMMON_IO_STR;
;  1454          1514  5
;  1455          1515  5                        RELEASE (IRAB[IRB$L_CURBDB]);
;  1456          1516  5
;  1457          1517  6                        RETURN RMSERR(BUG)
;  1458          1518  5                        END
;  1459          1519  5
;  1460          1520  5                    ! If the RRV of the record RMS has positioned to is the
;  1461          1521  5                    ! same as the RFA address of the current record, then
;  1462          1522  5                    ! RMS has successfully positioned to the current record
;  1463          1523  5                    ! by means of its RFA address.
;  1464          1524  5                    !
;  1465          1525  5                    ! Look for the next non-deleted primary data record.
;  1466          1526  5                    ! Begin the search with the current record if and
;  1467          1527  5                    ! only if the state bit IRB$V_SKIP_NEXT is clear.
;  1468          1528  5                    !
;  1469          1529  4                ELSE
```

```
;  1470     P 1530  4                    RETURN_ON_ERROR
;  1471       1531  4                        (RM$FIND_NONDEL (NOT .IRAB[IRB$V_SKIP_NEXT]));
;  1472       1532  3                END;
;  1473       1533  3
;  1474       1534  3            ! If the current record has been deleted then a random search
;  1475       1535  3            ! of the primary index must be done as part of positioning to
;  1476       1536  3            ! the primary data record to be returned.
;  1477       1537  3            !
;  1478       1538  3            IF .FLAG[FND_BY_SRCH]
;  1479       1539  3            THEN
;  1480       1540  4                BEGIN
;  1481       1541  4
;  1482       1542  4                ! If duplicate keys are not allowed and the current record is
;  1483       1543  4                ! deleted then it will have been completely deleted from a
;  1484       1544  4                ! prologue 3 file, and only a RRV will be left in a prologue 2
;  1485       1545  4                ! file. Therefore, a search for the first non-deleted primary
;  1486       1546  4                ! data record with a key equal to or greater than the key of
;  1487       1547  4                ! the current record will position RMS to the record it is to
;  1488       1548  4                ! return.
;  1489       1549  4                !
;  1490       1550  4                IF NOT .IDX_DFN[IDX$V_DUPKEYS]
;  1491       1551  4                THEN
;  1492       1552  5                    BEGIN
;  1493       1553  5
;  1494       1554  5                    ! Position to the next first record with a key value equal
;  1495       1555  5                    ! to or greater than the key of the current record.
;  1496       1556  5                    !
;  1497       1557  5                    IRAB[IRB$V_SRCHGE] = 1;
;  1498       1558  5                    RETURN_ON_ERROR (RM$CSEARCH_TREE());
;  1499       1559  5
;  1500       1560  5                    ! Return the first non-deleted primary record encountered
;  1501       1561  5                    ! beginning the search with the record RMS has positioned
;  1502       1562  5                    ! to.
;  1503       1563  5                    !
;  1504       1564  5                    RETURN_ON_ERROR (RM$FIND_NONDEL(1));
;  1505       1565  5                    END
;  1506       1566  5
;  1507       1567  5                ! If duplicate primary keys are allowed then the deleted
;  1508       1568  5                ! current record must still be somewhere in the file although
;  1509       1569  5                ! marked deleted. Position to it by searching all records of
;  1510       1570  5                ! key value identical to that of the current record until it
;  1511       1571  5                ! is located.
;  1512       1572  5                !
;  1513       1573  4                ELSE
;  1514       1574  5                    BEGIN
;  1515       1575  5
;  1516       1576  5                    ! Search for the first record in the file with the same
;  1517       1577  5                    ! primary key value as the key of the current record. If
;  1518       1578  5                    ! none is found then this represents a serious problem
;  1519       1579  5                    ! because the current record itself must always be
;  1520       1580  5                    ! somewhere in the file even if it has been deleted.
;  1521       1581  5                    !
;  1522       1582  5                    IF NOT RM$CSEARCH_TREE()
;  1523       1583  5                    THEN
;  1524       1584  6                        RETURN RMSERR(BUG)
;  1525       1585  6
;  1526       1586  6                    ! RMS has successfully positioned to a primary data record
```

```
;  1527     1587  6                    ! with the same key as that of the current record.
;  1528     1588  6                    ! Somewhere in this bucket (or in any of the following
;  1529     1589  6                    ! continuation buckets if duplicates are allowed) must be
;  1530     1590  6                    ! the (marked) deleted current record. Position to the
;  1531     1591  6                    ! current record. After positioning to the current record,
;  1532     1592  6                    ! the next non-deleted primary data record in the file is
;  1533     1593  6                    ! the record RMS wants to return.
;  1534     1594  6                    !
;  1535     1595  5                    ELSE
;  1536     1596  6                        BEGIN
;  1537     1597  6
;  1538     1598  6                        LOCAL
;  1539     1599  6                            STATUS;
;  1540     1600  6
;  1541     1601  6                        ! Since the current record must be in the file even if
;  1542     1602  6                        ! it is just marked deleted, RMS's search criteria is
;  1543     1603  6                        ! to sequentially find all records with key value
;  1544     1604  6                        ! equal to that of the current record until the current
;  1545     1605  6                        ! record is positioned to.
;  1546     1606  6                        !
;  1547     1607  6                        STATUS = 1;
;  1548     1608  6
;  1549     1609  6                        DO
;  1550     1610  6
;  1551     1611  6                            ! The record RMS has currently positioned to is
;  1552     1612  6                            ! the current record when its RRV matches the RFA
;  1553     1613  6                            ! address of the current record. In such a case
;  1554     1614  6                            ! find the next non-deleted primary data record in
;  1555     1615  6                            ! the file.
;  1556     1616  6                            !
;  1557     1617  6                            IF RM$CHECK_RRV()
;  1558     1618  6                            THEN
;  1559     1619  7                                BEGIN
;  1560     1620  7                                RETURN_ON_ERROR (RM$FIND_NONDEL (0));
;  1561     1621  7                                EXITLOOP;
;  1562     1622  7                                END
;  1563     1623  7
;  1564     1624  7                            ! If the record RMS has positioned to is not the
;  1565     1625  7                            ! current record (its RRV does not match the RFA
;  1566     1626  7                            ! of the current record), then position to the next
;  1567     1627  7                            ! record. RM$CSEARCH_TREE will then determine
;  1568     1628  7                            ! whether the key of this next record is the same
;  1569     1629  7                            ! as the key of the current record.
;  1570     1630  7                            !
;  1571     1631  6                            ELSE
;  157-     1632  6                                RM$GETNEXT_REC()
;  1573     1633  6
;  1574     1634  6                            WHILE (STATUS = RM$CSEARCH_TREE());
;  1575     1635  6
;  1576     1636  6                    ! If RMS was unable to find the current record in the
;  1577     1637  6                    ! file then this represents a serious error because
;  1578     1638  6                    ! current records can never be completely deleted from
;  1579     1639  6                    ! the file while they remain current records. Return
;  1580     1640  6                    ! status indicative of an invalid internal condition.
;  1581     1641  6                    !
;  1582     1642  6                    IF NOT .STATUS
;  1583     1643  6                    THEN
```

```
;  1584     1644  7                        IF .STATUS<0,16> EQL RMSERR(RNF)
;  1585     1645  6                        THEN
;  1586     1646  7                            RETURN RMSERR(BUG)
;  1587     1647  6                        ELSE
;  1588     1648  6                            RETURN .STATUS;
;  1589     1649  5                    END;
;  1590     1650  4                END;
;  1591     1651  3            END;
;  1592     1652  2        END;
;  1593     1653  2
;  1594     1654  2        ! If RMS is positioning by primary key of reference then all RMS has to do
;  1595     1655  2        ! is set up the next record list context for this record, and return
;  1596     1656  2        ! success.
;  1597     1657  2        !
;  1598     1658  2        IF .IDX_DFN[IDX$B_KEYREF] EQL 0
;  1599     1659  2        THEN
;  1600     1660  3            BEGIN
;  1601     1661  3            IRAB[IRB$W_RFA_ID] = IRC$_ID(REC_ADDR);
;  1602     1662  3            IRAB[IRB$L_RFA_VBN] = .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$L_VBN];
;  1603     1663  3            BBLOCK[.IRAB[IRB$L_CURBDB], BDB$V_PRM] = 1;
;  1604     1664  4            RETURN RMSSUC()
;  1605     1665  3            END
;  1606     1666  3
;  1607     1667  3        ! If RMS is positioning by means of an alternate key, it still must
;  1608     1668  3        ! position to the array element pointing to the non-deleted primary data
;  1609     1669  3        ! record to be returned.
;  1610     1670  3        !
;  1611     1671  2        ELSE
;  1612     1672  3            BEGIN
;  1613     1673  3
;  1614     1674  3            LOCAL
;  1615     1675  3                STATUS;
;  1616     1676  3
;  1617     1677  3            STATUS = 1;
;  1618     1678  3
;  1619     1679  3            ! Search all SIDRs having the same key as the SIDR RMS has positioned
;  1620     1680  3            ! to until the primary data record to be returned has been found, or
;  1621     1681  3            ! until there are no more SIDRs with this key value.
;  1622     1682  3            !
;  1623     1683  3            WHILE .STATUS
;  1624     1684  3            DO
;  1625     1685  4                BEGIN
;  1626     1686  4                IRAB[IRB$V_SRCHGE] = 0;
;  1627     1687  4                IRAB[IRB$V_PRM] = 1;
;  1628     1688  4
;  1629     1689  4                ! If RMS has successfully positioned to a non-deleted primary data
;  1630     1690  4                ! record, return success
;  1631     1691  4                !
;  1632     1692  4                IF STATUS = RM$SEARCH_SIDR()
;  1633     1693  4                THEN
;  1634     1694  4                    RETURN .STATUS;
;  1635     1695  4
;  1636     1696  4                ! If RMS was unable to find a non-deleted primary data record with
;  1637     1697  4                ! an alternate key with the same value as the SIDR RMS has
;  1638     1698  4                ! positioned to, then move onto the next SIDR with a key greater
;  1639     1699  4                ! then that of the current SIDR.
;  1640     1700  4                !
```

```
;  1641          1701  5            IF .STATUS<0, 16> EQL RMSERR(RNF)
;  1642          1702  4            THEN
;  1643          1703  5                BEGIN
;  1644          1704  5                IRAB[IRB$V_SRCHGE] = 1;
;  1645          1705  5                STATUS = RM$CSEARCH_TREE();
;  1646          1706  5
;  1647          1707  5                ! If RMS is unable to find a SIDR on a greater or equal search
;  1648          1708  5                ! then this can only be because it has reached the
;  1649          1709  5                ! end-of-file, and so return the appropriate status.
;  1650          1710  5                !
;  1651          1711  6                IF .STATUS<0, 16> EQL RMSERR(RNF)
;  1652          1712  5                THEN
;  1653          1713  6                    STATUS = RMSERR(EOF)
;  1654          1714  6
;  1655          1715  6                ! Otherwise continue the search for a non-deleted primary data
;  1656          1716  6                ! record with the first array element of the next SIDR.
;  1657          1717  6                !
;  1658          1718  5                ELSE
;  1659          1719  5                    IRAB[IRB$W_SAVE_POS] = 0;
;  1660          1720  4                END;
;  1661          1721  4
;  1662          1722  3            END;
;  1663          1723  3
;  1664          1724  3        RETURN .STATUS;
;  1665          1725  3
;  1666          1726  2        END;
;  1667          1727  2
;  1668          1728  1    END;                                    ! { end of routine  }
```

```
                              34  BB 00000 RM$POS_SEQ::
                                            PUSHR   #^M<R2,R4,R5>                          ; 0961
                           41   A9  94 00002         CLRB    65(IRAB)                       ; 1106
                           42   A9  B4 00005         CLRW    66(IRAB)                       ; 1107
                        00A8    C9  D5 00008         TSTL    168(IRAB)                      ; 1113
                              04  12 0000C         BNEQ    1$
                        00AC    C9  D5 0000E         TSTL    172(IRAB)                      ; 1115
                              03  13 00012 1$:      BEQL    2$
                           009A   31 00014         BRW     15$
                     42    A9  40   8F  88 00017 2$:      BISB2   #64, 66(IRAB)          ; 1119
                           0000G  30 0001C         BSBW    RM$CSEARCH_TREE                ; 1120
                        0A         50  E8 0001F         BLBS    STATUS, 4$
                     82B2  8F       50  B1 00022         CMPW    STATUS, #33458
                                 2E  13 00027         BEQL    8$
                           0288   31 00029 3$:      BRW     56$
                        52   20    A9  9E 0002C 4$:      MOVAB   32(R9), R2             ; 1135
                        50         62  D0 00030 5$:      MOVL    (R2), R0
                        55   18    A0  D0 00033         MOVL    24(R0), BKT_ADDR
                             21    A7  95 00037         TSTB    33(IDX_DFN)            ; 1143
                                 04  12 0003A         BNEQ    6$
                  06         66       03  E0 0003C         BBS     #3, (REC_ADDR), 7$     ; 1145
                        0E   04    A5  B1 00040 6$:      CMPW    4(BKT_ADDR), #14       ; 1147
                                 59  1A 00044         BGTRU   13$
                        0F   0D    A5  E9 00046 7$:      BLBC    13(BKT_ADDR), 9$       ; 1155
```

```
                                54              50  D0 0004A          MOVL    R0, BDB                         ; 1158
                                                62  D4 0004D          CLRL    (R2)
                                                7E  D4 0004F          CLRL    -(SP)
                                            0000G   30 00051          BSBW    RM$RLSBKT
                                5E                  04  C0 00054      ADDL2   #4, SP
                                                    78  11 00057 8$:   BRB     16$                             ; 1159
                                51              08  A5  D0 00059 9$:   MOVL    8(BKT_ADDR), VBN                ; 1165
                                54              20  A9  D0 0005D      MOVL    32(IRAB), BDB                   ; 1166
                                                20  A9  D4 00061      CLRL    32(IRAB)
                                                    7E  D4 00064      CLRL    -(SP)
                                            0000G   30 00066          BSBW    RM$RLSBKT
                                5E                  04  C0 00069      ADDL2   #4, SP
                                06                  06  AA  E9 0006C  BLBC    6(IFAB), 10$                    ; 1173
                        05      00A0    CA          01  E0 00070      BBS     #1, 160(IFAB), 11$              ; 1175
                                                21  A7  95 00076 10$:  TSTB    33(IDX_DFN)                     ; 1177
                                                    04  13 00079      BEQL    12$
                                40      A9          01  90 0007B 11$:  MOVB    #1, 64(IRAB)                    ; 1179
                                50              17  A7  9A 0007F 12$:  MOVZBL  23(IDX_DFN), R0                ; 1181
                        7E              50          09  78 00083      ASHL    #9, R0, -(SP)
                                                    51  DD 00087      PUSHL   VBN
                                            0000G   30 00089          BSBW    RM$GETBKT
                                5E                  08  C0 0008C      ADDL2   #8, SP
                                97                  50  E9 0008F      BLBC    STATUS, 3$
                                52              20  A9  9E 00092      MOVAB   32(R9), R2                      ; 1183
                                62                  54  D0 00096      MOVL    BDB, (R2)
                                56              0E  A5  9E 00099      MOVAB   14(R5), REC_ADDR                ; 1184
                                                    91  11 0009D      BRB     5$                              ; 1125
                                42      A9      40  8F  8A 0009F 13$:  BICB2   #64, 66(IRAB)                  ; 1191
                                                21  A7  95 000A4      TSTB    33(IDX_DFN)                     ; 1199
                                                    03  12 000A7      BNEQ    14$
                                            0155    31 000A9          BRW     39$
                                                76  A9  B4 000AC 14$:  CLRW    118(IRAB)                       ; 1209
                                                    38  11 000AF      BRB     21$                             ; 1113
                                                21  A7  95 000B1 15$:  TSTB    33(IDX_DFN)                     ; 1219
                                                    36  13 000B4      BEQL    22$
                                  42    A9              10  88 000B6  BISB2   #16, 66(IRAB)                   ; 1226
                                            FE8D    30 000BA          BSBW    RM$FIND_SIDR                    ; 1234
                                51                  50  D0 000BD      MOVL    R0, STATUS
                                1E                  51  E8 000C0      BLBS    STATUS, 20$
                              8262      8F          51  B1 000C3      CMPW    STATUS, #33378                  ; 1236
                                                    14  13 000C8      BEQL    19$
                              82B2      8F          51  B1 000CA      CMPW    STATUS, #33458                  ; 1243
                                                    07  12 000CF      BNEQ    17$
                                50          827A    8F  3C 000D1 16$:  MOVZWL  #33402, R0                   ; 1247
                                                    03  11 000D6      BRB     18$
                                50                  51  D0 000D8 17$:  MOVL    STATUS, R0
                                            01D6    31 000DB 18$:     BRW     56$
                                08                  51  E9 000DE 19$:  BLBC    STATUS, 21$                     ; 1253
                        03      05      A9          03  E1 000E1 20$:  BBC     #3, 5(IRAB), 21$
                                                76  A9  B6 000E6      INCW    118(IRAB)                       ; 1255
                                            0162    31 000E9 21$:     BRW     49$                             ; 1219
                                                55  94 000EC 22$:     CLRB    FLAG                            ; 1281
                                0A                  06  AA  E9 000EE  BLBC    6(IFAB), 23$                    ; 1298
                        04      00A0    CA          01  E1 000F2      BBC     #1, 160(IFAB), 23$              ; 1300
                                  40    A9          01  90 000F8      MOVB    #1, 64(IRAB)                    ; 1302
                                5C              00B8    C9  3C 000FC 23$:  MOVZWL  184(IRAB), AP               ; 1304
                                                00A8    C9  DD 00101      PUSHL   168(IRAB)                       ; 1305
```

```
                                    0000G 30 00105          BSBW    RM$FIND_BY_RFA                          ;
                                 5E   04  C0 00108          ADDL2   #4, SP                                  ;
                                 11   50  E8 0010B          BLBS    STATUS, 25$                             ; 1310
                           82B2  8F   50  B1 0010E          CMPW    STATUS, #33458                          ; 1312
                                 07   13  00113          BEQL    24$                                     ;
                           8262  8F   50  B1 00115          CMPW    STATUS, #33378                          ; 1314
                                 BF   12  0011A          BNEQ    18$                                     ;
                                 04   50  E9 0011C 24$:     BLBC    STATUS, 26$                             ; 1323
                              30 66   03  E1 0011F 25$:     BBC     #3, (REC_ADDR), 30$                     ; 1325
                           00AC  C9   00A8 C9  D1 00123 26$:     CMPL    168(IRAB), 172(IRAB)                    ; 1342
                                 0B   12  0012A          BNEQ    27$                                     ;
                                 0D   50  E9 0012C          BLBC    STATUS, 28$                             ; 1355
                              09 66   03  E1 0012F          BBC     #3, (REC_ADDR), 28$                     ; 1357
                              05 66   02  E0 00133          BBS     #2, (REC_ADDR), 28$                     ; 1359
                                 55   01  88 00137 27$:     BISB2   #1, FLAG                                ; 1361
                                 03   11  0013A          BRB     29$                                     ;
                                 55   02  88 0013C 28$:     BISB2   #2, FLAG                                ; 1375
                                 47   50  E9 0013F 29$:     BLBC    STATUS, 32$                             ; 1381
                                 54   20  A9  D0 00142          MOVL    32(IRAB), BDB                           ; 1383
                                      20  A9  D4 00146          CLRL    32(IRAB)                                ;
                                      7E  D4 00149          CLRL    -(SP)                                   ;
                                    0000G 30 0014B          BSBW    RM$RLSBKT                               ;
                                 5E   04  C0 0014E          ADDL2   #4, SP                                  ;
                                 36   11  00151          BRB     32$                                     ; 1323
                                    FCCB 30 00153 30$:     BSBW    RM$CHECK_RRV                            ; 1409
                                 1B   50  E8 00156          BLBS    R0, 31$                                 ;
                                 54   20  A9  D0 00159          MOVL    32(IRAB), BDB                           ; 1416
                                      20  A9  D4 0015D          CLRL    32(IRAB)                                ;
                                      7E  D4 00160          CLRL    -(SP)                                   ;
                                    0000G 30 00162          BSBW    RM$RLSBKT                               ;
                                 5E   04  C0 00165          ADDL2   #4, SP                                  ;
                                 03   00B7 CA  91 00168          CMPB    183(IFAB), #3                           ; 1418
                                      69  1E 0016D          BGEQU   36$                                     ;
                                 55   01  88 0016F          BISB2   #1, FLAG                                ; 1422
                                      15  11 00172          BRB     32$                                     ; 1418
7E            05  A9                01  03  EF 00174 31$:     EXTZV   #3, #1, 5(IRAB), -(SP)                  ; 1435
                                 6E   6E  D2 0017A          MCOML   (SP), (SP)                              ;
                                    FCC6 30 0017D          BSBW    RM$FIND_NONDEL                          ;
                                 5E   04  C0 00180          ADDL2   #4, SP                                  ;
                                 03   50  E8 00183          BLBS    STATUS, 32$                             ;
                                    0083 31 00186          BRW     40$                                     ;
                                 60   55  E9 00189 32$:     BLBC    FLAG, 38$                               ; 1444
                                 0A   06  AA  E9 0018C          BLBC    6(IFAB), 33$                            ; 1455
                           04    00A0 CA  01  E1 00190          BBC     #1, 160(IFAB), 33$                      ; 1457
                                 40   A9  01  90 00196          MOVB    #1, 64(IRAB)                            ; 1459
                                      7E  D4 0019A 33$:     CLRL    -(SP)                                   ; 1463
                                 7E   00BA C9  3C 0019C          MOVZWL  186(IRAB), -(SP)                        ; 1464
                                      00AC C9  DD 001A1          PUSHL   172(IRAB)                               ; 1463
                                    0000G 30 001A5          BSBW    RM$FIND_BY_RRV                          ;
                                 5E   0C  C0 001A8          ADDL2   #12, SP                                 ;
                                 15   50  E8 001AB          BLBS    STATUS, 35$                             ; 1472
                           8262  8F   50  B1 001AE          CMPW    STATUS, #33378                          ; 1474
                                 06   13  001B3          BEQL    34$                                     ;
                              0C 8F   50  D0 001B5          MOVL    STATUS, 12(RAB)                         ; 1477
                                 1D   11  001B9          BRB     36$                                     ; 1478
                                 05   50  E8 001BB 34$:     BLBS    STATUS, 35$                             ; 1488
                                 55   02  88 001BE          BISB2   #2, FLAG                                ; 1490
```

```
                                                29  11  001C1            BRB     38$                                     ;
                                              FC5B  30  001C3 35$:       BSBW    RM$CHECK_RRV                            ; 1508
                                        11      50  E8  001C6            BLBS    R0, 37$                                 ;
                                        54  20  A9  D0  001C9            MOVL    32(IRAB), BDB                           ; 1515
                                            20  A9  D4  001CD            CLRL    32(IRAB)                                ;
                                                7E  D4  001D0            CLRL    -(SP)                                   ;
                                              0000G 30  001D2            BSBW    RM$RLSBKT                               ;
                                        5E      04  C0  001D5            ADDL2   #4, SP                                  ;
                                                68  11  001D8 36$:       BRB     46$                                     ;
                7E      05  A9            01      03  EF  001DA 37$:       EXTZV   #3, #1, 5(IRAB), -(SP)                  ; 1531
                                        6E      6E  D2  001E0            MCOML   (SP), (SP)                              ;
                                              FC60  30  001E3            BSBW    RM$FIND_NONDEL                          ;
                                        5E      04  C0  001E6            ADDL2   #4, SP                                  ;
                                        60      50  E9  001E9            BLBC    STATUS, 48$                             ;
                    5E                  55      01  E1  001EC 38$:       BBC     #1, FLAG, 49$                           ;
                                        1A  1C  A7  E8  001F0            BLBS    28(IDX_DFN), 41$                        ; 1550
                                    42  A9      10  88  001F4            BISB2   #16, 66(IRAB)                           ; 1557
                                              0000G 30  001F8            BSBW    RM$CSEARCH_TREE                         ; 1558
                                        03      50  E8  001FB            BLBS    STATUS, 39$                             ;
                                              0080  31  001FE            BRW     54$                                     ;
                                                01  DD  00201 39$:       PUSHL   #1                                      ; 1564
                                              FC40  30  00203            BSBW    RM$FIND_NONDEL                          ;
                                        5E      04  C0  00206            ADDL2   #4, SP                                  ;
                                        42      50  E8  00209            BLBS    STATUS, 49$                             ;
                                                6E  11  0020C 40$:       BRB     52$                                     ;
                                              0000G 30  0020E 41$:       BSBW    RM$CSEARCH_TREE                         ; 1582
                                        2E      50  E9  00211            BLBC    R0, 46$                                 ;
                                        54      01  D0  00214            MOVL    #1, STATUS                              ; 1607
                                              FC07  30  00217 42$:       BSBW    RM$CHECK_RRV                            ; 1617
                                        0D      50  E9  0021A            BLBC    R0, 43$                                 ;
                                                7E  D4  0021D            CLRL    -(SP)                                   ; 1620
                                              FC24  30  0021F            BSBW    RM$FIND_NONDEL                          ;
                                        5E      04  C0  00222            ADDL2   #4, SP                                  ;
                                        10      50  E8  00225            BLBS    STATUS, 44$                             ;
                                                52  11  00228            BRB     52$                                     ;
                                              0000G 30  0022A 43$:       BSBW    RM$GETNEXT_REC                          ; 1632
                                              0000G 30  0022D            BSBW    RM$CSEARCH_TREE                         ; 1634
                                        54      50  D0  00230            MOVL    R0, STATUS                              ;
                                        05      54  E9  00233            BLBC    STATUS, 45$                             ;
                                                DF  11  00236            BRB     42$                                     ;
                                        13      54  E8  00238 44$:       BLBS    STATUS, 49$                             ; 1642
                                82B2    8F      54  B1  0023B 45$:       CMPW    STATUS, #33458                          ; 1644
                                                07  12  00240            BNEQ    47$                                     ;
                                        50  8434    8F  3C  00242 46$:       MOVZWL  #33844, R0                          ; 1648
                                                6B  11  00247            BRB     56$                                     ;
                                        50          54  D0  00249 47$:       MOVL    STATUS, R0                              ;
                                                    66  11  0024C 48$:       BRB     56$                                     ;
                                            21      A7  95  0024E 49$:       TSTB    33(IDX_DFN)                             ; 1658
                                                    2B  12  00251            BNEQ    53$                                     ;
                                        03      00B7    CA  91  00253            CMPB    183(IFAB), #3                       ; 1661
                                                    06  1E  00258            BGEQU   50$                                     ;
                                        50          01  A6  9A  0025A            MOVZBL  1(REC_ADDR), R0                     ;
                                                    04  11  0025E            BRB     51$                                     ;
                                        50          01  A6  3C  00260 50$:       MOVZWL  1(REC_ADDR), R0                     ;
                                    74  A9              50  B0  00264 51$:       MOVW    R0, 116(IRAB)                       ;
                                        50      20  A9      D0  00268            MOVL    32(IRAB), R0                        ; 1662
                                    70  A9      1C  A0      D0  0026C            MOVL    28(R0), 112(IRAB)                   ;
```

```
                     50      20    A9  D0 00271          MOVL    32(IRAB), R0                        ; 1663
                  0A A0         08  88 00275          BISB2   #8, 10(R0)                          ;
                     50         01  D0 00279          MOVL    #1, R0                              ; 1672
                                36  11 0027C 52$:      BRB     56$                                 ;
                     50         01  D0 0027E 53$:      MOVL    #1, STATUS                          ; 1677
                     30         50  E9 00281 54$:      BLBC    STATUS, 56$                         ; 1683
                  42 A9         10  8A 00284          BICB2   #16, 66(IRAB)                       ; 1686
                  42 A9      80 8F  88 00288          BISB2   #128, 66(IRAB)                      ; 1687
                           0000G 30  0028D            BSBW    RM$SEARCH_SIDR                      ; 1692
                     21         50  E8 00290          BLBS    STATUS, 56$                         ;
                82B2 8F         50  B1 00293          CMPW    STATUS, #33458                      ; 1701
                                E7  12 00298          BNEQ    54$                                 ;
                  42 A9         10  88 0029A          BISB2   #16, 66(IRAB)                       ; 1704
                           0000G 30  0029E            BSBW    RM$CSEARCH_TREE                     ; 1705
                82B2 8F         50  B1 002A1          CMPW    STATUS, #33458                      ; 1711
                                07  12 002A6          BNEQ    55$                                 ;
                     50    827A 8F  3C 002A8          MOVZWL  #33402, STATUS                      ; 1713
                                D2  11 002AD          BRB     54$                                 ;
                           76   A9  B4 002AF 55$:      CLRW    118(IRAB)                           ; 1719
                                CD  11 002B2          BRB     54$                                 ; 1701
                                34  BA 002B4 56$:      POPR    #^M<R2,R4,R5>                       ; 1728
                                    05 002B6          RSB                                         ;

; Routine Size:  695 bytes,    Routine Base:  RM$RMS3 + 01DF

; 1669          1729  1
; 1670          1730  1 END
; 1671          1731  1
; 1672          1732  0 ELUDOM



;                                PSECT SUMMARY

;      Name                     Bytes                          Attributes

;  RM$RMS3                        1174  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        86         2       154         00:00.4


; Information:  2
; Warnings:     0
```

```
: Errors:       0


;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3POSSEQ/OBJ=OBJ$:RM3POSSEQ MSRC$:RM3POSSEQ/UPDATE=(ENH$:RM3POSSEQ)

; Size:         1174 code + 0 data bytes
; Run Time:       00:33.5
; Elapsed Time:   01:04.6
; Lines/CPU Min:   3104
; Lexemes/CPU-Min: 15804
; Memory Used:  274 pages
; Compilation Complete
```

RM3NEXTRE
LIS

RM3OPEN
LIS

RM3POSRFA
LIS

RM3PCKUNP
LIS

RM3POSKEY
LIS

RM3POSSEQ
LIS

RM3PROBE
LIS

RM3SIOXSP
LIS

RM3PUTERR
LIS

RM3PUTUPD
LIS

RM3SPLUDR
LIS

RM3RRV
LIS

RM3ROOT
LIS

RM3PUT
LIS