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

```
RRRRRRRR    MM       MM    333333      CCCCCCCC  RRRRRRRR    EEEEEEEEEE    AAAAAA    TTTTTTTTTT  EEEEEEEEEE
RRRRRRRR    MM       MM    333333      CCCCCCCC  RRRRRRRR    EEEEEEEEEE    AAAAAA    TTTTTTTTTT  EEEEEEEEEE
RR      RR  MMMM   MMMM  33      33  CC            RR      RR  EE          AA      AA      TT      EE
RR      RR  MMMM   MMMM  33      33  CC            RR      RR  EE          AA      AA      TT      EE
RR      RR  MM MM MM MM        33  CC            RR      RR  EE          AA      AA      TT      EE
RR      RR  MM MM MM MM        33  CC            RR      RR  EE          AA      AA      TT      EE
RRRRRRRR    MM    MM    33    CC            RRRRRRRR    EEEEEEEE    AA      AA      TT      EEEEEEEE
RRRRRRRR    MM    MM    33    CC            RRRRRRRR    EEEEEEEE    AA      AA      TT      EEEEEEEE
RR  RR      MM    MM        33  CC            RR  RR      EE          AAAAAAAAAA      TT      EE
RR  RR      MM    MM        33  CC            RR  RR      EE          AAAAAAAAAA      TT      EE
RR    RR    MM    MM  33    33  CC            RR    RR    EE          AA      AA      TT      EE                ....
RR    RR    MM    MM  33    33  CC            RR    RR    EE          AA      AA      TT      EE                ....
RR      RR  MM    MM    333333      CCCCCCCC  RR      RR  EEEEEEEEEE  AA      AA      TT      EEEEEEEEEE        ....
RR      RR  MM    MM    333333      CCCCCCCC  RR      RR  EEEEEEEEEE  AA      AA      TT      EEEEEEEEEE        ....


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 RM3CREATE (LANGUAGE (BLISS32) ,
;     2          0002  0                    IDENT = 'V04-000'
;     3          0003  0                    ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !*****************************************************************************
;     7          0007  1 !*                                                                           *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                     *
;    11          0011  1 !*                                                                           *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17          0017  1 !*  TRANSFERRED.                                                             *
;    18          0018  1 !*                                                                           *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    21          0021  1 !*  CORPORATION.                                                             *
;    22          0022  1 !*                                                                           *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*****************************************************************************
```

```
;    29          0028  1 !++
;    30          0029  1 !
;    31          0030  1 ! FACILITY:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
;    32          0031  1 !
;    33          0032  1 ! ABSTRACT:
;    34          0033  1 !               This module performs the $CREATE function for indexed
;    35          0034  1 !               sequential files.
;    36          0035  1 !
;    37          0036  1 ! ENVIRONMENT:
;    38          0037  1 !
;    39          0038  1 !               VAX/VMS OPERATING SYSTEM
;    40          0039  1 !--
;    41          0040  1
;    42          0041  1 !
;    43          0042  1 !
;    44          0043  1 ! AUTHOR:        Maria del C. Nasr      CREATION DATE:          21-Sep-1982
;    45          0044  1 !
;    46          0045  1 !
;    47          0046  1 ! MODIFIED BY:
;    48          0047  1 !
;    49          0048  1 !       V03-031 RAS0284         Ron Schaefer            30-Mar-1984
;    50          0049  1 !               Fix STV value on error paths for area allocation and
;    51          0050  1 !               RMS$_RPL and RMS$_WPL errors.
;    52          0051  1 !
;    53          0052  1 !       V03-030 SHZ0002         Stephen H. Zalewski,    27-Feb-1984
;    54          0053  1 !               If you allocate a BDB, you MUST increment the available
;    55          0054  1 !               local buffer count (IFB$W_AVLCL).
;    56          0055  1 !
;    57          0056  1 !       V03-029 SHZ0001         Stephen H. Zalewski     15-Dec-1983
;    58          0057  1 !               Correct error handling when doing XAB processing.
;    59          0058  1 !
;    60          0059  1 !       V03-028 KPL0001         Peter Lieberwirth       18-May-1983
;    61          0060  1 !               Initial pass at implementing the journaling of $CREATE.
;    62          0061  1 !
;    63          0062  1 !       V03-027 RAS0152         Ron Schaefer            29-Apr-1983
;    64          0063  1 !               Update MCN0012 to remove the warning message.
;    65          0064  1 !               Turn the XAB reference from a GLOBAL REGISTER to a
;    66          0065  1 !               REGISTER parameter.
;    67          0066  1 !
;    68          0067  1 !       V03-026 MCN0012         Maria del C. Nasr       07-Apr-1983
;    69          0068  1 !               Modify calling sequence to RM$EXTENDO to match new linkage.
;    70          0069  1 !
;    71          0070  1 !       V03-025 MCN0011         Maria del C. Nasr       24-Mar-1983
;    72          0071  1 !               Change linkages for the GETSPC and RETSPC routines among
;    73          0072  1 !               others.  Also get rid of prologue and journaling check
;    74          0073  1 !               introduced by MCN0007 since it is not useful.
;    75          0074  1 !
;    76          0075  1 !       V03-024 MCN0010         Maria del C. Nasr       21-Mar-1983
;    77          0076  1 !               Fix some local linkages that were incorrectly defined
;    78          0077  1 !               by MCN0009.
;    79          0078  1 !
;    80          0079  1 !       V03-023 MCN0009         Maria del C. Nasr       07-Mar-1983
;    81          0080  1 !               Reorganize linkages.
;    82          0081  1 !
;    83          0082  1 !       V03-022 MCN0009         Maria del C. Nasr       07-Mar-1983
;    84          0083  1 !               When MRS and bucket size are not specified, RMS should
;    85          0084  1 !               use minimum record size to calculate bucket size of data
```

```
;   86          0085  1 !               level.  Also, change maximum bucket size to 65 and use
;   87          0086  1 !               symbolic name for it.
;   88          0087  1 !
;   89          0088  1 !       V03-021 DAS0001         David Solomon           01-Feb-1983
;   90          0089  1 !               Add support for 64-bit binary keys (IN8 and BN8); namely,
;   91          0090  1 !               add a case-value in routine CHECK_KEY_PARMS to validate
;   92          0091  1 !               the key size.
;   93          0092  1 !
;   94          0093  1 !       V03-020 MCN0008         Maria del C. Nasr       31-Jan-1983
;   95          0094  1 !               Under certain error conditions RM$CREATECOM exits RMS
;   96          0095  1 !               without returning status to RM$CREATE3B.  RMS then bugchecks
;   97          0096  1 !               because the space allocated for the key and area tables is
;   98          0097  1 !               not returned.  Fix code so that this condition is eliminated.
;   99          0098  1 !               Allocate table, but copy information back to XAB so that it
;  100          0099  1 !               can be returned before calling RM$CREATECOM.  Unfortunately,
;  101          0100  1 !               this change requires extra passes through the XAB chain.
;  102          0101  1 !
;  103          0102  1 !       V03-019 MCN0007         Maria del C. Nasr       25-Jan-1983
;  104          0103  1 !               Do not create the file if the prologue is 1 or 2 and
;  105          0104  1 !               any of the journal flags in the IFAB are set.
;  106          0105  1 !
;  107          0106  1 !       V03-018 KBT0461         Keith B. Thompson       13-Jan-1983
;  108          0107  1 !               Allocate a BDB and buffer for reading in the prologue
;  109          0108  1 !
;  110          0109  1 !       V03-017 MCN0006         Maria del C. Nasr       02-Dec-1982
;  111          0110  1 !               In main module, check the device type if file created
;  112          0111  1 !               with block I/O set.  If device is random access, continue
;  113          0112  1 !               to write prologue; otherwise, return.  This will correct the
;  114          0113  1 !               creation of a file with invalid prologue when all XABs were
;  115          0114  1 !               correctly defined.  If no XABs were given, an invalid file
;  116          0115  1 !               will still be created to support the COPY program.
;  117          0116  1 !
;  118          0117  1 !       V03-016 MCN0005         Maria del C. Nasr       01-Dec-1982
;  119          0118  1 !               Routine EXTEND:  Set the TOTAL_ALLOC field in the area
;  120          0119  1 !               descriptor for each area as it is extended.
;  121          0120  1 !
;  122          0121  1 !       V03-015 MCN0004         Maria del C. Nasr       18-Nov-1982
;  123          0122  1 !               Fix more bucket size problems caused by COPY since it does
;  124          0123  1 !               not define any keys, but it sets up area XABs.
;  125          0124  1 !
;  126          0125  1 !       V03-014 MCN0003         Maria del C. Nasr       15-Nov-1982
;  127          0126  1 !               - Do not change the FAB_BKS value if there are no area XABs.
;  128          0127  1 !               - If a given area is not related to a key or data bucket, and
;  129          0128  1 !               the bucket size is zero, return bucket size error.
;  130          0129  1 !               - If no area XABs, make sure that at least two keys fit in the
;  131          0130  1 !               bucket size specified in the FAB, if any.
;  132          0131  1 !               - Set the FAB and IFAB bucket size values to the largest bucket
;  133          0132  1 !               size in the file, not the primary data level size.
;  134          0133  1 !               - Call CHECK_MRS for the FAB only when we know that there are
;  135          0134  1 !               no area XABs.
;  136          0135  1 !               - Place the FWA address in R10 again, if RMSCREATCOM fails.
;  137          0136  1 !
;  138          0137  1 !       V03-013 MCN0002         Maria del C. Nasr       11-Nov-1982
;  139          0138  1 !               Delay the optimization of the bucket size for the data bucket
;  140          0139  1 !               (forcing it to be at least 2), after we have determined if the
;  141          0140  1 !               user gave a bucket size in an allocation XAB.  In that way, we
;  142          0141  1 !               will not return a BKZ error if he specified a value of 1.
```

```
;  143     0142  1 !
;  144     0143  1 !       V03-012 MCN0001         Maria del C. Nasr       11-Nov-1982
;  145     0144  1 !               If there are no key XABs and the file is being created with
;  146     0145  1 !               block I/O set, do not return error.  This is done so that
;  147     0146  1 !               COPYing of indexed files works, since no key XABs are defined
;  148     0147  1 !               for the output file.
;  149     0148  1 !
;  150     0149  1 !
;  151     0150  1 !*****
;  152     0151  1
;  153     0152  1 LIBRARY 'RMSLIB:RMS';
;  154     0153  1
;  155     0154  1 REQUIRE 'RMSSRC:RMSIDXDEF';
;  156     0219  1
;  157     0220  1 ! define default psects for code
;  158     0221  1 !
;  159     0222  1 PSECT
;  160     0223  1     CODE = RM$RMSMISC(PSECT_ATTR),
;  161     0224  1     PLIT = RM$RMSMISC(PSECT_ATTR);
;  162     0225  1
;  163     0226  1 ! External Linkages
;  164     0227  1 !
;  165     0228  1 LINKAGE
;  166     0229  1     L_ALDBUF,
;  167     0230  1     L_CACHE,
;  168     0231  1     L_EXTENDO,
;  169     0232  1     L_FABREG,
;  170     0233  1     L_GETSPC,
;  171     0234  1     L_LINK_7_10_11,
;  172     0235  1     L_CHKSUM,
;  173     0236  1     L_RELEASE_FAB,
;  174     0237  1     L_RETSPC;
;  175     0238  1
;  176     0239  1 ! Local linkage definitions
;  177     0240  1 !
;  178     0241  1 LINKAGE
;  179     0242  1     RL$AL_KEY           = JSB () : GLOBAL (COMMON_FABREG),
;  180     0243  1     RL$CHECK_DENSE      = JSB (),
;  181     0244  1     RL$CHECK_INPUT      = JSB (;REGISTER = 7) :
;  182     0245  1                           GLOBAL (COMMON_FABREG, XAB = 6),
;  183     0246  1     RL$CHECK_KEY_PARMS  = JSB (REGISTER = 1, REGISTER = 2, REGISTER = 3) :
;  184     0247  1                           GLOBAL (XAB = 6, R_IFAB)
;  185     0248  1                           NOTUSED (9, 11),
;  186     0249  1     RL$CHECK_OVERLAP    = JSB (REGISTER = 3) : GLOBAL (XAB = 6),
;  187     0250  1     RL$CHECK_PROLOG     = JSB (REGISTER = 0, REGISTER = 3; REGISTER = 7) :
;  188     0251  1                           GLOBAL (XAB = 6, R_FAB, R_IFAB),
;  189     0252  1     RL$CHK_AND_REPROBE  = JSB () : GLOBAL (R_FAB, R_IFAB, XAB = 6),
;  190     0253  1     RL$CLEAR            = JSB (),
;  191     0254  1     RL$EXTEND           = JSB (REGISTER = 7, REGISTER=6) :
;  192     0255  1                           GLOBAL (COMMON_FABREG)  NOPRESERVE (2),
;  193     0256  1     FL$FIND_PROLOGUE    = JSB (REGISTER = 3; REGISTER = 7) :
;  194     0257  1                           GLOBAL (XAB = 6, R_FAB, R_IFAB),
;  195     0258  1     RL$GET_BKS          = JSB (;REGISTER = 1) :
;  196     0259  1                           GLOBAL (R_FAB, R_IFAB, XAB = 6),
;  197     0260  1     RL$GET_KEY_PARMS    = JSB (;REGISTER = 1, REGISTER = 2, REGISTER = 3) :
;  198     0261  1                           GLOBAL (XAB = 6)
;  199     0262  1                           NOTUSED (9, 10, 11),
```

```
;  200          0263  1      RL$RW_PROLOG          = JSB () : GLOBAL (COMMON_FABREG, R_BDB),
;  201          0264  1      RL$WRITE_KEY_DESC     = JSB () : GLOBAL (R_IFAB, R_FAB, XAB = 6, BUF = 7);
;  202          0265  1
;  203          0266  1  ! Forward Routines
;  204          0267  1  !
;  205          0268  1
;  206          0269  1  FORWARD ROUTINE
;  207          0270  1      RM$CREATE3B           : RL$FABREG,
;  208          0271  1      AL_KEY_DESC           : RL$AL_KEY,
;  209          0272  1      CHECK_AREAS           : RL$CHK_AND_REPROBE,
;  210          0273  1      CHECK_DENSE           : RL$CHECK_DENSE,
;  211          0274  1      CHECK_INPUT           : RL$CHECK_INPUT,
;  212          0275  1      CHECK_KEY_PARMS       : RL$CHECK_KEY_PARMS,
;  213          0276  1      CHECK_MRS             : RL$CHK_AND_REPROBE,
;  214          0277  1      CHECK_OVERLAP         : RL$CHECK_OVERLAP,
;  215          0278  1      CHECK_PROLOG          : RL$CHECK_PROLOG,
;  216          0279  1      CLEAR                 : RL$CLEAR_NOVALUE,
;  217          0280  1      EXTEND                : RL$EXTEND,
;  218          0281  1      FIND_PROLOGUE         : RL$FIND_PROLOGUE,
;  219          0282  1      GET_BKS               : RL$GET_BKS,
;  220          0283  1      GET_KEY_PARMS         : RL$GET_KEY_PARMS NOVALUE,
;  221          0284  1      MOVE                  : RL$WRITE_KEY_DESC,
;  222          0285  1      READ_PROLOGUE         : RL$RW_PROLOG,
;  223          0286  1      REPROBE               : RL$CHK_AND_REPROBE,
;  224          0287  1      WRITE_KEY_DESC        : RL$WRITE_KEY_DESC,
;  225          0288  1      WRITE_PROLOGUE        : RL$RW_PROLOG;
;  226          0289  1
;  227          0290  1  ! External Routines
;  228          0291  1  !
;  229          0292  1  EXTERNAL ROUTINE
;  230          0293  1      RM$AL_KEY_DESC        : RL$LINK_7_10_11 ADDRESSING_MODE(GENERAL),
;  231          0294  1      RM$ALBLB              : RL$LINK_7_10_11 ADDRESSING_MODE(GENERAL),
;  232          0295  1      RM$ALDBUF             : RL$ALDBUF     ADDRESSING_MODE( GENERAL ),
;  233          0296  1      RM$CACHE              : RL$CACHE ADDRESSING_MODE(GENERAL),
;  234          0297  1      RM$CREATECOM          : RL$FABREG ADDRESSING_MODE(GENERAL),
;  235          0298  1      RM$EXTENDO            : RL$EXTENDO ADDRESSING_MODE(GENERAL),
;  236          0299  1      RM$GETSPC             : RL$GETSPC ADDRESSING_MODE(GENERAL),
;  237          0300  1      RM$MAKSUM             : RL$CHKSUM ADDRESSING_MODE(GENERAL),
;  238          0301  1      RM$RELEASE            : RL$RELEASE_FAB ADDRESSING_MODE(GENERAL),
;  239          0302  1      RM$RETSPC             : RL$RETSPC ADDRESSING_MODE(GENERAL) NOVALUE,
;  240          0303  1      RM$RND_DEV            : RL$LINK_7_10_11 ADDRESSING_MODE(GENERAL);
;  241          0304  1
;  242          0305  1  EXTERNAL
;  243          0306  1      PIO$GB_RMSPROLOG: ADDRESSING_MODE (GENERAL),        ! These symbols are defined in the SHELL
;  244          0307  1      SYS$GB_RMSPROLOG: ADDRESSING_MODE (GENERAL);        ! These symbols are defined in SYSPARAM
;  245          0308  1
```

```
;  247          0309  1 %SBTTL  'RM$CREATE3B'
;  248          0310  1 GLOBAL ROUTINE RM$CREATE3B : RL$FABREG =
;  249          0311  1
;  250          0312  1 !++
;  251          0313  1 !
;  252          0314  1 !  RM$CREATE3B
;  253          0315  1 !
;  254          0316  1 !       This routine checks the input (XABs), creates the file,
;  255          0317  1 !       fills in key descriptor, allocates index descriptor, and writes
;  256          0318  1 !       the prologue.
;  257          0319  1 !
;  258          0320  1 ! CALLING SEQUENCE:
;  259          0321  1 !
;  260          0322  1 !       RM$CREATE3B()
;  261          0323  1 !
;  262          0324  1 ! INPUT PARAMETERS:
;  263          0325  1 !
;  264          0326  1 !       FAB     - address of user's file access block
;  265          0327  1 !       IFAB_FILE - address of internal RMS file access block
;  266          0328  1 !
;  267          0329  1 ! IMPLICIT INPUTS:
;  268          0330  1 !       None
;  269          0331  1 !
;  270          0332  1 ! IMPLICIT OUTPUTS:
;  271          0333  1 !       None
;  272          0334  1 !
;  273          0335  1 ! ROUTINE VALUE:
;  274          0336  1 !
;  275          0337  1 !       RFM,BKS,IMX,REF,FLG,DTP,SIZ,POS,NPK,XAB,IAN,LAN,DAN,IFL,DFL,
;  276          0338  1 !       BKZ,PLV,MRS,SEG,KNM,AID,IBK,KSI,and SUC
;  277          0339  1 !
;  278          0340  1 ! SIDE EFFECTS:
;  279          0341  1 !       File is created
;  280          0342  1 !
;  281          0343  1 !--
;  282          0344  1
;  283          0345  2     BEGIN
;  284          0346  2
;  285          0347  2     ! These macros are defined to make code more readable.  They are
;  286          0348  2     ! executed when an error occurs, and buffers must be released.
;  287          0349  2     !
;  288          0350  2     MACRO
;  289        M 0351  2         RETURN_VBN1 =
;  290        M 0352  2             BEGIN
;  291        M 0353  2             IF .BDB_VBN1 EQL 0
;  292        M 0354  2               THEN BDB = .BLB_VBN1
;  293        M 0355  2               ELSE BDB = .BDB_VBN1;
;  294        M 0356  2             RM$RELEASE (0);
;  295          0357  2             END; %,
;  296          0358  2
;  297        M 0359  2         RETURN_ALL_BUFS =
;  298        M 0360  2             BEGIN
;  299        M 0361  2             IF .BDB EQL .BDB_VBN1
;  300        M 0362  2             THEN
;  301        M 0363  2                 RM$RELEASE (0)
;  302        M 0364  2             ELSE
;  303        M 0365  2                 BEGIN
```

```
;  304        M 0366  2                RM$RELEASE (0);
;  305        M 0367  2                RETURN_VBN1;
;  306        M 0368  2                END;
;  307           0369  2            END; %;
;  308           0370  2
;  309           0371  2      LOCAL
;  310           0372  2          CRE_STATUS;
;  311           0373  2
;  312           0374  2      EXTERNAL REGISTER
;  313           0375  2          COMMON_FAB_STR;
;  314           0376  2
;  315           0377  2      GLOBAL REGISTER
;  316           0378  2          BUF = 7 : REF BBLOCK,
;  317           0379  2          XAB = 6 : REF BBLOCK;
;  318           0380  2
;  319           0381  2      ! Set up IFAB in R10, but have to save the FWA address for RM$CREATECOM
;  320           0382  2      !
;  321           0383  3      BEGIN
;  322           0384  3
;  323           0385  3      LOCAL
;  324           0386  3          SPACE_ADDR,
;  325           0387  3          FWA;
;  326           0388  3
;  327           0389  3      LITERAL
;  328           0390  3          TABLE_SIZE = 64;
;  329           0391  3
;  330           0392  3      FWA = .IFAB;
;  331           0393  3      IFAB = .IFAB_FILE;
;  332           0394  3
;  333           0395  3      ! Get space for key and area bit masks
;  334           0396  3      !
;  335           0397  3
;  336           0398  3      RETURN_ON_ERROR ( RM$GETSPC (.IFAB, TABLE_SIZE; SPACE_ADDR) );
;  337           0399  3
;  338           0400  3      ! Call CHECK_INPUT to verify that the XABs are correct.
;  339           0401  3      !
;  340           0402  4      BEGIN
;  341           0403  4
;  342           0404  4      LOCAL
;  343           0405  4          PROLOGUE,
;  344           0406  4          STATUS;
;  345           0407  4
;  346           0408  4      STATUS = CHECK_INPUT ( .SPACE_ADDR; PROLOGUE );
;  347           0409  4
;  348           0410  4      ! Before checking STATUS, return space used for bits masks
;  349           0411  4      !
;  350           0412  4
;  351           0413  4      RM$RETSPC ( TABLE_SIZE, .IFAB, .SPACE_ADDR );
;  352           0414  4
;  353           0415  4      ! If CHECK_INPUT failed, return error.  Otherwise, save
;  354           0416  4      ! prologue version, and continue processing.
;  355           0417  4      !
;  356           0418  4
;  357           0419  4      IF NOT .STATUS
;  358           0420  4      THEN
;  359           0421  4          RETURN .STATUS;
;  360           0422  4
```

```
;  361          0423  4      IFAB [IFB$B_PLG_VER] = .PROLOGUE;
;  362          0424  4
;  363          0425  3      END;                            ! end of STATUS and PROLOGUE definition
;  364          0426  3
;  365          0427  3      ! Get space for key and area tables
;  366          0428  3      !
;  367          0429  3
;  368          0430  4      BEGIN
;  369          0431  4
;  370          0432  4      LOCAL
;  371          0433  4          SIZE,
;  372          0434  4          STATUS;
;  373          0435  4
;  374          0436  4      SIZE = ( 8 * .IFAB [IFB$B_NUM_KEYS] ) + .IFAB [IFB$B_AMAX];
;  375          0437  4
;  376          0438  4      IF .SIZE LSS 12
;  377          0439  4      THEN
;  378          0440  4          SIZE = 12;
;  379          0441  4      RETURN_ON_ERROR ( RMSGETSPC (.IFAB, .SIZE; SPACE_ADDR) );
;  380          0442  4
;  381          0443  4      ! Call routine to check area information against key XABs, and
;  382          0444  4      ! return space used by tables.
;  383          0445  4      !
;  384          0446  4
;  385          0447  4      STATUS = CHECK_AREAS ( .SPACE_ADDR );
;  386          0448  4
;  387          0449  4      RMSRETSPC ( .SIZE, .IFAB, .SPACE_ADDR );
;  388          0450  4
;  389          0451  4      ! If CHECK_AREAS failed, return error.
;  390          0452  4      !
;  391          0453  4
;  392          0454  4      IF NOT .STATUS
;  393          0455  4      THEN
;  394          0456  4          RETURN .STATUS;
;  395          0457  3      END;                                ! end of SIZE, STATUS def
;  396          0458  3
;  397          0459  3      ! Actually create the file on the disk
;  398          0460  3      !
;  399          0461  3      IFAB = .FWA;                        ! restore FWA adddress for call
;  400          0462  3      CRE_STATUS = RMSCREATECOM();
;  401          0463  3
;  402          0464  3      IF  NOT .CRE_STATUS
;  403          0465  4        OR ( .FAB [FAB$V_CIF]
;  404          0466  4          AND  .CRE_STATUS EQL RMSSUC() )
;  405          0467  3      THEN
;  406          0468  3          RETURN .CRE_STATUS;
;  407          0469  3
;  408          0470  2      END;                                ! end of FWA and SPACE_ADDR definition
;  409          0471  2
;  410          0472  2      ! From now on, we need the IFAB address in R10
;  411          0473  2      !
;  412          0474  2      IFAB = .IFAB_FILE;
;  413          0475  2
;  414          0476  2      ! If block I/O, and device not random, then just return.  Otherwise,
;  415          0477  2      ! continue to write prologue.
;  416          0478  2      !
;  417          0479  2
```

```
;  418     0480  3     BEGIN
;  419     0481  3
;  420     0482  3     GLOBAL REGISTER
;  421     0483  3         R_IDX_DFN;
;  422     0484  3
;  423     0485  3     IF .FAB [FAB$V_BIO]
;  424     0486  3       AND NOT RM$RND_DEV ()
;  425     0487  3     THEN
;  426     0488  3         RETURN .CRE_STATUS;
;  427     0489  3
;  428     0490  2     END;
;  429     0491  2
;  430     0492  3     BEGIN
;  431     0493  3
;  432     0494  3     GLOBAL REGISTER
;  433     0495  3         R_BDB_STR;
;  434     0496  3
;  435     0497  4     BEGIN
;  436     0498  4
;  437     0499  4     ! Allocate a BDB and a 1 block buffer for reading in the prologue
;  438     0500  4     ! More if either AI or BI journaling, see discussion in RM3CONN.
;  439     0501  4     !
;  440     0502  4     LOCAL
;  441     0503  4         BKS,                            ! max bucket size
;  442     0504  4         BUFSIZ,                         ! size of buffer to allocate
;  443     0505  4         JBDB:   REF BLOCK[,BYTE],       ! pointer to jnlbdb
;  444     0506  4         DBDB:   REF BLOCK[,BYTE];       ! pointer to data bdb
;  445     0507  4
;  446     0508  4     BUFSIZ = BKS = 512;
;  447     0509  4
;  448     0510  4     IF .IFAB[IFB$V_AI]
;  449     0511  4     THEN
;  450     0512  4         BUFSIZ = .BUFSIZ + RJR$C_BKTLEN + BDB$C_BLN;
;  451     0513  4
;  452     0514  4     IF .IFAB[IFB$V_BI]
;  453     0515  4     THEN
;  454     0516  4         BUFSIZ = .BUFSIZ + .BKS + RJR$C_BKTLEN + BDB$C_BLN;
;  455     0517  4
;  456     0518  4     RETURN_ON_ERROR( RM$ALDBUF( .BUFSIZ ) );    ! Allocate a BDB.
;  457     0519  4     IFAB[IFB$W_AVLCL] = .IFAB[IFB$W_AVLCL] + 1; ! Increment the local buffer count.
;  458     0520  4
;  459     0521  4     ! The new BDB is allocated onto the end of the BDB chain.  Now fill in the
;  460     0522  4     ! BDB to describe the journaling fields and buffers.
;  461     0523  4     !
;  462     0524  4     DBDB = .IFAB[IFB$L_BDB_BLNK];
;  463     0525  4
;  464     0526  4     DBDB[BDB$W_SIZE] = .BKS;
;  465     0527  4
;  466     0528  4     IF .IFAB[IFB$V_AI]
;  467     0529  4     THEN
;  468     0530  5         BEGIN
;  469     0531  5         JBDB = .DBDB[BDB$L_ADDR];
;  470     0532  5         DBDB[BDB$L_AI_BDB] = .JBDB;
;  471     0533  5
;  472     0534  5         JBDB[BDB$B_BID] = BDB$C_BID;
;  473     0535  5         JBDB[BDB$B_BLN] = BDB$C_BLN/4;
;  474     0536  5         JBDB[BDB$L_FLINK] = .JBDB;
```

```
;  475     0537  5          JBDB[BDB$L_BLINK] = .JBDB;
;  476     0538  5          JBDB[BDB$W_SIZE] = .BKS + RJR$C_BKTLEN;
;  477     0539  5          JBDB[BDB$L_ADDR] = .JBDB + BDB$C_BLN;
;  478     0540  5          DBDB[BDB$L_ADDR] = .JBDB + BDB$C_BLN + RJR$C_BKTLEN;
;  479     0541  4          END;
;  480     0542  4
;  481     0543  4      IF .IFAB[IFB$V_BI]
;  482     0544  4      THEN
;  483     0545  5          BEGIN
;  484     0546  5          JBDB = .DBDB[BDB$L_ADDR] + .BKS;
;  485     0547  5          DBDB[BDB$L_BI_BDB] = .JBDB;
;  486     0548  5
;  487     0549  5          JBDB[BDB$B_BID] = BDB$C_BID;
;  488     0550  5          JBDB[BDB$B_BLN] = BDB$C_BLN/4;
;  489     0551  5          JBDB[BDB$L_FLINK] = .JBDB;
;  490     0552  5          JBDB[BDB$L_BLINK] = .JBDB;
;  491     0553  5          JBDB[BDB$W_SIZE] = .BKS + RJR$C_BKTLEN;
;  492     0554  5          JBDB[BDB$L_ADDR] = .JBDB + BDB$C_BLN;
;  493     0555  4          END;
;  494     0556  4
;  495     0557  3      END;
;  496     0558  3
;  497     0559  3      ! One BLB is allocated as part of a shared open if necessary.  If
;  498     0560  3      ! locking is required, a lock BLB must be allocated now.
;  499     0561  3      !
;  500     0562  3      IF NOT .IFAB [IFB$V_NORECLK]
;  501     0563  3      THEN
;  502     0564  4          BEGIN
;  503     0565  4
;  504     0566  4          GLOBAL REGISTER
;  505     0567  4              R_IDX_DFN;
;  506     0568  4
;  507     0569  4          RETURN_ON_ERROR( RM$ALBLB() );
;  508     0570  3          END;
;  509     0571  3
;  510     0572  3      ! Define and initialize some loca' variables.
;  511     0573  3      !
;  512     0574  4      BEGIN
;  513     0575  4                                                                    ;
;  514     0576  4      LOCAL
;  515     0577  4          VBN,                                ! VBN number
;  516     0578  4          DISP,                               ! offset into VBN
;  517     0579  4          LAST_KEY_VBN,                       ! will be used to calculate where
;  518     0580  4                                              ! first area descriptor goes in plg
;  519     0581  4          COMPR : BYTE,                       ! compression flag
;  520     0582  4          ALLOC : BYTE,                       ! alloc XAB found flag
;  521     0583  4          XAB_TYP : BYTE,                     ! XAB type
;  522     0584  4          BDB_VBN1,                           ! stores BDB address of VBN 1
;  523     0585  4          BLB_VBN1,                           ! stores BLB address of VBN 1
;  524     0586  4          AREA0_SIZE;                         ! total size allocated to area 0
;  525     0587  4
;  526     0588  4      COMPR = 0;
;  527     0589  4      ALLOC = 0;
;  528     0590  4      BDB_VBN1 = 0;
;  529     0591  4      BLB_VBN1 = 0;
;  530     0592  4      LAST_KEY_VBN = ( .IFAB [IFB$B_NUM_KEYS] + 3 ) / 5 + 1;
;  531     0593  4      AREA0_SIZE = .IFAB [IFB$L_HBK];
```

```
;  533          0594  4     ! If we are sharing this file, then get a lock on VBN 1 to keep everyone
;  534          0595  4     ! else out while we write the prologue.  It will remain locked for the
;  535          0596  4     ! rest of the file creation.  The BLB address is saved in BLB_VBN1 so we
;  536          0597  4     ! can return the lock if errors occur later.
;  537          0598  4
;  538          0599  4     IF NOT .IFAB [IFB$V_NORECLK]
;  539          0600  4     THEN
;  540          0601  5         BEGIN
;  541          0602  5
;  542          0603  5         GLOBAL REGISTER
;  543          0604  5             R_BKT_ADDR_STR;
;  544          0605  5
;  545        P 0606  5         RETURN_ON_ERROR (RMSCACHE(1,512,CSH$M_NOREAD OR CSH$M_LOCK OR CSH$M_NOBUFFER),
;  546          0607  5                          STATUS = RMSERR (ENQ) );
;  547          0608  5         BLB_VBN1 = .BDB;                        ! BLB returned instead of BDB.
;  548          0609  4         END;
;  549          0610  4
;  550          0611  4     ! Get start of XAB chain, and start scanning
;  551          0612  4     !
;  552          0613  4     XAB = .FAB [FAB$L_XAB];
;  553          0614  4
;  554          0615  4     WHILE 1
;  555          0616  4     DO
;  556          0617  4
;  557          0618  5         BEGIN
;  558          0619  5
;  559          0620  5         ! If no more XABs, check if we found an allocation XAB.  If we did,
;  560          0621  5         ! exit loop.  Else, set the XAB type to ALL, so that we force at least
;  561          0622  5         ! one alloc XAB definition.
;  562          0623  5         !
;  563          0624  5
;  564          0625  5         IF .XAB EQL 0
;  565          0626  5         THEN
;  566          0627  6             BEGIN
;  567          0628  6
;  568          0629  6             IF .ALLOC
;  569          0630  6             THEN
;  570          0631  6                 EXITLOOP
;  571          0632  6             ELSE
;  572          0633  6                 XAB_TYP = XAB$C_ALL;
;  573          0634  6             END
;  574          0635  5         ELSE
;  575          0636  5
;  576          0637  5             ! If this is a valid XAB, reprobe it and save the type.
;  577          0638  5             !
;  578          0639  6             BEGIN
;  579          0640  6             RETURN_ON_ERROR ( REPROBE () );
;  580          0641  6             XAB_TYP = .XAB [XAB$B_COD];
;  581          0642  5             END;
```

```
;  583          0643  5          CASE .XAB_TYP FROM XAB$C_DAT TO XAB$C_TRM OF
;  584          0644  5              SET
;  585          0645  5
;  586          0646  5              [XAB$C_KEY] :
;  587          0647  5
;  588          0648  6                  BEGIN
;  589          0649  6
;  590          0650  6                  ! Calculate the VBN and offset where this key descriptor
;  591          0651  6                  ! will go in the prologue based on the key of reference.
;  592          0652  6                  !
;  593          0653  6
;  594          0654  6                  IF .XAB [XAB$B_REF] EQL 0
;  595          0655  6                  THEN
;  596          0656  7                      BEGIN
;  597          0657  7                      VBN = 1;
;  598          0658  7                      DISP = 0;
;  599          0659  7                      END
;  600          0660  6                  ELSE
;  601          0661  7                      BEGIN
;  602          0662  7                      VBN = ( .XAB [XAB$B_REF] + 4 ) / 5 + 1;
;  603          0663  8                      DISP = ( ( .XAB [XAB$B_REF] - 1 ) MOD 5 )
;  604          0664  7                                          * ( KEY$C_BLN + KEY$C_SPARE );
;  605          0665  6                      END;
;  606          0666  6
;  607        P 0667  6                  RETURN_ON_ERROR ( READ_PROLOGUE (.VBN),
;  608          0668  6                                    RETURN_VBN1 );
;  609          0669  6
;  610          0670  6                  ! If VBN 1 is read, save its BDB address in BDB_VBN1 so that
;  611          0671  6                  ! if we fail reading or writing the prologue, this locked VBN
;  612          0672  6                  ! can be released.
;  613          0673  6                  !
;  614          0674  6                  IF .VBN EQL 1
;  615          0675  6                  THEN
;  616          0676  6                      BDB_VBN1 = .BDB;
;  617          0677  6
;  618          0678  6                  ! Set the starting address of the key descriptor and call the
;  619          0679  6                  ! routine which fills in the fields.  If in error, must release
;  620          0680  6                  ! all buffers that have been locked.
;  621          0681  6                  !
;  622          0682  6                  BUF = .BDB [BDB$L_ADDR] + .DISP;
;  623          0683  6
;  624        P 0684  6                  RETURN_ON_ERROR ( WRITE_KEY_DESC (),
;  625          0685  6                                    RETURN_ALL_BUFS );
;  626          0686  6
;  627          0687  6                  ! Allocate in_core index descriptor
;  628          0688  6                  !
;  629        P 0689  6                  RETURN_ON_ERROR ( AL_KEY_DESC ( .BUF, .VBN, .DISP ),
;  630          0690  6                                    RETURN_ALL_BUFS );
;  631          0691  6
;  632          0692  6                  ! Write the VBN we just filled.  If sharing and we are writing
;  633          0693  6                  ! VBN 1 then clear bdb as it will be released when we write
;  634          0694  6                  ! the buffer.  Clearing bdb before the write is okay, because
;  635          0695  6                  ! if an error occurs, all we can do is release the lock.
;  636          0696  6                  !
;  637          0697  6                  IF NOT .IFAB [IFB$V_NORECLK]
;  638          0698  6                     AND .BDB [BDB$L_VBN] EQL 1
;  639          0699  6                  THEN
```

```
;  640          0700  6          BDB_VBN1 = 0;
;  641          0701  6
;  642        P 0702  6      RETURN_ON_ERROR ( WRITE_PROLOGUE (),
;  643          0703  6                        RETURN_VBN1 );
;  644          0704  6
;  645          0705  6      ! If the key or index is to be compressed, then set the
;  646          0706  6      ! compression flag for future reference.
;  647          0707  6      !
;  648          0708  6
;  649          0709  6      IF NOT .XAB [XAB$V_IDX_NCMPR] OR NOT .XAB [XAB$V_KEY_NCMPR]
;  650          0710  6      THEN
;  651          0711  6          COMPR = 1;
;  652          0712  5      END;
```

```
;  654          0713  5          [XAB$C_ALL]:
;  655          0714  5
;  656          0715  6              BEGIN
;  657          0716  6
;  658          0717  6              ! The area descriptors go in the prologue depending on the
;  659          0718  6              ! position of the last key descriptor and the area id number.
;  660          0719  6              !
;  661          0720  6
;  662          0721  6              IF .XAB EQL 0
;  663          0722  6              THEN
;  664          0723  7                  BEGIN
;  665          0724  7                  VBN = .LAST_KEY_VBN + 1;
;  666          0725  7                  DISP = 0;
;  667          0726  7                  END
;  668          0727  6              ELSE
;  669          0728  7                  BEGIN
;  670          0729  7                  VBN = ( .XAB [XAB$B_AID] + 8 ) / 8 + .LAST_KEY_VBN;
;  671          0730  7                  DISP = ( .XAB [XAB$B_AID] MOD 8 ) * AREA$C_BLN;
;  672          0731  6                  END;
;  673          0732  6
;  674        P 0733  6              RETURN_ON_ERROR ( READ_PROLOGUE (.VBN),
;  675          0734  6                                RETURN_VBN1 );
;  676          0735  6
;  677          0736  6              ! If VBN 1 is read, save its BDB address in BDB_VBN1 so that
;  678          0737  6              ! if we fail reading or writing the prologue, this locked VBN
;  679          0738  6              ! can be released.
;  680          0739  6              !
;  681          0740  6              IF .VBN EQL 1
;  682          0741  6              THEN
;  683          0742  6                  BDB_VBN1 = .BDB;
;  684          0743  6
;  685          0744  6              ! Set the starting address of the area descriptor,
;  686          0745  6              ! and clear the buffer.
;  687          0746  6              !
;  688          0747  6              BUF = .BDB [BDB$L_ADDR] + .DISP;
;  689          0748  6              CLEAR (.BUF, AREA$C_BLN);
;  690          0749  6
;  691          0750  6              ! If no allocation XAB, then default values
;  692          0751  6              !
;  693          0752  6
;  694          0753  6              IF .XAB EQL 0
;  695          0754  6              THEN
;  696          0755  7                  BEGIN
;  697          0756  7                  BUF [AREA$L_CVBN] = 1;
;  698          0757  7                  BUF [AREA$B_ARBKTSZ] = .IFAB [IFB$B_BKS];
;  699          0758  7                  BUF [AREA$L_NXTVBN] = .VBN + 1;
;  700          0759  7                  BUF [AREA$L_CNBLK] = .AREA0_SIZE;
;  701          0760  7                  BUF [AREA$L_USED] = .VBN;
;  702          0761  7                  BUF [AREA$W_DEQ] = .IFAB [IFB$W_DEQ];
;  703          0762  7                  BUF [AREA$L_TOTAL_ALLOC] = .AREA0_SIZE;
;  704          0763  7                  IFAB [IFB$B_AVBN] = .VBN;
;  705          0764  7                  END
;  706          0765  6              ELSE
;  707          0766  7                  BEGIN
;  708          0767  7                  BUF [AREA$B_AREAID] = .XAB [XAB$B_AID];
;  709          0768  7                  BUF [AREA$W_DEQ] = .XAB [XAB$W_DEQ];
;  710          0769  7                  BUF [AREA$B_ARBKTSZ] = .XAB [XAB$B_BKZ];
```

```
;  711     0770  7          BUF [AREA$V_CBT] = .XAB [XAB$V_CBT];
;  712     0771  7          BUF [AREA$V_CTG] = .XAB [XAB$V_CTG];
;  713     0772  7          BUF [AREA$V_ONC] = .XAB [XAB$V_ONC];
;  714     0773  7          BUF [AREA$B_ALN] = .XAB [XAB$B_ALN];
;  715     0774  7
;  716     0775  7          ! The alignment data are stored in the area descriptor
;  717     0776  7          ! depending on the alignment option the user specified.
;  718     0777  7          !
;  719     0778  7          CASE .XAB [XAB$B_ALN] FROM XAB$C_ANY TO XAB$C_RFI OF
;  720     0779  7              SET
;  721     0780  7
;  722     0781  7              [XAB$C_ANY] : 0;
;  723     0782  7
;  724     0783  7              [XAB$C_CYL, XAB$C_LBN]:
;  725     0784  8                  BEGIN
;  726     0785  8                  BUF [AREA$W_VOLUME] = .XAB [XAB$W_VOL];
;  727     0786  8                  BUF [AREA$L_LOC] = .XAB [XAB$L_LOC];
;  728     0787  8                  BUF [AREA$V_HARD] = .XAB [XAB$V_HRD];
;  729     0788  7                  END;
;  730     0789  7
;  731     0790  7              [XAB$C_VBN]:
;  732     0791  7                  BUF [AREA$L_LOC] = .XAB [XAB$L_LOC];
;  733     0792  7
;  734     0793  7              [XAB$C_RFI]:
;  735     0794  8                  BEGIN
;  736     0795  8
;  737     0796  8                  LOCAL
;  738     0797  8                      RFI : REF VECTOR [,WORD];
;  739     0798  8
;  740     0799  8                  RFI = BUF [AREA$W_RFI];
;  741     0800  8                  RFI [0] = .XAB [XAB$W_RFI0];
;  742     0801  8                  RFI [1] = .XAB [XAB$W_RFI2];
;  743     0802  8                  RFI [2] = .XAB [XAB$W_RFI4];
;  744     0803  8                  BUF [AREA$L_LOC] = .XAB [XAB$L_LOC];
;  745     0804  7                  END;
;  746     0805  7
;  747     0806  7              [OUTRANGE] :
;  748     0807  8                  BEGIN
;  749     0808  8                  FAB [FAB$L_STV] = .XAB [XAB$B_AID];
;  750   P 0809  8                  RETURN_ON_ERROR ( RMSERR (ALN),
;  751     0810  8                                    RETURN_ALL_BUFS );
;  752     0811  7                  END;
;  753     0812  7              TES;
;  754     0813  7
;  755     0814  7          IF .XAB [XAB$B_AID] NEQ 0
;  756     0815  7          THEN
;  757     0816  8              BEGIN
;  758     0817  8
;  759     0818  8              IF .XAB [XAB$L_ALQ] NEQ 0
;  760     0819  8              THEN
;  761     0820  8
;  762   P 0821  8                  RETURN_ON_ERROR ( EXTEND ( .BUF, .XAB),
;  763     0822  8                                    RETURN_ALL_BUFS );
;  764     0823  8              END
;  765     0824  8
;  766     0825  7          ELSE
;  767     0826  7
```

```
;  768          0827  7                    ! Area number 0
;  769          0828  7                    !
;  770          0829  8                    BEGIN
;  771          0830  8                    BUF [AREA$L_USED] = .IFAB [IFB$B_AVBN];
;  772          0831  8                    BUF [AREA$L_NXTVBN] = .IFAB [IFB$B_AVBN] + 1;
;  773          0832  8                    IFAB [IFB$B_AVBN] = .VBN;
;  774          0833  8                    BUF [AREA$L_CVBN] = 1;
;  775          0834  8                    BUF [AREA$L_CNBLK] = .AREA0_SIZE;
;  776          0835  8                    BUF [AREA$L_TOTAL_ALLOC] = .AREA0_SIZE;
;  777          0836  7                    END;
;  778          0837  6                END;
;  779          0838  6
;  780          0839  6                ! Write the VBN we just filled.  If sharing and we are writing
;  781          0840  6                ! VBN 1 then clear bdb as it will be released when we write
;  782          0841  6                ! the buffer.  Clearing bdb before the write is okay, because
;  783          0842  6                ! if an error occurs, all we can do is release the lock.
;  784          0843  6                !
;  785          0844  6                IF NOT .IFAB [IFB$V_NORECLK]
;  786          0845  6                   AND .BDB [BDB$L_VBN] EQL 1
;  787          0846  6                THEN
;  788          0847  6                    BDB_VBN1 = 0;
;  789          0848  6
;  790        P 0849  6                RETURN_ON_ERROR ( WRITE_PROLOGUE (),
;  791          0850  6                                  RETURN_VBN1 );
;  792          0851  6
;  793          0852  6                ALLOC = 1;              ! an area descriptor was found
;  794          0853  5                END;
;  795          0854  5
;  796          0855  5            [INRANGE] : 0;
;  797          0856  5            [OUTRANGE] : 0;
;  798          0857  5
;  799          0858  5            TES;
;  800          0859  5
;  801          0860  5        IF .XAB NEQ 0
;  802          0861  5        THEN
;  803          0862  5            XAB = .XAB [XAB$L_NXT];             ! get next XAB
;  804          0863  4        END;                                    ! end of WHILE loop
```

```
;  806          0864  4
;  807          0865  4      ! If the index of any of the keys, or the key itself is compressed,
;  808          0866  4      ! then add two bytes for compression overhead to the key buffer.
;  809          0867  4      !
;  810          0868  4
;  811          0869  4      IF .COMPR
;  812          0870  4      THEN
;  813          0871  4          IFAB [IFB$W_KBUFSZ] = .IFAB [IFB$W_KBUFSZ] + 2;
;  814          0872  4
;  815          0873  3      END;                                    !  end of def of VBN, DISP and others
;  816          0874  3
;  817          0875  3      ! Reread VBN 1 and fill in summary area information and prologue
;  818          0876  3      ! version number
;  819          0877  3      !
;  820          0878  4      BEGIN
;  821          0879  4
;  822          0880  4      GLOBAL REGISTER
;  823          0881  4          R_BKT_ADDR_STR;
;  824          0882  4
;  825        P 0883  4      RETURN_ON_ERROR ( RMSCACHE ( 1, 512, CSH$M_LOCK ),
;  826        P 0884  4                        BEGIN
;  827        P 0885  4                        IF .FAB [FAB$L_STV] EQL 0
;  828        P 0886  4                        THEN
;  829        P 0887  4                            FAB [FAB$L_STV] = .STATUS OR 1^16;
;  830        P 0888  4                        STATUS = RMSERR (RPL)
;  831          0889  4                        END );
;  832          0890  4
;  833          0891  4      BKT_ADDR [PLG$B_AVBN] = .IFAB [IFB$B_AVBN];
;  834          0892  4      BKT_ADDR [PLG$B_AMAX] = .IFAB [IFB$B_AMAX];
;  835          0893  4      BKT_ADDR [PLG$W_VER_NO] = .IFAB [IFB$B_PLG_VER];
;  836          0894  4      RMS$MAKSUM ( .BKT_ADDR );
;  837          0895  3      END;
;  838          0896  3
;  839          0897  3      BDB [BDB$V_VAL] = 1;
;  840          0898  3      BDB [BDB$V_DRT] = 1;
;  841          0899  3
;  842        P 0900  3      RETURN_ON_ERROR ( RMSRELEASE ( RLS$M_WRT_THRU ),
;  843        P 0901  3                        BEGIN
;  844        P 0902  3                        IF .FAB [FAB$L_STV] EQL 0
;  845        P 0903  3                        THEN
;  846        P 0904  3                            FAB [FAB$L_STV] = .STATUS OR 1^16;
;  847        P 0905  3                        STATUS = RMSERR (WPL)
;  848          0906  3                        END);
;  849          0907  3
;  850          0908  2      END;                                    ! end of BDB register declaration
;  851          0909  2
;  852          0910  2      RETURN .CRE_STATUS;
;  853          0911  2
;  854          0912  1      END;


                                                                    .TITLE  RM3CREATE
                                                                    .IDENT  \V04-000\

                                                                    .EXTRN  RMSAL_KEY_DESC, RMSALBLB
                                                                    .EXTRN  RMSALDBUF, RMSCACHE
                                                                    .EXTRN  RMSCREATECOM, RMSEXTENDO
```

```
                                                  .EXTRN   RM$GETSPC, RM$MAKSUM
                                                  .EXTRN   RM$RELEASE, RM$RETSPC
                                                  .EXTRN   RM$RND_DEV, PIO$GB_RMSPROLOG
                                                  .EXTRN   SYS$GB_RMSPROLOG

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

                      00FC   8F   BB 00000 RM$CREATE3B::
                                                  PUSHR    #^M<R2,R3,R4,R5,R6,R7>                  ; 0310
                5E           2C   C2 00004        SUBL2    #44, SP                                 ;
         24     AE           5A   D0 00007        MOVL     IFAB, FWA                               ; 0392
                5A           59   D0 0000B        MOVL     IFAB_FILE, IFAB                         ; 0393
                52     40    8F   9A 0000E        MOVZBL   #64, R2                                 ; 0398
                51           5A   D0 00012        MOVL     IFAB, R1                                ;
                  00000000G  00   16 00015        JSB      RM$GETSPC                               ;
         20     AE           51   D0 0001B        MOVL     R1, 32(SP)                              ;
                78           50   E9 0001F        BLBC     STATUS, 3$                              ;
                       20    AE   DD 00022        PUSHL    SPACE_ADDR                              ; 0408
                      0000V  30 00025             BSBW     CHECK_INPUT                             ;
                5E           04   C0 00028        ADDL2    #4, SP                                  ;
         28     AE           50   D0 0002B        MOVL     R0, STATUS                              ;
                54     20    AE   D0 0002F        MOVL     SPACE_ADDR, R4                          ; 0413
                53           5A   D0 00033        MOVL     IFAB, R3                                ;
                52     40    8F   9A 00036        MOVZBL   #64, R2                                 ;
                  00000000G  00   16 0003A        JSB      RM$RETSPC                               ;
                06     28    AE   E8 00040        BLBS     STATUS, 1$                              ; 0419
                50     28    AE   D0 00044        MOVL     STATUS, R0                              ; 0421
                             50   11 00048        BRB      3$                                      ;
       00B7     CA           57   90 0004A 1$:    MOVB     PROLOGUE, 183(IFAB)                     ; 0423
                50   00B2    CA   9A 0004F        MOVZBL   178(IFAB), R0                           ; 0436
                51   00B1    CA   9A 00054        MOVZBL   177(IFAB), R1                           ;
                55         6140   7E 00059        MOVAQ    (R1)[R0], SIZE                          ;
                0C           55   D1 0005D        CMPL     SIZE, #12                               ; 0438
                             03   18 00060        BGEQ     2$                                      ;
                55           0C   D0 00062        MOVL     #12, SIZE                               ; 0440
                52           55   D0 00065 2$:    MOVL     SIZE, R2                                ; 0441
                51           5A   D0 00068        MOVL     IFAB, R1                                ;
                  00000000G  00   16 0006B        JSB      RM$GETSPC                               ;
         20     AE           51   D0 00071        MOVL     R1, 32(SP)                              ;
                7D           50   E9 00075        BLBC     STATUS, 11$                             ;
                       20    AE   DD 00078        PUSHL    SPACE_ADDR                              ; 0447
                      0000V  30 0007B             BSBW     CHECK_AREAS                             ;
                5E           04   C0 0007E        ADDL2    #4, SP                                  ;
                57           50   D0 00081        MOVL     R0, STATUS                              ;
                54     20    AE   D0 00084        MOVL     SPACE_ADDR, R4                          ; 0449
                53           5A   D0 00088        MOVL     IFAB, R3                                ;
                52           55   D0 0008B        MOVL     SIZE, R2                                ;
                  00000000G  00   16 0008E        JSB      RM$RETSPC                               ;
                06           57   E8 00094        BLBS     STATUS, 4$                              ; 0454
                50           57   D0 00097        MOVL     STATUS, R0                              ; 0456
                      0405   31 0009A 3$:         BRW      64$                                     ;
                5A     24    AE   D0 0009D 4$:    MOVL     FWA, IFAB                               ; 0461
                  00000000G  00   16 000A1        JSB      RM$CREATECOM                            ; 0462
         24     AE           50   D0 000A7        MOVL     R0, CRE_STATUS                          ;
                03     24    AE   E8 000AB        BLBS     CRE_STATUS, 6$                          ; 0464
                      03EC   31 000AF 5$:         BRW      63$                                     ;
  06     07     A8           01   E1 000B2 6$:    BBC      #1, 7(FAB), 7$                          ; 0465
```

```
                                01        24    AE   D1 000B7          CMPL    CRE_STATUS, #1                          ; 0466
                                          F2   13 000BB              BEQL    5$                                      ;
                                5A              59   D0 000BD 7$:     MOVL    IFAB_FILE, IFAB                         ; 0474
                             C9       16    A8   05   E1 000C0       BBC     #5, 22(FAB), 8$                         ; 0485
                                00000000G     00   16 000C5          JSB     RM$RND_DEV                              ; 0486
                                E1              50   E9 000CB          BLBC    R0, 5$                                  ;
                                57      0200    8F   3C 000CE 8$:     MOVZWL  #512, BKS                               ; 0508
                                55      0200    8F   3C 000D3       MOVZWL  #512, BUFSIZ                            ;
                             05       00A0    CA   03   E1 000D8       BBC     #3, 160(IFAB), 9$                       ; 0510
                                55      0094    C5   9E 000DE       MOVAB   148(R5), BUFSIZ                         ; 0512
                             06       00A0    CA   02   E1 000E3 9$:     BBC     #2, 160(IFAB), 10$                      ; 0514
                                55      0094 C745   9E 000E9       MOVAB   148(BKS)[BUFSIZ], BUFSIZ                ; 0516
                                00000000G     00   16 000EF 10$:    JSB     RM$ALDBUF                               ; 0518
                                A2              50   E9 000F5 11$:    BLBC    STATUS, 3$                              ;
                                        0084    CA   B6 000F8       INCW    132(IFAB)                               ; 0519
                                51        44    AA   D0 000FC       MOVL    68(IFAB), DBDB                          ; 0524
                                   16   A1          57   B0 00100       MOVW    BKS, 22(DBDB)                           ; 0526
                             27       00A0    CA   03   E1 00104       BBC     #3, 160(IFAB), 12$                      ; 0528
                                50        18    A1   D0 0010A       MOVL    24(DBDB), JBDB                          ; 0531
                                   34   A1          50   D0 0010E       MOVL    JBDB, 52(DBDB)                          ; 0532
                                   08   A0    140C   8F   B0 00112       MOVW    #5132, 8(JBDB)                          ; 0534
                                60              50   D0 00118       MOVL    JBDB, (JBDB)                            ; 0536
                                   04   A0          50   D0 0011B       MOVL    JBDB, 4(JBDB)                           ; 0537
                          16   A0              57   8F   A1 0011F       ADDW3   #68, BKS, 22(JBDB)                      ; 0538
                                   18   A0      50   A0   9E 00126       MOVAB   80(R0), 24(JBDB)                        ; 0539
                                   18   A1    0094   C0   9E 0012B       MOVAB   148(R0), 24(DBDB)                       ; 0540
                             22       00A0    CA   02   E1 00131 12$:    BBC     #2, 160(IFAB), 13$                      ; 0543
                             50              57        18   A1   C1 00137       ADDL3   24(DBDB), BKS, JBDB             ; 0546
                                   30   A1          50   D0 0013C       MOVL    JBDB, 48(DBDB)                          ; 0547
                                   08   A0    140C   8F   B0 00140       MOVW    #5132, 8(JBDB)                          ; 0549
                                60              50   D0 00146       MOVL    JBDB, (JBDB)                            ; 0551
                                   04   A0          50   D0 00149       MOVL    JBDB, 4(JBDB)                           ; 0552
                          16   A0              57   8F   A1 0014D       ADDW3   #68, BKS, 22(JBDB)                      ; 0553
                                   18   A0      50   A0   9E 00154       MOVAB   80(R0), 24(JBDB)                        ; 0554
                             09       06    AA   03   E0 00159 13$:    BBS     #3, 6(IFAB), 14$                        ; 0562
                                00000000G     00   16 0015E          JSB     RM$ALBLB                                ; 0569
                                3B              50   E9 00164          BLBC    STATUS, 15$                             ;
                                        28    AE   94 00167 14$:    CLRB    COMPR                                   ; 0588
                                        1C    AE   94 0016A       CLRB    ALLOC                                   ; 0589
                                              6E   7C 0016D       CLRQ    BDB_VBN1                                ; 0590
                                50      00B2    CA   9A 0016F       MOVZBL  178(IFAB), R0                           ; 0592
                                50              03   C0 00174       ADDL2   #3, R0                                  ;
                                50              05   C6 00177       DIVL2   #5, R0                                  ;
                                   20   AE        01   A0   9E 0017A       MOVAB   1(R0), LAST_KEY_VBN                     ;
                                   10   AE        70   AA   D0 0017F       MOVL    112(IFAB), AREA0_SIZE                   ; 0593
                             20       06    AA   03   E0 0C184       BBS     #3, 6(IFAB), 17$                        ; 0599
                                53              0D   D0 00189       MOVL    #13, R3                                 ; 0607
                                52      0200    8F   3C 0018C       MOVZWL  #512, R2                                ;
                                51              01   D0 00191       MOVL    #1, R1                                  ;
                                00000000G     00   16 00194       JSB     RM$CACHE                                ;
                                08              50   E8 0019A       BLBS    STATUS, 16$                             ;
                                50      C134    8F   3C 0019D       MOVZWL  #49460, STATUS                          ;
                                              02FD   31 001A2 15$:    BRW     64$                                     ;
                                   04   AE          54   D0 001A5 16$:    MOVL    BDB, BLB_VBN1                           ; 0608
                                56        24    A8   D0 001A9 17$:    MOVL    36(FAB), XAB                            ; 0613
                                              0D   12 001AD 18$:    BNEQ    21$                                     ; 0625
                                03        1C    AE   E9 001AF 19$:    BLBC    ALLOC, 20$                              ; 0629
```

```
                                027F 31 001B3         BRW     58$
                  18   AE         14 90 001B6 20$:    MOVB    #20, XAB_TYP                  ; 0633
                                  0A 11 001BA         BRB     22$                           ; 0625
                               0000V 30 001BC 21$:    BSBW    REPROBE                       ; 0640
                       E0         50 E9 001BF         BLBC    STATUS, 15$                   ;
                  18   AE         66 90 001C2         MOVB    (XAB), XAB_TYP                ; 0641
       0D              12   18    AE 8F 001C6 22$:    CASEB   XAB_TYP, #18, #13             ; 0643
001F   00C9          025C       025C    001CB 23$:    .WORD   56$-23$,-                     ;
025C   025C          025C       025C    001D3                 56$-23$,-                     ;
025C   025C          025C       025C    001DB                 35$-23$,-                     ;
                     025C       025C    001E3                 24$-23$,-                     ;
                                                              56$-23$,-                     ;
                                                              56$-23$,-                     ;
                                                              56$-23$,-                     ;
                                                              56$-23$,-                     ;
                                                              56$-23$,-                     ;
                                                              56$-23$,-                     ;
                                                              56$-23$,-                     ;
                                                              56$-23$,-                     ;
                                                              56$-23$,-                     ;
                                                              56$-23$                       ;
                                023D 31 001E7         BRW     56$                           ;
                       51   17    A6 9A 001EA 24$:    MOVZBL  23(XAB), R1                   ; 0654
                                  08 12 001EE         BNEQ    25$                           ;
                       55         01 D0 001F0         MOVL    #1, VBN                       ; 0657
                            08    AE D4 001F3         CLRL    DISP                          ; 0658
                                  22 11 001F6         BRB     26$                           ; 0654
                       50   04    A1 9E 001F8 25$:    MOVAB   4(R1), R0                     ; 0662
                       50         05 C6 001FC         DIVL2   #5, R0                        ;
                       55   01    A0 9E 001FF         MOVAB   1(R0), VBN                    ;
7E FFFFFFFF      8F    51         01 7A 00203         EMUL    #1, R1, #-1, -(SP)            ; 0663
50               50    8E         05 7B 0020C         EDIV    #5, (SP)+, R0, R0             ;
         08      AE    50 00000066 8F C5 00211        MULL3   #102, R0, DISP                ; 0664
                                  55 DD 0021A 26$:    PUSHL   VBN                           ; 0668
                               0000V 30 0021C         BSBW    READ_PROLOGUE                 ;
                       5E         04 C0 0021F         ADDL2   #4, SP                        ;
                  14   AE         50 D0 00222         MOVL    R0, STATUS                    ;
                       29   14    AE E9 00226         BLBC    STATUS, 30$                   ;
                       01         55 D1 0022A         CMPL    VBN, #1                       ; 0674
                                  03 12 0022D         BNEQ    27$                           ;
                       6E         54 D0 0022F         MOVL    BDB, BDB_VBN1                 ; 0676
           57     18   A4   08    AE C1 00232 27$:    ADDL3   DISP, 24(BDB), BUF            ; 0682
                               0000V 30 00238         BSBW    WRITE_KEY_DESC                ; 0685
                  14   AE         50 D0 0023B         MOVL    R0, STATUS                    ;
                       13   14    AE E8 0023F         BLBS    STATUS, 31$                   ;
                       6E         54 D1 00243 28$:    CMPL    BDB, BDB_VBN1                 ;
                                  03 12 00246         BNEQ    29$                           ;
                                01CA 31 00248         BRW     54$                           ;
                                  53 D4 0024B 29$:    CLRL    R3                            ;
                       00000000G  00 16 0024D         JSB     RM$RELEASE                    ;
                                01B2 31 00253 30$:    BRW     52$                           ;
                            08    AE DD 00256 31$:    PUSHL   DISP                          ; 0690
                                  55 DD 00259         PUSHL   VBN                           ;
                                  57 DD 0025B         PUSHL   BUF                           ;
                               0000V 30 0025D         BSBW    AL_KEY_DESC                   ;
                       5E         0C C0 00260         ADDL2   #12, SP                       ;
                  14   AE         50 D0 00263         MOVL    R0, STATUS                    ;
```

```
                           D8         14   AE  E9 00267          BLBC    STATUS, 28$                               ;
               08      06  AA              03  E0 0026B          BBS     #3, 6(IFAB), 32$                          ; 0697
                           01         1C   A4  D1 00270          CMPL    28(BDB), #1                               ; 0698
                                           02  12 00274          BNEQ    32$                                       ;
                                           6E  D4 00276          CLRL    BDB_VBN1                                  ; 0700
                                        0000V  30 00278 32$:     BSBW    WRITE_PROLOGUE                            ; 0703
                       14  AE              50  D0 0027B          MOVL    R0, STATUS                                ;
                           D0         14   AE  E9 0027F          BLBC    STATUS, 30$                               ;
               05      12  A6              03  E1 00283          BBC     #3, 18(XAB), 33$                          ; 0709
               04      12  A6              06  E0 00288          BBS     #6, 18(XAB), 34$                          ;
                       28  AE              01  90 0028D 33$:     MOVB    #1, COMPR                                 ; 0711
                                         0193  31 00291 34$:     BRW     56$                                       ; 0643
                                      14   AE  D4 00294 35$:     CLRL    20(SP)                                    ; 0721
                                           56  D5 00297          TSTL    XAB                                       ;
                                           0D  12 00299          BNEQ    36$                                       ;
                                      14   AE  D6 0029B          INCL    20(SP)                                    ;
               55      20  AE              01  C1 0029E          ADDL3   #1, LAST_KEY_VBN, VBN                     ; 0724
                                      08   AE  D4 002A3          CLRL    DISP                                      ; 0725
                                           22  11 002A6          BRB     37$                                       ; 0721
                           50         17   A6  9A 002A8 36$:     MOVZBL  23(XAB), R0                               ; 0729
                           50              08  C0 002AC          ADDL2   #8, R0                                    ;
                           50              08  C6 002AF          DIVL2   #8, R0                                    ;
               55          50         20   AE  C1 002B2          ADDL3   LAST_KEY_VBN, R0, VBN                     ;
                           50         17   A6  9A 002B7          MOVZBL  23(XAB), R0                               ; 0730
     7E        00          50              01  7A 002BB          EMUL    #1, R0, #0, -(SP)                         ;
     50        50          8E              08  7B 002C0          EDIV    #8, (SP)+, R0, R0                         ;
          08   AE          50              06  78 002C5          ASHL    #6, R0, DISP                              ;
                                           55  DD 002CA 37$:     PUSHL   VBN                                       ; 0734
                                        0000V  30 002CC          BSBW    READ_PROLOGUE                             ;
                           5E              04  C0 002CF          ADDL2   #4, SP                                    ;
                       0C  AE              50  D0 002D2          MOVL    R0, STATUS                                ;
                           1C         0C   AE  E8 002D6          BLBS    STATUS, 40$                               ;
                                           6E  D5 002DA          TSTL    BDB_VBN1                                  ;
                                           06  12 002DC          BNEQ    38$                                       ;
                           54         04   AE  D0 002DE          MOVL    BLB_VBN1, BDB                             ;
                                           03  11 002E2          BRB     39$                                       ;
                           54              6E  D0 002E4 38$:     MOVL    BDB_VBN1, BDB                             ;
                                           53  D4 002E7 39$:     CLRL    R3                                        ;
                                00000000G  00  16 002E9          JSB     RM$RELEASE                                ;
                           50         0C   AE  D0 002EF          MOVL    STATUS, R0                                ;
                                         01AC  31 002F3          BRW     64$                                       ;
                           01              55  D1 002F6 40$:     CMPL    VBN, #1                                   ; 0740
                                           03  12 002F9          BNEQ    41$                                       ;
                           6E              54  D0 002FB          MOVL    BDB, BDB_VBN1                             ; 0742
               57      18  A4         08   AE  C1 002FE 41$:     ADDL3   DISP, 24(BDB), BUF                        ; 0747
                           7E         40   8F  9A 00304          MOVZBL  #64, -(SP)                                ; 0748
                                           57  DD 00308          PUSHL   BUF                                       ;
                                        0000V  30 0030A          BSBW    CLEAR                                     ;
                           5E              08  C0 0030D          ADDL2   #8, SP                                    ;
                           29         14   AE  E9 00310          BLBC    20(SP), 42$                               ; 0761
                       0C  A7              01  D0 00314          MOVL    #1, 12(BUF)                               ; 0756
                       03  A7         5E   AA  90 00318          MOVB    94(IFAB), 3(BUF)                          ; 0757
                       18  A7         01   A5  9E 0031D          MOVAB   1(R5), 24(BUF)                            ; 0758
                       10  A7         10   AE  D0 00322          MOVL    AREA0_SIZE, 16(BUF)                       ; 0759
                       14  A7              55  D0 00327          MOVL    VBN, 20(BUF)                              ; 0760
                       24  A7         62   AA  B0 0032B          MOVW    98(IFAB), 36(BUF)                         ; 0761
                       32  A7         10   AE  D0 00330          MOVL    AREA0_SIZE, 50(BUF)                       ; 0762
```

```
                                00B0   CA          55  90 00335          MOVB    VBN, 176(IFAB)                          ; 0763
                                       00B3        31  0033A          BRW     50$                                     ; 0753
                                02     A7      17  A6  90 0033D 42$:   MOVB    23(XAB), 2(BUF)                         ; 0767
                                24     A7      14  A6  B0 00342        MOVW    20(XAB), 36(BUF)                        ; 0768
                                03     A7      16  A6  90 00347        MOVB    22(XAB), 3(BUF)                         ; 0769
        50      08     A6                      01      05 EF 0034C        EXTZV   #5, #1, 8(XAB), R0                      ; 0770
07      A7             01                      05      50 F0 00352        INSV    R0, #5, #1, 7(BUF)                      ;
        50      08     A6                      01      07 EF 00358        EXTZV   #7, #1, 8(XAB), R0                      ; 0771
07      A7             01                      07      50 F0 0035E        INSV    R0, #7, #1, 7(BUF)                      ;
        50      08     A6                      01      01 EF 00364        EXTZV   #1, #1, 8(XAB), R0                      ; 0772
07      A7             01                      01      50 F0 0036A        INSV    R0, #1, #1, 7(BUF)                      ;
                               06      A7      09  A6  90 00370          MOVB    9(XAB), 6(BUF)                          ; 0773
                04             00      09  A6  8F 00375          CASEB   9(XAB), #0, #4                          ; 0778
    0037               0017            0017        003C       0037A 43$:   .WORD   47$-43$,-                               ;
                                           002A        00382                44$-43$,-                               ;
                                                                            44$-43$,-                               ;
                                                                            46$-43$,-                               ;
                                                                            45$-43$                                 ;
                                0C     A8      17  A6  9A 00384          MOVZBL  23(XAB), 12(FAB)                        ; 0808
                                14     AE  83FC    8F  3C 00389          MOVZWL  #33788, STATUS                          ; 0810
                                           3A      11  0038F          BRB     48$                                     ;
                                04     A7      0A  A6  B0 00391 44$:   MOVW    10(XAB), 4(BUF)                         ; 0785
                                28     A7      0C  A6  D0 00396        MOVL    12(XAB), 40(BUF)                        ; 0786
07      A7             01              00      08  A6  F0 0039B        INSV    8(XAB), #0, #1, 7(BUF)                  ; 0787
                                           12      11  003A2          BRB     47$                                     ; 0778
                                       50  2C      A7  9E 003A4 45$:   MOVAB   44(R7), RFI                             ; 0799
                                       60  18      A6  D0 003A8        MOVL    24(XAB), (RFI)                          ; 0800
                                04     A0  1C      A6  B0 003AC        MOVW    28(XAB), 4(RFI)                         ; 0802
                                28     A7  0C      A6  D0 003B1 46$:   MOVL    12(XAB), 40(BUF)                        ; 0803
                                           17      A6  95 003B6 47$:   TSTB    23(XAB)                                 ; 0814
                                           13      13  003B9          BEQL    49$                                     ;
                                           10      A6  D5 003BB        TSTL    16(XAB)                                 ; 0818
                                           30      13  003BE          BEQL    50$                                     ;
                                       0000V       30  003C0          BSBW    EXTEND                                  ; 0822
                                14     AE          50  D0 003C3        MOVL    R0, STATUS                              ;
                                       25      14  AE  E8 003C7        BLBS    STATUS, 50$                             ;
                                           FE75        31  003CB 48$:   BRW     28$                                     ;
                                14     A7  00B0    CA  9A 003CE 49$:   MOVZBL  176(IFAB), 20(BUF)                      ; 0830
                                18     A7  00B0    CA  9A 003D4        MOVZBL  176(IFAB), 24(BUF)                      ; 0831
                                           18      A7  D6 003DA        INCL    24(BUF)                                 ;
                                00B0   CA          55  90 003DD        MOVB    VBN, 176(IFAB)                          ; 0832
                                0C     A7          01  D0 003E2        MOVL    #1, 12(BUF)                             ; 0833
                                10     A7      10  AE  D0 003E6        MOVL    AREA0_SIZE, 16(BUF)                     ; 0834
                                32     A7      10  AE  D0 003EB        MOVL    AREA0_SIZE, 50(BUF)                     ; 0835
08              06     AA                      03  E0 003F0 50$:   BBS     #3, 6(IFAB), 51$                        ; 0844
                               01      1C      A4  D1 003F5        CMPL    28(BDB), #1                             ; 0845
                                           02      12  003F9          BNEQ    51$                                     ;
                                           6E      D4  003FB          CLRL    BDB_VBN1                                ; 0847
                                       0000V       30  003FD 51$:   BSBW    WRITE_PROLOGUE                          ; 0850
                                14     AE          50  D0 00400        MOVL    R0, STATUS                              ;
                                       1B      14  AE  E8 00404        BLBS    STATUS, 55$                             ;
                                           6E      D5  00408 52$:   TSTL    BDB_VBN1                                ;
                                           06      12  0040A          BNEQ    53$                                     ;
                                       54      04  AE  D0 0040C        MOVL    BLB_VBN1, BDB                           ;
                                           03      11  00410          BRB     54$                                     ;
                                       54          6E  D0 00412 53$:   MOVL    BDB_VBN1, BDB                           ;
                                                   53  D4 00415 54$:   CLRL    R3                                      ;
```

```
                    00000000G  00  16 00417          JSB     RM$RELEASE                                      ;
                50        14   AE  D0 0041D          MOVL    STATUS, R0                                      ;
                             7F  11 00421          BRB     64$                                             ;
              1C    AE           01  90 00423 55$:   MOVB    #1, ALLOC                                       ; 0852
                             56  D5 00427 56$:   TSTL    XAB                                             ; 0860
                             03  12 00429          BNEQ    57$                                             ;
                          FD81   31 0042B          BRW     19$                                             ;
                56        04   A6  D0 0042E 57$:   MOVL    4(XAB), XAB                                     ; 0862
                          FD78   31 00432          BRW     18$                                             ; 0615
                05        28   AE  E9 00435 58$:   BLBC    COMPR, 59$                                      ; 0869
              00B4    CA           02  A0 00439          ADDW2   #2, 180(IFAB)                           ; 0871
                53           01  D0 0043E 59$:   MOVL    #1, R3                                          ; 0889
                52      02C0   8F  3C 00441          MOVZWL  #512, R2                                        ;
                51           01  D0 00446          MOVL    #1, R1                                          ;
                    00000000G  00  16 00449          JSB     RM$CACHE                                        ;
                15           50  E8 0044F          BLBS    STATUS, 61$                                     ;
                          0C   A8  D5 00452          TSTL    12(FAB)                                         ;
                             09  12 00455          BNEQ    60$                                             ;
          0C    A8        50 00010000  8F  C9 00457          BISL3   #65536, STATUS, 12(FAB)                 ;
                50      C104   8F  3C 00460 60$:   MOVZWL  #49412, STATUS                                  ;
                             3B  11 00465          BRB     64$                                             ;
              66    A5      00B0   CA  B0 00467 61$:   MOVW    176(IFAB), 102(BKT_ADDR)                        ; 0891
              74    A5      00B7   CA  9B 0046D          MOVZBW  183(IFAB), 116(BKT_ADDR)                    ; 0893
                    00000000G  00  16 00473          JSB     RM$MAKSUM                                       ; 0894
              0A    A4           03  88 00479          BISB2   #3, 10(BDB)                                     ; 0898
                53           02  D0 0047D          MOVL    #2, R3                                          ; 0906
                    00000000G  00  16 00480          JSB     RM$RELEASE                                      ;
                15           50  E8 00486          BLBS    STATUS, 63$                                     ;
                          0C   A8  D5 00489          TSTL    12(FAB)                                         ;
                             09  12 0048C          BNEQ    62$                                             ;
          0C    A8        50 00010000  8F  C9 0048E          BISL3   #65536, STATUS, 12(FAB)                 ;
                50      C11C   8F  3C 00497 62$:   MOVZWL  #49436, STATUS                                  ;
                             04  11 0049C          BRB     64$                                             ;
                50        24   AE  D0 0049E 63$:   MOVL    CRE_STATUS, R0                                  ; 0910
                5E           2C  C0 004A2 64$:   ADDL2   #44, SP                                         ; 0912
                          0FC0   8F  BA 004A5          POPR    #^M<R2,R3,R4,R5,R6,R7>                          ;
                             05  00 004A9          RSB                                                     ;

; Routine Size:  1194 bytes,    Routine Base:  RM$RMSMISC + 0000

;  855          0913  1
```

```
;  857          0914  1 %SBTTL  'AL_KEY_DESC'
;  858          0915  1 ROUTINE AL_KEY_DESC ( BUF,VBN,OFFSET ) : RL$AL_KEY =
;  859          0916  1
;  860          0917  1 !++
;  861          0918  1 !
;  862          0919  1 ! AL_KEY_DESC
;  863          0920  1 !       This routine allocates a key descriptor.  This is in a separate module
;  864          0921  1 !       so as not to clobber R7.
;  865          0922  1 !
;  866          0923  1 !
;  867          0924  1 ! CALLING SEQUENCE:
;  868          0925  1 !       AL_KEY_DESC (BUF, VBN, OFFSET)
;  869          0926  1 !
;  870          0927  1 ! INPUT PARAMETERS:
;  871          0928  1 !       BUF - address of buffer where data are stored
;  872          0929  1 !       VBN - virtual block number corresponding to buffer
;  873          0930  1 !       OFFSET - displacement into buffer
;  874          0931  1 !
;  875          0932  1 ! IMPLICIT INPUTS:
;  876          0933  1 !       None
;  877          0934  1 !
;  878          0935  1 ! OUTPUT PARAMETERS:
;  879          0936  1 !       none
;  880          0937  1 !
;  881          0938  1 ! IMPLICIT OUTPUTS:
;  882          0939  1 !       none
;  883          0940  1 !
;  884          0941  1 ! ROUTINE VALUE:
;  885          0942  1 !       Error code or success
;  886          0943  1 !
;  887          0944  1 ! SIDE EFFECTS:
;  888          0945  1 !       none
;  889          0946  1 !
;  890          0947  1 !--
;  891          0948  1
;  892          0949  2     BEGIN
;  893          0950  2
;  894          0951  2     EXTERNAL REGISTER
;  895          0952  2         COMMON_FABREG;
;  896          0953  2
;  897          0954  2     GLOBAL REGISTER
;  898          0955  2         R_IDX_DFN;
;  899          0956  2
;  900          0957  2     RETURN_ON_ERROR ( RM$AL_KEY_DESC ( .BUF, .VBN, .OFFSET ) );
;  901          0958  2
;  902          0959  2     RETURN RMSSUC();
;  903          0960  2
;  904          0961  1     END;


                                     57  DD 00000 AL_KEY_DESC:
                                                   PUSHL   R7                              ; 0915
                                 10  AE  DD 00002          PUSHL   OFFSET                  ; 0957
                                 10  AE  DD 00005          PUSHL   VBN                     ;
```

```
                    10   AE  DD 00008          PUSHL   BUF                    ;
             00000000G   00  16 0000B          JSB     RM$AL_KEY_DESC         ;
          5E             0C  C0 00011          ADDL2   #12, SP                ;
          03             50  E9 00014          BLBC    STATUS, 1$             ;
          50             01  D0 00017          MOVL    #1, R0                 ; 0959
          57             8E  D0 0001A 1$:      MOVL    (SP)+, R7              ; 0961
                         05 0001D              RSB                            ;

; Routine Size:  30 bytes,    Routine Base:  RM$RMSMISC + 04AA
```

```
;  906     0962  1 %SBTTL  'CHECK_AREAS'
;  907     0963  1 ROUTINE CHECK_AREAS ( TABLES ) : RL$CHK_AND_REPROBE =
;  908     0964  1
;  909     0965  1 !++
;  910     0966  1 !
;  911     0967  1 ! CHECK_AREAS
;  912     0968  1 !
;  913     0969  1 !       This routine loops thru the XAB chain again to validate key
;  914     0970  1 !       information against area allocation information, and also obtain
;  915     0971  1 !       a valid bucket size value for each of the index levels.  It stores
;  916     0972  1 !       all the needed information in a table in memory for future reference
;  917     0973  1 !       to avoid future passes thru the XAB chain.
;  918     0974  1 !
;  919     0975  1 ! CALLING SEQUENCE:
;  920     0976  1 !       CHECK_AREAS (TABLES)
;  921     0977  1 !
;  922     0978  1 ! INPUT PARAMETERS:
;  923     0979  1 !       TABLES - Address for tables in memory
;  924     0980  1 !
;  925     0981  1 ! IMPLICIT INPUTS:
;  926     0982  1 !       FAB, IFAB, XAB
;  927     0983  1 !
;  928     0984  1 ! OUTPUT PARAMETERS:
;  929     0985  1 !       None
;  930     0986  1 !
;  931     0987  1 ! IMPLICIT OUTPUTS:
;  932     0988  1 !       None
;  933     0989  1 !
;  934     0990  1 ! ROUTINE VALUE:
;  935     0991  1 !       Success or error code.
;  936     0992  1 !
;  937     0993  1 ! SIDE EFFECTS:
;  938     0994  1 !       None
;  939     0995  1 !--
;  940     0996  1
;  941     0997  2     BEGIN
;  942     0998  2
;  943     0999  2     EXTERNAL REGISTER
;  944     1000  2         R_FAB_STR,
;  945     1001  2         R_IFAB_STR,
;  946     1002  2         XAB = 6 : REF BBLOCK;
;  947     1003  2
;  948     1004  2     LITERAL
;  949     1005  2         VBN_SIZE = 4;
;  950     1006  2
;  951     1007  2     ! Fields for the key table
;  952     1008  2     !
;  953     1009  2     FIELD
;  954     1010  2         KEY_TBL_FLD =
;  955     1011  2             SET
;  956     1012  2             KSZ = [0,0,8,0],                ! key size
;  957     1013  2             DAN = [1,0,8,0],                ! data area number
;  958     1014  2             IAN = [2,0,8,0],                ! index area number
;  959     1015  2             LAN = [3,0,8,0],                ! lowest level index area number
;  960     1016  2             IFL = [4,0,16,0],               ! index bucket fill size
;  961     1017  2             DFL = [6,0,16,0]                ! data bucket fill size
;  962     1018  2             TES;
```

```
;  963     1019  2
;  964     1020  2      ! Define the tables and others.
;  965     1021  2      !
;  966     1022  2      LOCAL
;  967     1023  2          BIG_BKS,                                    ! stores biggest bucket size
;  968     1024  2          AREA_FOUND,                                 ! indicates if areas were defined
;  969     1025  2          MAX_MINREC,
;  970     1026  2          KEY_TBL  : REF BLOCKVECTOR [1,8,BYTE] FIELD(KEY_TBL_FLD),
;  971     1027  2          AREA_BKS : REF VECTOR [1,BYTE];             ! the only field in the area
;  972     1028  2                                                      !  table is the bucket size
;  973     1029  2
;  974     1030  2      ! Get the address for each table
;  975     1031  2      !
;  976     1032  2      KEY_TBL = .TABLES;
;  977     1033  2      AREA_BKS = .TABLES + ( 8 * .IFAB [IFB$B_NUM_KEYS] );
;  978     1034  2      BIG_BKS = 0;
;  979     1035  2      AREA_FOUND = 0;
;  980     1036  2      MAX_MINREC = 0;
;  981     1037  2
;  982     1038  2      XAB = .FAB [FAB$L_XAB];
;  983     1039  2
;  984     1040  2      WHILE .XAB NEQ 0
;  985     1041  2      DO
;  986     1042  2
;  987     1043  3          BEGIN
;  988     1044  3          RETURN_ON_ERROR ( REPROBE () );
;  989     1045  3
;  990     1046  3          ! First verify that the area numbers in the key XABs point to a valid                  ;
;  991     1047  3          ! allocation XAB.  Then fill the table with the information that must
;  992     1048  3          ! be verified later.                                                                   ;
;  993     1049  3          !
;  994     1050  3          CASE .XAB [XAB$B_COD] FROM XAB$C_DAT TO XAB$C_TRM OF
;  995     1051  3              SET
;  996     1052  3
;  997     1053  3              [XAB$C_KEY]:
;  998     1054  3
;  999     1055  4                  BEGIN
; 1000     1056  4                  FAB [FAB$L_STV] = .XAB [XAB$B_REF];
; 1001     1057  4
; 1002     1058  4                  IF .XAB [XAB$B_IAN] GEQU .IFAB [IFB$B_AMAX]
; 1003     1059  4                  THEN
; 1004     1060  4                      RETURN RMSERR (IAN);
; 1005     1061  4
; 1006     1062  4                  IF .XAB [XAB$B_LAN] GEQU .IFAB [IFB$B_AMAX]
; 1007     1063  4                  THEN
; 1008     1064  4                      RETURN RMSERR (LAN);
; 1009     1065  4
; 1010     1066  4                  IF .XAB [XAB$B_DAN] GEQU .IFAB [IFB$B_AMAX]
; 1011     1067  4                  THEN
; 1012     1068  4                      RETURN RMSERR (DAN);
; 1013     1069  4
; 1014     1070  4                  ! If LAN is defined as area 0, set it equal to IAN.
; 1015     1071  4                  !
; 1016     1072  4
; 1017     1073  4                  IF .XAB [XAB$B_LAN] EQL 0
; 1018     1074  4                  THEN
; 1019     1075  4                      XAB [XAB$B_LAN] = .XAB [XAB$B_IAN];
```

```
;1020     1076  4                    BEGIN
;1021     1077  5
;1022     1078  5                    LOCAL
;1023     1079  5                        KEYSIZE,
;1024     1080  5                        MINREC,
;1025     1081  5                        SEGS;
;1026     1082  5
;1027     1083  5                    ! Get key size, and minimum record size.  Store key size
;1028     1084  5                    ! in table, and save biggest minrec so far.
;1029     1085  5                    !
;1030     1086  5
;1031     1087  5                    GET_KEY_PARMS (;KEYSIZE, MINREC, SEGS);
;1032     1088  5                    KEY_TBL [.XAB [XAB$B_REF], KSZ] = .KEYSIZE;
;1033     1089  5
;1034     1090  5                    IF .MAX_MINREC LSS .MINREC
;1035     1091  5                    THEN
;1036     1092  5                        MAX_MINREC = .MINREC;
;1037     1093  5                    END;
;1038     1094  4
;1039     1095  4                    ! Store area numbers and fill values
;1040     1096  4                    !
;1041     1097  4                    KEY_TBL [.XAB [XAB$B_REF], IAN] = .XAB [XAB$B_IAN];
;1042     1098  4                    KEY_TBL [.XAB [XAB$B_REF], LAN] = .XAB [XAB$B_LAN];
;1043     1099  4                    KEY_TBL [.XAB [XAB$B_REF], DAN] = .XAB [XAB$B_DAN];
;1044     1100  4                    KEY_TBL [.XAB [XAB$B_REF], IFL] = .XAB [XAB$W_IFL];
;1045     1101  4                    KEY_TBL [.XAB [XAB$B_REF], DFL] = .XAB [XAB$W_DFL];
;1046     1102  4                    END;
;1047     1103  3
;1048     1104  3                [XAB$C_ALL]:
;1049     1105  3
;1050     1106  3                    ! For the allocation XAB, just save the bucket size
;1051     1107  3                    ! specified for the given area.
;1052     1108  3                    !
;1053     1109  3                    BEGIN
;1054     1110  4                    AREA_FOUND = 1;
;1055     1111  4                    AREA_BKS [.XAB [XAB$B_AID]] = .XAB [XAB$B_BKZ];
;1056     1112  4
;1057     1113  4                    IF .XAB [XAB$B_BKZ] GTR .BIG_BKS
;1058     1114  4                    THEN
;1059     1115  4                        BIG_BKS = .XAB [XAB$B_BKZ];
;1060     1116  4                    END;
;1061     1117  3
;1062     1118  3                    [OUTRANGE] : 0;
;1063     1119  3                    [INRANGE]  : 0;
;1064     1120  3
;1065     1121  3                    TES;
;1066     1122  3
;1067     1123  3                XAB = .XAB [XAB$L_NXT];
;1068     1124  3
;1069     1125  3                END;                                ! end of WHILE loop
;1070     1126  2
;1071     1127  2            ! Now that all required data have been stored in the tables,
;1072     1128  2            ! find the best bucket size for each area based on the key sizes,
;1073     1129  2            ! if the user did not give a value.  If he specified a size, then
;1074     1130  2            ! make sure that it is big enough to hold the keys and data record
;1075     1131  2            ! which correspond to the given area number.
;1076     1132  2
```

```
;  1077          1133  2     !
;  1078          1134  3     BEGIN
;  1079          1135  3
;  1080          1136  3     LOCAL
;  1081          1137  3         LAST_BKS,                       ! last bucket size found
;  1082          1138  3         MIN_BKS;                        ! minimum bucket size
;  1083          1139  3
;  1084          1140  3     ! Get the bucket size for a given area number.
;  1085          1141  3     !
;  1086          1142  4     INCR AREANO FROM 0 TO ( .IFAB [IFB$B_AMAX] - 1 )
;  1087          1143  3     DO
;  1088          1144  4         BEGIN
;  1089          1145  4
;  1090          1146  4         LAST_BKS = 0;
;  1091          1147  4
;  1092          1148  4         ! Loop thru all keys
;  1093          1149  4         !
;  1094          1150  5         INCR KEYNO FROM 0 TO ( .IFAB [IFB$B_NUM_KEYS] - 1 )
;  1095          1151  4         DO
;  1096          1152  5             BEGIN
;  1097          1153  5
;  1098          1154  5             ! If the area number is equal to the index or lowest area
;  1099          1155  5             ! number for the given key, calculate the minimum bucket size.
;  1100          1156  5             ! It is equal to the bucket overhead plus two keys and their
;  1101          1157  5             ! overhead, depending on the prologue version.
;  1102          1158  5             !
;  1103          1159  5
;  1104          1160  5             IF .AREANO EQL .KEY_TBL [.KEYNO, IAN]
;  1105          1161  5               OR .AREANO EQL .KEY_TBL [.KEYNO, LAN]
;  1106          1162  5             THEN
;  1107          1163  6                 BEGIN
;  1108          1164  6
;  1109          1165  6                 IF .IFAB [IFB$B_PLG_VER] LSS PLG$C_VER_3
;  1110          1166  6                 THEN
;  1111          1167  7                     MIN_BKS = 2 * ( .KEY_TBL [.KEYNO, KSZ] + VBN_SIZE
;  1112          1168  6                         + IRC$C_IDXOVHDSZ ) + BKT$C_OVERHDSZ
;  1113          1169  6                 ELSE
;  1114          1170  7                     MIN_BKS = 2 * ( .KEY_TBL [.KEYNO, KSZ] + VBN_SIZE
;  1115          1171  6                         + IRC$C_KEYCMPOVR ) + BKT$C_OVERHDSZ + BRT$C_ENDOVHD;
;  1116          1172  6
;  1117          1173  6                 ! The bucket size should be in number of blocks, not bytes.
;  1118          1174  6                 !
;  1119          1175  6                 MIN_BKS = ( .MIN_BKS / 512 ) + 1;
;  1120          1176  6
;  1121          1177  6                 ! If the user gave a bucket size value for the given area,
;  1122          1178  6                 ! make sure it is big enough to hold two keys.  If no area XAB
;  1123          1179  6                 ! found, use the FAB value if not zero.
;  1124          1180  6                 !
;  1125          1181  6
;  1126          1182  7                 IF ( .AREA_BKS [.AREANO] NEQ 0
;  1127          1183  7                   AND .MIN_BKS GTR .AREA_BKS [.AREANO] )
;  1128          1184  7                     OR ( NOT .AREA_FOUND
;  1129          1185  7                       AND .FAB [FAB$B_BKS] NEQ 0
;  1130          1186  7                         AND .MIN_BKS GTR .FAB [FAB$B_BKS] )
;  1131          1187  6                 THEN
;  1132          1188  7                     BEGIN
;  1133          1189  7                     FAB [FAB$L_STV] = .KEYNO;
```

```
;  1134     1190  7                RETURN RMSERR (KSI);
;  1135     1191  6                END;
;  1136     1192  6
;  1137     1193  6            ! Then get the largest bucket size found so far.
;  1138     1194  6            !
;  1139     1195  6
;  1140     1196  6            IF .LAST_BKS LSS .MIN_BKS
;  1141     1197  6            THEN
;  1142     1198  6                LAST_BKS = .MIN_BKS;
;  1143     1199  5            END;
;  1144     1200  5
;  1145     1201  5        ! If the area number is equal to the one specified for the
;  1146     1202  5        ! data level, find the minimum bucket size depending on the key
;  1147     1203  5        ! of reference.
;  1148     1204  5        !
;  1149     1205  5
;  1150     1206  5        IF .AREANO EQL .KEY_TBL [.KEYNO, DAN]
;  1151     1207  5        THEN
;  1152     1208  6            BEGIN
;  1153     1209  6
;  1154     1210  6            ! For the SIDR level, just make it equal to the key size since
;  1155     1211  6            ! we know that at least one key will always fit in the bucket.
;  1156     1212  6            !
;  1157     1213  6
;  1158     1214  6            IF .KEYNO NEQ 0
;  1159     1215  6            THEN
;  1160     1216  6                MIN_BKS = ( .KEY_TBL [.KEYNO, KSZ] / 512 ) + 1
;  1161     1217  6
;  1162     1218  6            ! For t e primary key, it depends on the bucket size value
;  1163     1219  6            ! specified in the allocation XAB.
;  1164     1220  6            !
;  1165     1221  6
;  1166     1222  6        ELSE
;  1167     1223  7            BEGIN
;  1168     1224  7
;  1169     1225  7            ! If an area XAB was given, use the allocation XAB bucket
;  1170     1226  7            ! size, else use the FAB value.
;  1171     1227  7            ! Call CHECK_MRS to make sure the record fits in the bucket.
;  1172     1228  7            !
;  1173     1229  7
;  1174     1230  7            IF .AREA_FOUND
;  1175     1231  7            THEN
;  1176     1232  7                IFAB [IFB$B_BKS] = .AREA_BKS [.AREANO]
;  1177     1233  7            ELSE
;  1178     1234  7                IFAB [IFB$B_BKS ] = .FAB [FAB$B_BKS];
;  1179     1235  7
;  1180     1236  7            IF .IFAB [IFB$W_MRS] NEQU 0
;  1181     1237  7            THEN
;  1182     1238  8                RETURN_ON_ERROR ( CHECK_MRS (.IFAB [IFB$B_PLG_VER], .IFAB [IFB$W_MRS]) )
;  1183     1239  7            ELSE
;  1184     1240  7                RETURN_ON_ERROR ( CHECK_MRS (.IFAB [IFB$B_PLG_VER], .MAX_MINREC) );
;  1185     1241  7
;  1186     1242  7            ! The best value is in the IFAB now.
;  1187     1243  7            !
;  1188     1244  7            MIN_BKS = .IFAB [IFB$B_BKS];
;  1189     1245  6            END;                                ! primary key case
;  1190     1246  6
```

```
;  1191     1247  6                ! Get largest one so far
;  1192     1248  6                !
;  1193     1249  6
;  1194     1250  6                IF .LAST_BKS LSS .MIN_BKS
;  1195     1251  6                THEN
;  1196     1252  6                    LAST_BKS = .MIN_BKS;
;  1197     1253  5                END;
;  1198     1254  4            END;                            ! end of INCR KEYNO loop
;  1199     1255  4
;  1200     1256  4        ! Now that we searched all keys for one area number, store the bucket
;  1201     1257  4        ! size calculated if the user did not give one.
;  1202     1258  4        !
;  1203     1259  4
;  1204     1260  4        IF .AREA_BKS [.AREANO] EQL 0
;  1205     1261  4        THEN
;  1206     1262  4            AREA_BKS [.AREANO] = .LAST_BKS
;  1207     1263  4        ELSE
;  1208     1264  4
;  1209     1265  4            ! If he gave one, make sure it is big enough.
;  1210     1266  4            !
;  1211     1267  4
;  1212     1268  4            IF .LAST_BKS GTR .AREA_BKS [.AREANO]
;  1213     1269  4            THEN
;  1214     1270  5                BEGIN
;  1215     1271  5                FAB [FAB$L_STV] = .AREANO;
;  1216     1272  5                RETURN RMSERR (BKZ);
;  1217     1273  4                END;
;  1218     1274  4
;  1219     1275  4        ! If no reference was made to this area, and the bucket size
;  1220     1276  4        ! given is zero, return "bucket size" error.
;  1221     1277  4        !
;  1222     1278  4
;  1223     1279  4            IF .AREA_BKS [.AREANO] EQL 0
;  1224     1280  4              AND .AREA_FOUND
;  1225     1281  4            THEN
;  1226     1282  5                BEGIN
;  1227     1283  5                FAB [FAB$L_STV] = .AREANO;
;  1228     1284  5                RETURN RMSERR (BKZ);
;  1229     1285  4                END;
;  1230     1286  4
;  1231     1287  4            ! Store the biggest bucket size value so far.
;  1232     1288  4            !
;  1233     1289  4            IF .BIG_BKS LSS .LAST_BKS
;  1234     1290  4            THEN
;  1235     1291  4                BIG_BKS = .LAST_BKS;
;  1236     1292  4
;  1237     1293  3            END;                                ! end of INCR AREANO loop
;  1238     1294  3
;  1239     1295  3        ! Store the biggest bucket size value in the FAB and IFAB.  But before, do
;  1240     1296  3        ! some validation to check the cases in which there are no key nor area
;  1241     1297  3        ! XABs defined, for example when the file is being created with block I/O set.
;  1242     1298  3        !
;  1243     1299  3
;  1244     1300  3        IF .BIG_BKS EQL 0
;  1245     1301  3        THEN
;  1246     1302  4            BEGIN
;  1247     1303  4            IFAB [IFB$B_BKS] = .FAB [FAB$B_BKS];
```

```
;  1248     1304  4
;  1249     1305  4          IF .IFAB [IFB$W_MRS] NEQU 0
;  1250     1306  4          THEN
;  1251     1307  5              RETURN_ON_ERROR ( CHECK_MRS (.IFAB [IFB$B_PLG_VER], .IFAB [IFB$W_MRS]) )
;  1252     1308  4          ELSE
;  1253     1309  4              RETURN_ON_ERROR ( CHECK_MRS (.IFAB [IFB$B_PLG_VER], .MAX_MINREC) );
;  1254     1310  4          BIG_BKS = .IFAB [IFB$B_BKS];
;  1255     1311  4          END
;  1256     1312  3      ELSE
;  1257     1313  4          BEGIN
;  1258     1314  4          FAB [FAB$B_BKS] = .BIG_BKS;
;  1259     1315  4          IFAB [IFB$B_BKS] = .BIG_BKS;
;  1260     1316  3          END;
;  1261     1317  3
;  1262     1318  3      IF .BIG_BKS EQL 0
;  1263     1319  3      THEN
;  1264     1320  4          BEGIN
;  1265     1321  4          FAB [FAB$L_STV] = 0;            ! no valid area id to give
;  1266     1322  4          RETURN RMSERR (BKS);
;  1267     1323  3          END;
;  1268     1324  2      END;                                ! end of LAST_BKS, and MIN_BKS def
;  1269     1325  2
;  1270     1326  2      ! Loop thru the key table again to validate some fields.
;  1271     1327  2      !
;  1272     1328  2
;  1273     1329  3      INCR KEYNO FROM 0 TO ( .IFAB [IFB$B_NUM_KEYS] - 1 )
;  1274     1330  2      DO
;  1275     1331  3          BEGIN
;  1276     1332  3          FAB [FAB$L_STV] = .KEYNO;
;  1277     1333  3
;  1278     1334  3          ! Make sure the bucket size of the index level is equal to the bucket
;  1279     1335  3          ! size of the lowest index level for all keys.
;  1280     1336  3          !
;  1281     1337  3
;  1282     1338  3          IF .AREA_BKS [.KEY_TBL [.KEYNO, IAN]] NEQ .AREA_BKS [.KEY_TBL [.KEYNO, LAN]]
;  1283     1339  3          THEN
;  1284     1340  3              RETURN RMSERR (IBK);
;  1285     1341  3
;  1286     1342  3          ! Check the IFL and DFL values.  They cannot be greater than the bucket
;  1287     1343  3          ! size given in number of bytes.
;  1288     1344  3          !
;  1289     1345  3
;  1290     1346  4          IF .KEY_TBL [.KEYNO, IFL] GTR (.AREA_BKS [.KEY_TBL [.KEYNO, IAN]] * 512)
;  1291     1347  3          THEN
;  1292     1348  3              RETURN RMSERR (IFL);
;  1293     1349  3
;  1294     1350  4          IF .KEY_TBL [.KEYNO, DFL] GTR (.AREA_BKS [.KEY_TBL [.KEYNO, DAN]] * 512)
;  1295     1351  3          THEN
;  1296     1352  3              RETURN RMSERR (DFL);
;  1297     1353  2          END;
;  1298     1354  2
;  1299     1355  2      FAB [FAB$L_STV] = 0;
;  1300     1356  2
;  1301     1357  2      ! Loop thru the XAB's again to fill in the correct bucket size value in the
;  1302     1358  2      ! area XABs.
;  1303     1359  2      !
;  1304     1360  2
```

```
 2      XAB = .FAB [FAB$L_XAB];
 2
 2      WHILE .XAB NEQ 0
 2      DO
 3          BEGIN
 3          RETURN_ON_ERROR ( REPROBE () );
 3
 3          IF .XAB [XAB$B_COD] EQL XAB$C_ALL
 3          THEN
 3              XAB [XAB$B_BKZ] = .AREA_BKS [.XAB [XAB$B_AID]];
 3          XAB = .XAB [XAB$L_NXT];
 2          END;
 2
 2      RETURN RMS$SUC ();
 1      END;



                                 0ABC   8F  BB 00000 CHECK_AREAS:
                                                              PUSHR   #^M<R2,R3,R4,R5,R7,R9,R11>                 ; 0963
                           5E            08  C2 00004          SUBL2   #8, SP                                     ;
                           54        28  AE  D0 00007          MOVL    TABLES, KEY_TBL                            ; 1032
                           50      00B2  CA  9A 0000B          MOVZBL  178(IFAB), R0                              ; 1033
                           55     28 BE40    7E 00010          MOVAQ   @TABLES[R0], AREA_BKS                      ;
                                         5B  D4 00015          CLRL    BIG_BKS                                    ; 1034
                                         7E  7C 00017          CLRQ    MAX_MINREC                                 ; 1036
                           56        24  A8  D0 00019          MOVL    36(FAB), XAB                               ; 1038
                                         03  12 0001D 1$:      BNEQ    2$                                         ; 1040
                                       00C3  31 0001F          BRW     14$                                        ;
                                      0000V  30 00022 2$:      BSBW    REPROBE                                    ; 1044
                           51            50  E9 00025          BLBC    STATUS, 7$                                 ;
               0D          12            56  8F 00028          CASEB   (XAB), #18, #13                            ; 1050
    001E     0099        00B2          00B2     0002C 3$:      .WORD   13$-3$,-                                   ;
    00B2     00B2        00B2          0062     00034                  13$-3$,-                                   ;
    00B2     00B2        00B2          00B2     0003C                  12$-3$,-                                   ;
                         00B2          00B2     00044                  4$-3$,-                                    ;
                                                                       13$-3$,-                                   ;
                                                                       13$-3$,-                                   ;
                                                                       13$-3$,-                                   ;
                                                                       13$-3$,-                                   ;
                                                                       13$-3$,-                                   ;
                                                                       13$-3$,-                                   ;
                                                                       13$-3$,-                                   ;
                                                                       13$-3$,-                                   ;
                                                                       13$-3$,-                                   ;
                                                                       13$-3$                                     ;
                                         79  11 00048          BRB     11$                                        ;
                      0C   A8        17  A6  9A 0004A 4$:      MOVZBL  23(XAB), 12(FAB)                           ; 1056
                           50      00B1  CA  9A 0004F          MOVZBL  177(IFAB), R0                              ; 1058
                           50        08  A6  91 00054          CMPB    8(XAB), R0                                 ;
                                         07  1F 00058          BLSSU   5$                                         ;
                           50      8554  8F  3C 0005A          MOVZWL  #34132, R0                                 ; 1060
                                         18  11 0005F          BRB     7$                                         ;
                           50        09  A6  91 00061 5$:      CMPB    9(XAB), R0                                 ; 1062
                                         07  1F 00065          BLSSU   6$                                         ;
```

```
                    50      85AC    8F   3C 00067          MOVZWL  #34220, R0                      ; 1064
                              0B    11   0006C          BRB     7$                               ;
                    50        0A    A6   91 0006E 6$:     CMPB    0(XAB), R0                       ; 1066
                              08    1F   00072          BLSSU   8$                               ;
                    50      84BC    8F   3C 00074          MOVZWL  #33980, R0                       ; 1068
                            0263    31   00079 7$:        BRW     55$                              ;
                              09    A6   95 0007C 8$:     TSTB    9(XAB)                           ; 1073
                              05    12   0007F          BNEQ    9$                               ;
                09    A6        08    A6   90 00081          MOVB    8(XAB), 9(XAB)                   ; 1075
                            0000V   30   00086 9$:        BSBW    GET_KEY_PARMS                    ; 1088
                    50        17    A6   9A 00089          MOVZBL  23(XAB), R0                      ; 1089
                            6440    7F   0008D          PUSHAQ  (KEY_TBL)[R0]                    ;
                    9E          51    90 00090          MOVB    KEYSIZE, @(SP)+                  ;
                    52          6E    D1 00093          CMPL    MAX_MINREC, MINREC               ; 1091
                                03    18 00096          BGEQ    10$                              ;
                    6E          52    D0 00098          MOVL    MINREC, MAX_MINREC               ; 1093
                              02 A440   7F 0009B 10$:     PUSHAQ  2(KEY_TBL)[R0]                   ; 1098
                    9E        08    A6   90 0009F          MOVB    8(XAB), @(SP)+                   ;
                              03 A440   7F 000A3          PUSHAQ  3(KEY_TBL)[R0]                   ; 1099
                    9E        09    A6   90 000A7          MOVB    9(XAB), @(SP)+                   ;
                              01 A440   7F 000AB          PUSHAQ  1(KEY_TBL)[R0]                   ; 1100
                    9E        0A    A6   90 000AF          MOVB    10(XAB), @(SP)+                  ;
                              04 A440   7F 000B3          PUSHAQ  4(KEY_TBL)[R0]                   ; 1101
                    9E        1A    A6   B0 000B7          MOVW    26(XAB), @(SP)+                  ;
                              06 A440   7F 000BB          PUSHAQ  6(KEY_TBL)[R0]                   ; 1102
                    9E        1C    A6   B0 000BF          MOVW    28(XAB), @(SP)+                  ;
                                19    11   000C3 11$:     BRB     13$                              ; 1050
                04    AE          01    D0 000C5 12$:     MOVL    #1, AREA_FOUND                   ; 1111
                    50        17    A6   9A 000C9          MOVZBL  23(XAB), R0                      ; 1112
                            6045    16    A6   90 000CD          MOVB    22(XAB), (R0)[AREA_BKS]          ;
5B      16    A6                08    00   ED 000D2          CMPZV   #0, #8, 22(XAB), BIG_BKS         ; 1114
                                04    15   000D8          BLEQ    13$                              ;
                    5B        16    A6   9A 000DA          MOVZBL  22(XAB), BIG_BKS                 ; 1116
                    56        04    A6   D0 000DE 13$:     MOVL    4(XAB), XAB                      ; 1124
                            FF38    31   000E2          BRW     1$                               ; 1040
                0C  AE      00B1    CA   9A 000E5 14$:     MOVZBL  177(IFAB), 12(SP)                ; 1142
                    57          01    CE   000EB          MNEGL   #1, AREANO                       ; 1260
                              0115    31   000EE          BRW     38$                              ;
                                59    D4   000F1 15$:     CLRL    LAST_BKS                         ; 1146
                08  AE      00B2    CA   9A 000F3          MOVZBL  178(IFAB), 8(SP)                 ; 1150
                    53          01    CE   000F9          MNEGL   #1, KEYNO                        ; 1160
                              00CE    31   000FC          BRW     31$                              ;
                              02 A443   7F 000FF 16$:     PUSHAQ  2(KEY_TBL)[KEYNO]                ;
57      9E                    08    00   ED 00103          CMPZV   #0, #8, @(SP)+, AREANO           ;
                                0B    13   00108          BEQL    17$                              ;
                              03 A443   7F 0010A          PUSHAQ  3(KEY_TBL)[KEYNO]                ; 1161
57      9E                    08    00   ED 0010E          CMPZV   #0, #8, @(SP)+, AREANO           ;
                                5F    12   00113          BNEQ    24$                              ;
                    50        6443    7E   00115 17$:     MOVAQ   (KEY_TBL)[KEYNO], R0             ; 1167
                    03      00B7    CA   91 00119          CMPB    183(IFAB), #3                    ; 1165
                                0C    1E   0011E          BGEQU   18$                              ;
                    50          60    9A   00120          MOVZBL  (R0), R0                         ; 1167
          52        50          01    78   00123          ASHL    #1, R0, MIN_BKS                  ; 1168
                    52          18    C0   00127          ADDL2   #24, MIN_BKS                     ;
                                0A    11   0012A          BRB     19$                              ; 1167
                    50          60    9A   0012C 18$:     MOVZBL  (R0), R0                         ; 1170
          52        50          01    78   0012F          ASHL    #1, R0, MIN_BKS                  ; 1171
```

```
                        52          1E  C0 00133       ADDL2   #30, MIN_BKS                           :
        50              52 00000200 8F  C7 00136 19$:  DIVL3   #512, MIN_BKS, R0                      : 1175
                        52          01  A0  9E 0013E   MOVAB   1(R0), MIN_BKS                         :
                                  6745  95 00142       TSTB    (AREANO)[AREA_BKS]                     : 1182
                                    08  13 00145       BEQL    20$                                    :
  52          6745      08          00  ED 00147       CMPZV   #0, #8, (AREANO)[AREA_BKS], MIN_BKS    : 1183
                                    11  19 0014D       BLSS    21$                                    :
                        19      04  AE  E8 0014F 20$:  BLBS    AREA_FOUND, 23$                        : 1184
                                3E  A8  95 00153       TSTB    62(FAB)                                : 1185
                                    14  13 00156       BEQL    23$                                    :
  52       3E   A8      08          00  ED 00158       CMPZV   #0, #8, 62(FAB), MIN_BKS               : 1186
                                    0C  18 0015E       BGEQ    23$                                    :
                  0C    A8          53  D0 00160 21$:  MOVL    KEYNO, 12(FAB)                         : 1189
                        50    8784  8F  3C 00164       MOVZWL  #34692, R0                             : 1190
                                  0173  31 00169 22$:  BRW     55$                                    :
                        52          59  D1 0016C 23$:  CMPL    LAST_BKS, MIN_BKS                      : 1196
                                    03  18 0016F       BGEQ    24$                                    :
                        59          52  D0 00171       MOVL    MIN_BKS, LAST_BKS                      : 1198
                             01 A443  7F 00174 24$:    PUSHAQ  1(KEY_TBL)[KEYNO]                      : 1206
  57          9E        08          00  ED 00178       CMPZV   #0, #8, @(SP)+, AREANO                 :
                                    4E  12 0017D       BNEQ    31$                                    :
                                    53  D5 0017F       TSTL    KEYNO                                  : 1214
                                    13  13 00181       BEQL    25$                                    :
                                  6443  7F 00183       PUSHAQ  (KEY_TBL)[KEYNO]                       : 1216
                        50          9E  9A 00186       MOVZBL  @(SP)+, R0                             :
                        50 00000200 8F  C6 00189       DIVL2   #512, R0                               :
                        52          01  A0  9E 00190   MOVAB   1(R0), MIN_BKS                         :
                                    2F  11 00194       BRB     30$                                    :
                        07      04  AE  E9 00196 25$:  BLBC    AREA_FOUND, 26$                        : 1230
                  5E    AA    6745  90 0019A           MOVB    (AREANO)[AREA_BKS], 94(IFAB)           : 1232
                                    05  11 0019F       BRB     27$                                    :
                  5E    AA      3E  A8  90 001A1 26$:  MOVB    62(FAB), 94(IFAB)                      : 1234
                                60  AA  B5 001A6 27$:  TSTW    96(IFAB)                               : 1236
                                    06  13 001A9       BEQL    28$                                    :
                        7E      60  AA  3C 001AB       MOVZWL  96(IFAB), -(SP)                        : 1238
                                    02  11 001AF       BRB     29$                                    :
                                    6E  DD 001B1 28$:  PUSHL   MAX_MINREC                             : 1240
                        7E    00B7  CA  9A 001B3 29$:  MOVZBL  183(IFAB), -(SP)                       :
                                 0000V  30 001B8       BSBW    CHECK_MRS                              :
                        5E          08  C0 001BB       ADDL2   #8, SP                                 :
                        A8          50  E9 001BE       BLBC    STATUS, 22$                            :
                        52      5E  AA  9A 001C1       MOVZBL  94(IFAB), MIN_BKS                      : 1244
                        52          59  D1 001C5 30$:  CMPL    LAST_BKS, MIN_BKS                      : 1250
                                    03  18 001C8       BGEQ    31$                                    :
                        59          52  D0 001CA       MOVL    MIN_BKS, LAST_BKS                      : 1252
        02              53      08  AE  F2 001CD 31$:  AOBLSS  8(SP), KEYNO, 32$                      : 1150
                                    03  11 001D2       BRB     33$                                    :
                                  FF28  31 001D4 32$:  BRW     16$                                    :
                                  6745  95 001D7 33$:  TSTB    (AREANO)[AREA_BKS]                     : 1260
                                    06  12 001DA       BNEQ    34$                                    :
                  6745              59  90 001DC       MOVB    LAST_BKS, (AREANO)[AREA_BKS]           : 1262
                                    08  11 001E0       BRB     35$                                    :
  59          6745      08          00  ED 001E2 34$:  CMPZV   #0, #8, (AREANO)[AREA_BKS], LAST_BKS   : 1268
                                    09  19 001E8       BLSS    36$                                    :
                                  6745  95 001EA 35$:  TSTB    (AREANO)[AREA_BKS]                     : 1279
                                    0F  12 001ED       BNEQ    37$                                    :
                        08      04  AE  E9 001EF       BLBC    AREA_FOUND, 37$                        : 1280
```

```
                0C   A8              57   D0 001F3 36$:    MOVL    AREANO, 12(FAB)                          ; 1283
                     50        8424   8F   3C 001F7         MOVZWL  #33828, R0                               ; 1284
                                      7A   11 001FC         BRB     47$                                      ;
                     59              5B   D1 001FE 37$:    CMPL    BIG_BKS, LAST_BKS                        ; 1289
                                      03   18 00201         BGEQ    38$                                      ;
                     5B               59   D0 00203         MOVL    LAST_BKS, BIG_BKS                        ; 1291
                02   57         0C   AE   F2 00206 38$:    AOBLSS  12(SP), AREANO, 39$                      ; 1142
                                      03   11 0020B         BRB     40$                                      ;
                                  FEE1   31 0020D 39$:      BRW     15$                                      ;
                                      5B   D5 00210 40$:    TSTL    BIG_BKS                                  ; 1300
                                      26   12 00212         BNEQ    43$                                      ;
                5E   AA          3E   A8   90 00214         MOVB    62(FAB), 94(IFAB)                        ; 1303
                                      60   AA   B5 00219         TSTW    96(IFAB)                             ; 1305
                                      06   13 0021C         BEQL    41$                                      ;
                     7E          60   AA   3C 0021E         MOVZWL  96(IFAB), -(SP)                          ; 1307
                                      02   11 00222         BRB     42$                                      ;
                                      6E   DD 00224 41$:    PUSHL   MAX_MINREC                               ; 1309
                     7E        0087   CA   9A 00226 42$:    MOVZBL  183(IFAB), -(SP)                         ;
                                0000V 30   00228         BSBW    CHECK_MRS                                ;
                     5E              08   C0 0022E         ADDL2   #8, SP                                   ;
                     7F              50   E9 00231         BLBC    STATUS, 50$                              ;
                     5B          5E   AA   9A 00234         MOVZBL  94(IFAB), BIG_BKS                        ; 1310
                                      08   11 00238         BRB     44$                                      ; 1300
                3E   A8              5B   90 0023A 43$:    MOVB    BIG_BKS, 62(FAB)                         ; 1314
                5E   AA              5B   90 0023E         MOVB    BIG_BKS, 94(IFAB)                        ; 1315
                                      5B   D5 00242 44$:    TSTL    BIG_BKS                                  ; 1318
                                      0A   12 00244         BNEQ    45$                                      ;
                          0C          A8   D4 00246         CLRL    12(FAB)                                  ; 1321
                     50        841C   8F   3C 00249         MOVZWL  #33820, R0                               ; 1322
                                      63   11 0024E         BRB     50$                                      ;
                     53        00B2   CA   9A 00250 45$:    MOVZBL  178(IFAB), R3                            ; 1329
                     50              01   CE 00255         MNEGL   #1, KEYNO                                ; 1332
                                      5B   11 00258         BRB     51$                                      ;
                0C   A8              50   D0 0025A 46$:    MOVL    KEYNO, 12(FAB)                           ;
                          02   A440   7F   00 0025E         PUSHAQ  2(KEY_TBL)[KEYNO]                        ; 1338
                     51              9E   9A 00262         MOVZBL  @(SP)+, R1                               ;
                          03   A440   7F   00 00265         PUSHAQ  3(KEY_TBL)[KEYNO]                        ;
                     52              9E   9A 00269         MOVZBL  @(SP)+, R2                               ;
                   6245              6145 91 0026C         CMPB    (R1)[AREA_BKS], (R2)[AREA_BKS]           ;
                                      07   13 00271         BEQL    48$                                      ;
                     50        877C   8F   3C 00273         MOVZWL  #34684, R0                               ; 1340
                                      65   11 00278 47$:    BRB     55$                                      ;
                     51              6145 9A 0027A 48$:    MOVZBL  (R1)[AREA_BKS], R1                       ; 1346
           51        51              09   78 0027E         ASHL    #9, R1, R1                               ;
                          04   A440   7F   00 00282         PUSHAQ  4(KEY_TBL)[KEYNO]                        ;
     51   9E        10              00   ED 00286         CMPZV   #0, #16, @(SP)+, R1                      ;
                                      07   15 0028B         BLEQ    49$                                      ;
                     50        8764   8F   3C 0028D         MOVZWL  #34660, R0                               ; 1348
                                      4B   11 00292         BRB     55$                                      ;
                          01   A440   7F   00 00294 49$:    PUSHAQ  1(KEY_TBL)[KEYNO]                        ; 1350
                     51              9E   9A 00298         MOVZBL  @(SP)+, R1                               ;
                     51              6145 9A 0029B         MOVZBL  (R1)[AREA_BKS], R1                       ;
           51        51              09   78 0029F         ASHL    #9, R1, R1                               ;
                          06   A440   7F   00 002A3         PUSHAQ  6(KEY_TBL)[KEYNO]                        ;
     51   9E        10              00   ED 002A7         CMPZV   #0, #16, @(SP)+, R1                      ;
                                      07   15 002AC         BLEQ    51$                                      ;
                     50        876C   8F   3C 002AE         MOVZWL  #34668, R0                               ; 1352
```

```
                                    2A  11 002B3 50$:    BRB     55$                                     ;
                A1          50      53  F2 002B5 51$:    AOBLSS  R3, KEYNO, 46$                          ; 1329
                         0C    A8   D4 002B9            CLRL    12(FAB)                                 ; 1355
                         56    24   A8  D0 002BC        MOVL    36(FAB), XAB                            ; 1361
                                    1A  13 002C0 52$:    BEQL    54$                                     ; 1363
                               0000V    30 002C2        BSBW    REPROBE                                 ; 1366
                         17         50  E9 002C5        BLBC    STATUS, 55$                             ;
                         14         66  91 002C8        CMPB    (XAB), #20                              ; 1368
                                    09  12 002CB        BNEQ    53$                                     ;
                         50    17   A6  9A 002CD        MOVZBL  23(XAB), R0                             ; 1370
                      16 A6         6045  90 002D1      MOVB    (R0)[AREA_BKS], 22(XAB)                 ;
                         56    04   A6  D0 002D6 53$:    MOVL    4(XAB), XAB                             ; 1371
                                    E4  11 002DA        BRB     52$                                     ; 1363
                         50         01  D0 002DC 54$:    MOVL    #1, R0                                  ; 1374
                         5E         10  C0 002DF 55$:    ADDL2   #16, SP                                 ; 1375
                               0ABC  8F  BA 002E2        POPR    #^M<R2,R3,R4,R5,R7,R9,R11>              ;
                                         05 002E6        RSB                                             ;

; Routine Size:  743 bytes,    Routine Base:  RM$RMSMISC + 04C8
```

```
;  1321     1376  1 %SBTTL  'CHECK_DENSE'
;  1322     1377  1 ROUTINE CHECK_DENSE (BITS_MASK) : RL$CHECK_DENSE =
;  1323     1378  1
;  1324     1379  1 !++
;  1325     1380  1 !
;  1326     1381  1 ! CHECK_DENSE
;  1327     1382  1 !       This routine determines if the key and area XABs are dense
;  1328     1383  1 !       or not.
;  1329     1384  1 !
;  1330     1385  1 ! CALLING SEQUENCE:
;  1331     1386  1 !       CHECK_DENSE (BITS_MASK)
;  1332     1387  1 !
;  1333     1388  1 ! INPUT PARAMETERS:
;  1334     1389  1 !       BITS_MASK - Bit mask address for XABs
;  1335     1390  1 !
;  1336     1391  1 ! IMPLICIT INPUTS:
;  1337     1392  1 !       None
;  1338     1393  1 !
;  1339     1394  1 ! OUTPUT PARAMETERS:
;  1340     1395  1 !       None
;  1341     1396  1 !
;  1342     1397  1 ! IMPLICIT OUTPUTS:
;  1343     1398  1 !       None
;  1344     1399  1 !
;  1345     1400  1 ! ROUTINE VALUE:
;  1346     1401  1 !       IMX error - XABs are not dense
;  1347     1402  1 !       SUCCESS - XABs are dense
;  1348     1403  1 !
;  1349     1404  1 ! SIDE EFFECTS:
;  1350     1405  1 !       None
;  1351     1406  1 !--
;  1352     1407  1
;  1353     1408  2     BEGIN
;  1354     1409  2
;  1355     1410  2     LOCAL
;  1356     1411  2         TOTAL_BITS,
;  1357     1412  2         FLD_SIZE,
;  1358     1413  2         BIT_NUM;
;  1359     1414  2
;  1360     1415  2     TOTAL_BITS = 255;
;  1361     1416  2     FLD_SIZE = 32;
;  1362     1417  2     BIT_NUM = 0;
;  1363     1418  2
;  1364     1419  2     ! Loop until we find the first bit clear in the given field.
;  1365     1420  2     !
;  1366     1421  2
;  1367     1422  2     WHILE FFC ( BIT_NUM, FLD_SIZE, .BITS_MASK, BIT_NUM )
;  1368     1423  2     DO
;  1369     1424  3         BEGIN
;  1370     1425  3
;  1371     1426  3         ! None found, compute number of bits left to search
;  1372     1427  3         !
;  1373     1428  3         FLD_SIZE = .TOTAL_BITS - .BIT_NUM;
;  1374     1429  3
;  1375     1430  3         IF .FLD_SIZE EQL 0
;  1376     1431  3         THEN
;  1377     1432  3             RETURN RMSSUC ();
```

```
;  1378     1433  3
;  1379     1434  3          IF .FLD_SIZE GTR 32
;  1380     1435  3          THEN
;  1381     1436  3              FLD_SIZE = 32;
;  1382     1437  2          END;                                ! end of WHILE loop
;  1383     1438  2
;  1384     1439  2      ! There was a clear bit found.  Calculate number of bits left to search.
;  1385     1440  2      ! Add one to BIT_NUM since we do not want to search that one again.
;  1386     1441  2      !
;  1387     1442  2      BIT_NUM = .BIT_NUM + 1;
;  1388     1443  2      FLD_SIZE = .TOTAL_BITS - .BIT_NUM;
;  1389     1444  2
;  1390     1445  2      ! If the clear bit was the last one, then we know the xabs are dense.
;  1391     1446  2      !
;  1392     1447  2
;  1393     1448  2      IF .FLD_SIZE EQL 0
;  1394     1449  2      THEN
;  1395     1450  2          RETURN RMSSUC ();
;  1396     1451  2
;  1397     1452  2      IF .FLD_SIZE GTR 32
;  1398     1453  2      THEN
;  1399     1454  2          FLD_SIZE = 32;
;  1400     1455  2
;  1401     1456  2      ! Loop until we find the first bit set in the given field.
;  1402     1457  2      !
;  1403     1458  2
;  1404     1459  2      WHILE FFS ( BIT_NUM, FLD_SIZE, .BITS_MASK, BIT_NUM )
;  1405     1460  2      DO
;  1406     1461  3          BEGIN
;  1407     1462  3
;  1408     1463  3          ! None found, compute number of bits left to search
;  1409     1464  3          !
;  1410     1465  3          FLD_SIZE = .TOTAL_BITS - .BIT_NUM;
;  1411     1466  3
;  1412     1467  3          ! If there are not bits left, then the xabs are dense.
;  1413     1468  3          !
;  1414     1469  3
;  1415     1470  3          IF .FLD_SIZE EQL 0
;  1416     1471  3          THEN
;  1417     1472  3              RETURN RMSSUC ();
;  1418     1473  3
;  1419     1474  3          IF .FLD_SIZE GTR 32
;  1420     1475  3          THEN
;  1421     1476  3              FLD_SIZE = 32;
;  1422     1477  2          END;                                ! end of WHILE loop
;  1423     1478  2
;  1424     1479  2      ! If we are here, then the key or area xabs are not dense.
;  1425     1480  2      !
;  1426     1481  2      RETURN RMSERR (IMX);
;  1427     1482  2
;  1428     1483  1      END;


                                    52  DD 00000 CHECK_DENSE:
```

```
                                                         PUSHL   R2                                       ; 1377
                  52       FF    8F  9A 00002            MOVZBL  #255, TOTAL_BITS                         ; 1415
                  50             20  7D 00006            MOVQ    #32, FLD_SIZE                            ; 1416
 51      08   BE  50             51  EB 00009  1$:       FFC     BIT_NUM, FLD_SIZE, @BITS_MASK, BIT_NUM   ; 1422
                                 10  12 0000F            BNEQ    2$                                       ;
              50  52             51  C3 00011            SUBL3   BIT_NUM, TOTAL_BITS, FLD_SIZE            ; 1428
                                 28  13 00015            BEQL    5$                                       ; 1430
              20                 50  D1 00017            CMPL    FLD_SIZE, #32                            ; 1434
                                 ED  15 0001A            BLEQ    1$                                       ;
              50                 20  D0 0001C            MOVL    #32, FLD_SIZE                            ; 1436
                                 E8  11 0001F            BRB     1$                                       ; 1422
                                 51  D6 00021  2$:       INCL    BIT_NUM                                  ; 1442
              50  52             51  C3 00023            SUBL3   BIT_NUM, TOTAL_BITS, FLD_SIZE            ; 1443
                                 16  13 00027            BEQL    5$                                       ; 1448
              20                 50  D1 00029  3$:       CMPL    FLD_SIZE, #32                            ; 1452
                                 03  15 0002C            BLEQ    4$                                       ;
                  50             20  D0 0002E            MOVL    #32, FLD_SIZE                            ; 1454
 51      08   BE  50             51  EA 00031  4$:       FFS     BIT_NUM, FLD_SIZE, @BITS_MASK, BIT_NUM   ; 1459
                                 0B  12 00037            BNEQ    6$                                       ;
              50  52             51  C3 00039            SUBL3   BIT_NUM, TOTAL_BITS, FLD_SIZE            ; 1465
                                 EA  12 0003D            BNEQ    3$                                       ; 1470
                  50             01  D0 0003F  5$:       MOVL    #1, R0                                   ; 1472
                                 05  11 00042            BRB     7$                                       ;
                  50     856C    8F  3C 00044  6$:       MOVZWL  #34156, R0                               ; 1481
                                 04  BA 00049  7$:       POPR    #^M<R2>                                  ; 1483
                                     05 0004B            RSB                                              ;

; Routine Size:  76 bytes,    Routine Base:  RM$RMSMISC + 07AF

; 1429          1484  1
```

```
;1431     1485  1 %SBTTL  'CHECK_INPUT'
;1432     1486  1 ROUTINE CHECK_INPUT ( BITS_MASK; PROLOGUE ) : RL$CHECK_INPUT =
;1433     1487  1
;1434     1488  1 !++
;1435     1489  1 !
;1436     1490  1 ! CHECK_INPUT
;1437     1491  1 !
;1438     1492  1 !       This routine performs input checks specific to indexed sequential
;1439     1493  1 !       files.  Among these are:
;1440     1494  1 !       - record formats allowed:  VAR and FIX (UDF only if BIO access)
;1441     1495  1 !       - legal bucket size (0<=BKS<=BKT$C_MAXBKTSIZ)
;1442     1496  1 !       - key XAB checks
;1443     1497  1 !           - correct XAB length
;1444     1498  1 !           - valid key of reference
;1445     1499  1 !           - valid key size
;1446     1500  1 !           - last key character cannot be larger than minimum record size
;1447     1501  1 !           - data type is within range
;1448     1502  1 !           - check flags on primary key:  CHG and NUL not allowed
;1449     1503  1 !       - area XAB checks
;1450     1504  1 !           - correct XAB length and readable
;1451     1505  1 !           - valid area identification number
;1452     1506  1 !           - legal bucket size
;1453     1507  1 !       - both key and area XABs must be dense
;1454     1508  1 !       - there must be a primary key
;1455     1509  1 !
;1456     1510  1 ! CALLING SEQUENCE:
;1457     1511  1 !       CHECK_INPUT ( BITS_MASK; PROLOGUE )
;1458     1512  1 !
;1459     1513  1 ! INPUT PARAMETERS:
;1460     1514  1 !       Bits mask address
;1461     1515  1 !
;1462     1516  1 ! IMPLICIT INPUTS:
;1463     1517  1 !       FAB     - address of user's file access block
;1464     1518  1 !       IFAB    - address of internal RMS file access block
;1465     1519  1 !
;1466     1520  1 ! OUTPUT PARAMETERS:
;1467     1521  1 !       PROLOGUE - prologue version number
;1468     1522  1 !
;1469     1523  1 ! IMPLICIT OUTPUTS:
;1470     1524  1 !       IFAB [IFB$W_KBUFSZ]
;1471     1525  1 !       IFAB [IFB$B_BKS]
;1472     1526  1 !       IFAB [IFB$W_DEQ]
;1473     1527  1 !       IFAB [IFB$B_NUM_KEYS]
;1474     1528  1 !       IFAB [IFB$W_RTDEQ]
;1475     1529  1 !       IFAB [IFB$B_AMAX]
;1476     1530  1 !       IFAB [IFB$B_AVBN] - # of prologue VBN
;1477     1531  1 !       FAB [FAB$L_ALQ] - if area XAB
;1478     1532  1 !       FAB [FAB$W_DEQ] - if area XAB
;1479     1533  1 !
;1480     1534  1 ! ROUTINE VALUE:
;1481     1535  1 !       RFM,BKS,IMX,REF,FLG,DTP,SIZ,POS,NPK,XAB,IAN,LAN,DAN,IFL,DFL
;1482     1536  1 !       SEG,AID,IBK,KSI,BKZ,PLV,MRS, and SUC
;1483     1537  1 !
;1484     1538  1 ! SIDE EFFECTS:
;1485     1539  1 !       none
;1486     1540  1 !
;1487     1541  1 !--
```

```
;  1488          1542  1
;  1489          1543  2      BEGIN
;  1490          1544  2
;  1491          1545  2      EXTERNAL REGISTER
;  1492          1546  2          COMMON_FAB_STR,
;  1493          1547  2          XAB = 6 : REF BBLOCK;
;  1494          1548  2
;  1495          1549  2      ! Must be a valid record format for indexed seq files.  If undefined
;  1496          1550  2      ! record format, the BIO access bit must be on.
;  1497          1551  2      !
;  1498          1552  2
;  1499          1553  2      IF .IFAB [IFB$B_RFMORG] GTRU FAB$C_VAR
;  1500          1554  2      THEN
;  1501          1555  2          RETURN RMSERR (RFM);
;  1502          1556  2
;  1503          1557  2      IF   .IFAB [IFB$B_RFMORG] EQL FAB$C_UDF
;  1504          1558  2        AND
;  1505          1559  2          NOT .FAB [FAB$V_BIO]
;  1506          1560  2      THEN
;  1507          1561  2          RETURN RMSERR (RFM);
;  1508          1562  2
;  1509          1563  2      ! Bucket size cannot be larger than BKT$C_MAXBKTSIZ blocks
;  1510          1564  2      !
;  1511          1565  2
;  1512          1566  2      IF .FAB [FAB$B_BKS] GTRU BKT$C_MAXBKTSIZ
;  1513          1567  2      THEN
;  1514          1568  2          RETURN RMSERR (BKS);
;  1515          1569  2
;  1516          1570  2      IFAB [IFB$B_BKS] = .FAB [FAB$B_BKS];
;  1517          1571  2
;  1518          1572  2      ! Now do checks on KEY and AREA XABs.
;  1519          1573  2      !
;  1520          1574  2
;  1521          1575  3      BEGIN
;  1522          1576  3
;  1523          1577  3      LOCAL
;  1524          1578  3          KEY_BITS  : REF BITVECTOR [256],
;  1525          1579  3          AREA_BITS : REF BITVECTOR [256],
;  1526          1580  3          AREAS,                                  ! # of area XAB's
;  1527          1581  3          KEYS;                                   ! # of key XAB's
;  1528          1582  3
;  1529          1583  3      KEYS = 0;                                   ! initialize them
;  1530          1584  3      AREAS = 0;
;  1531          1585  3
;  1532          1586  3      CLEAR ( .BITS_MASK, 64 );                   ! clear bit mask
;  1533          1587  3      KEY_BITS = .BITS_MASK;
;  1534          1588  3      AREA_BITS = .BITS_MASK + 32;
;  1535          1589  3      XAB = .FAB [FAB$L_XAB];                     ! get start of chain
;  1536          1590  3
;  1537          1591  3      WHILE .XAB NEQ 0
;  1538          1592  3      DO
;  1539          1593  3
;  1540          1594  4          BEGIN
;  1541          1595  4
;  1542          1596  4          ! Make sure we can read and write user's XAB.  Save XAB address
;  1543          1597  4          ! in status value field (STV) to be returned in case of error.
;  1544          1598  4          !
```

```
;  1545          1599  4          RETURN_ON_ERROR ( REPROBE() );
;  1546          1600  4          FAB [FAB$C_STV] = .XAB;
;  1547          1601  4
;  1548          1602  4          CASE .XAB [XAB$B_COD] FROM XAB$C_DAT TO XAB$C_TRM OF
;  1549          1603  4              SET
;  1550          1604  4
;  1551          1605  4              [XAB$C_KEY] :                       ! key XAB
;  1552          1606  5                  BEGIN
;  1553          1607  5
;  1554          1608  5                  ! Check that XAB length is at least equal minimum
;  1555          1609  5                  !
;  1556          1610  5
;  1557          1611  5                  IF .XAB [XAB$B_BLN] LSS XAB$C_KEYLEN
;  1558          1612  5                  THEN
;  1559          1613  5                      RETURN RMSERR (XAB);
;  1560          1614  5
;  1561          1615  5                  FAB [FAB$L_STV] = .XAB [XAB$B_REF];
;  1562          1616  5
;  1563          1617  5                  ! If more than 255 keys, return error
;  1564          1618  5                  !
;  1565          1619  5
;  1566          1620  5                  IF .XAB [XAB$B_REF] EQL 255
;  1567          1621  5                  THEN
;  1568          1622  5                      RETURN RMSERR (REF);
;  1569          1623  5
;  1570          1624  6                  BEGIN
;  1571          1625  6
;  1572          1626  6                  LOCAL
;  1573          1627  6                      KEYSIZE,
;  1574          1628  6                      MINREC,
;  1575          1629  6                      SEGS;
;  1576          1630  6
;  1577          1631  6                  GET_KEY_PARMS (;KEYSIZE, MINREC, SEGS);
;  1578          1632  6                  RETURN_ON_ERROR ( CHECK_KEY_PARMS ( .KEYSIZE, .MINREC, .SEGS ) );
;  1579          1633  6
;  1580          1634  6                  ! Set key buffer size, using the size of the longest key
;  1581          1635  6                  !
;  1582          1636  6
;  1583          1637  6                  IF .KEYSIZE<0, 8> GTRU .IFAB [IFB$W_KBUFSZ]
;  1584          1638  6                  THEN
;  1585          1639  6                      IFAB [IFB$W_KBUFSZ] = .KEYSIZE;
;  1586          1640  6
;  1587          1641  6                  ! Get the prologue number from the primary key XAB.
;  1588          1642  6                  !
;  1589          1643  6
;  1590          1644  6                  IF .XAB [XAB$B_REF] EQL 0
;  1591          1645  6                  THEN
;  1592          1646  7                      BEGIN
;  1593          1647  7
;  1594          1648  7                      IF .XAB [XAB$V_CHG]  OR  .XAB [XAB$V_NUL]
;  1595          1649  7                      THEN
;  1596          1650  7                          RETURN RMSERR (FLG);
;  1597          1651  7                      RETURN_ON_ERROR ( FIND_PROLOGUE ( .SEGS; PROLOGUE ) );
;  1598          1652  6                      END;
;  1599          1653  6
;  1600          1654  5                  END;                        ! end of def of KEYSIZE, MINREC, SEGS
;  1601          1655  5
```

```
;  1602          1656  5              ! If prolog ' defined, decide if prolog 2 is required.
;  1603          1657  5              !
;  1604          1658  5
;  1605          1659  5              IF .XAB [XAB$B_DTP] NEQ XAB$C_STG
;  1606          1660  5                 AND .PROLOGUE EQL PLG$C_VER_NO
;  1607          1661  5              THEN
;  1608          1662  5                  PROLOGUE = PLG$C_VER_IDX;
;  1609          1663  5
;  1610          1664  5              ! One more key has been defined.  Set corresponding bit
;  1611          1665  5              ! in mask.
;  1612          1666  5              !
;  1613          1667  5              KEYS = .KEYS + 1;
;  1614          1668  5              KEY_BITS [ .XAB [XAB$B_REF] ] = 1;
;  1615          1669  5
;  1616          1670  4              END;                                ! of key XAB processing
;  1617          1671  4
;  1618          1672  4          [XAB$C_ALL] :                           ! area XAB
;  1619          1673  5              BEGIN
;  1620          1674  5
;  1621          1675  5              IF .XAB [XAB$P_BLN] LSS XAB$C_ALLLEN
;  1622          1676  5              THEN
;  1623          1677  5                  RETURN RMSERR (XAB);
;  1624          1678  5
;  1625          1679  5              FAB [FAB$L_STV] = .XAB [XAB$B_AID];
;  1626          1680  5
;  1627          1681  5              IF .XAB [XAB$B_AID] EQL 255
;  1628          1682  5              THEN
;  1629          1683  5                  RETURN RMSERR (AID);
;  1630          1684  5
;  1631          1685  5              IF .XAB [XAB$B_BKZ] GTRU BKT$C_MAXBKTSIZ
;  1632          1686  5              THEN
;  1633          1687  5                  RETURN RMSERR (BKZ);
;  1634          1688  5
;  1635          1689  5              ! first area XAB overrides FAB values
;  1636          1690  5              !
;  1637          1691  5
;  1638          1692  5              IF .XAB [XAB$B_AID] EQL 0
;  1639          1693  5              THEN
;  1640          1694  6                  BEGIN
;  1641          1695  6                  FAB [FAB$L_ALQ] = .XAB [XAB$L_ALQ];
;  1642          1696  6                  FAB [FAB$W_DEQ] = .XAB [XAB$W_DEQ];
;  1643          1697  5                  END;
;  1644          1698  5
;  1645          1699  5              AREAS = .AREAS + 1;
;  1646          1700  5              AREA_BITS [ .XAB [XAB$B_AID] ] = 1;
;  1647          1701  5
;  1648          1702  4              END;                            ! of area XAB processing
;  1649          1703  4
;  1650          1704  4          [INRANGE] : 0;
;  1651          1705  4          [OUTRANGE] :0;
;  1652          1706  4
;  1653          1707  4          TES;
;  1654          1708  4
;  1655          1709  4      XAB = .XAB [XAB$L_NXT];
;  1656          1710  4
;  1657          1711  3      END;                        ! end of WHILE XAB check loop
;  1658          1712  3
```

```
;  1659          1713  3      ! If we are creating with block I/O and no record I/O at all, make sure
;  1660          1714  3      ! there is at least a primary key defined.
;  1661          1715  3      !
;  1662          1716  3
;  1663          1717  4      IF NOT ( .FAB [FAB$V_BIO] AND NOT .FAB [FAB$V_BRO] )
;  1664          1718  3        AND .KEYS EQL 0
;  1665          1719  3      THEN
;  1666          1720  3          RETURN RMSERR (NPK);
;  1667          1721  3
;  1668          1722  3      ! Get key XABs density.
;  1669          1723  3      !
;  1670          1724  3
;  1671          1725  4      BEGIN
;  1672          1726  4
;  1673          1727  4      LOCAL
;  1674          1728  4          STATUS;
;  1675          1729  4
;  1676          1730  4      ! Set the STV to the XAB address for "non-dense" XAB error.
;  1677          1731  4      !
;  1678          1732  4      FAB [FAB$L_STV] = .FAB [FAB$L_XAB];
;  1679          1733  4      STATUS = CHECK_DENSE (.KEY_BITS);
;  1680          1734  4
;  1681          1735  4      ! If the keys are not dense, find out if there is a primary key.
;  1682          1736  4      ! If not there, return "no primary key" error.
;  1683          1737  4      !
;  1684          1738  4
;  1685          1739  4      IF NOT .STATUS
;  1686          1740  4      THEN
;  1687          1741  5          BEGIN
;  1688          1742  5
;  1689          1743  5          IF .KEY_BITS[0] EQL 0
;  1690          1744  5          THEN
;  1691          1745  5              STATUS = RMSERR (NPK);
;  1692          1746  5          RETURN .STATUS;
;  1693          1747  4          END;
;  1694          1748  3      END;                                    ! end of STATUS definition
;  1695          1749  3
;  1696          1750  3
;  1697          1751  3      ! If no areas defined, default to one.  Else, make sure the area
;  1698          1752  3      ! XABs defined are dense.  Set maximum number of areas, and total
;  1699          1753  3      ! number of keys.
;  1700          1754  3      !
;  1701          1755  3
;  1702          1756  3      IF .AREAS EQL 0
;  1703          1757  3      THEN
;  1704          1758  3          AREAS = 1
;  1705          1759  3      ELSE
;  1706          1760  3          RETURN_ON_ERROR ( CHECK_DENSE (.AREA_BITS) );
;  1707          1761  3
;  1708          1762  3      IFAB [IFB$B_AMAX] = .AREAS;
;  1709          1763  3      IFAB [IFB$B_NUM_KEYS] = .KEYS;
;  1710          1764  3      FAB [FAB$L_STV] = 0;
;  1711          1765  3
;  1712          1766  3      ! Determine minimum number of VBN's to allocate for prologue and force it
;  1713          1767  3      !
;  1714          1768  4      BEGIN
;  1715          1769  4
```

```
;  1716          1770  4     LOCAL
;  1717          1771  4         MIN_ALLOC;
;  1718          1772  4
;  1719          1773  4     MIN_ALLOC = ( (.AREAS + 7) / 8 ) + 1 + ( (.KEYS + 3) / 5 );
;  1720          1774  4
;  1721          1775  4     ! Save this value in area VBN because it is the number of VBNs
;  1722          1776  4     ! used by the prologue.
;  1723          1777  4     !
;  1724          1778  4     IFAB [IFB$B_AVBN] = .MIN_ALLOC;
;  1725          1779  4
;  1726          1780  4     IF .MIN_ALLOC GTRU .FAB [FAB$L_ALQ]
;  1727          1781  4     THEN
;  1728          1782  4         FAB [FAB$L_ALQ] = .MIN_ALLOC;
;  1729          1783  4
;  1730          1784  3     END;                    ! end local definition of MIN_ALLOC
;  1731          1785  3
;  1732          1786  3     ! Now that FAB [DEQ] has been defaulted from first area XAB if one,
;  1733          1787  3     ! default IFAB [DEQ] and IFAB [RTDEQ].
;  1734          1788  3     !
;  1735          1789  3     IFAB [IFB$W_DEQ] = .FAB [FAB$W_DEQ];
;  1736          1790  3     IFAB [IFB$W_RTDEQ] = .FAB [FAB$W_DEQ];
;  1737          1791  3
;  1738          1792  2     END;                    ! end local definitions of AREAS, KEYS, and others
;  1739          1793  2
;  1740          1794  2     RETURN RMSSUC();        ! everything checked out correctly
;  1741          1795  2
;  1742          1796  1     END;

                                3C  BB 00000 CHECK_INPUT:
                                                 PUSHR   #^M<R2,R3,R4,R5>                ; 1486
                             5E                  0C  C2 00002                SUBL2   #12, SP                 ;
                             02          50      AA  91 00005                CMPB    80(IFAB), #2            ; 1553
                                         0A      1A 00009                    BGTRU   1$                      ;
                                         50      AA  95 0000B                TSTB    80(IFAB)                ; 1557
                                         0C      12 0000E                    BNEQ    2$                      ;
                       07        16      A8          05  E0 00010            BBS     #5, 22(FAB), 2$         ; 1559
                             50          8664    8F  3C 00015 1$:            MOVZWL  #34404, R0              ; 1561
                                         78      11 0001A                    BRB     8$                      ;
                             3F          3E      A8  91 0001C 2$:            CMPB    62(FAB), #63            ; 1566
                                         07      1B 00020                    BLEQU   3$                      ;
                             50          841C    8F  3C 00022                MOVZWL  #33820, R0              ; 1568
                                         6B      11 00027                    BRB     8$                      ;
                             5E    AA    3E      A8  90 00029 3$:            MOVB    62(FAB), 94(IFAB)       ; 1570
                                         54      7C 0002E                    CLRQ    AREAS                   ; 1584
                             7E          40      8F  9A 00030                MOVZBL  #64, -(SP)              ; 1586
                                         24      AE  DD 00034                PUSHL   BITS_MASK               ;
                                       0000V     30 00037                    BSBW    CLEAR                   ;
                             5E                  08  C0 0003A                ADDL2   #8, SP                  ;
                       04    AE          20      AE  D0 0003D                MOVL    BITS_MASK, KEY_BITS     ; 1587
                 08    AE    20          AE      20  C1 00042                ADDL3   #32, BITS_MASK, AREA_BITS ; 1588
                             56          24      A8  D0 00048                MOVL    36(FAB), XAB            ; 1589
                                         03      12 0004C 4$:                BNEQ    5$                      ; 1591
                                       00E4      31 0004E                    BRW     25$                     ;
```

```
                                      0000V 30 00051 5$:   BSBW    REPROBE                          ; 1599
                             73          50 E9 00054       BLBC    STATUS, 13$                      ;
                       0C    A8          56 D0 00057       MOVL    XAB, 12(FAB)                     ; 1600
              0D             12          66 8F 0005B       CASEB   (XAB), #18, #13                  ; 1602
   001E       0088         00CF        00CF    0005F 6$:   .WORD   24$-6$,-                         ;
   00CF       00CF         00CF        00CF    00067               24$-6$,-                         ;
   00CF       00CF         00CF        00CF    0006F               17$-6$,-                         ;
                           00CF        00CF    00077               7$-6$,-                          ;
                                                                   24$-6$,-                         ;
                                                                   24$-6$,-                         ;
                                                                   24$-6$,-                         ;
                                                                   24$-6$,-                         ;
                                                                   24$-6$,-                         ;
                                                                   24$-6$,-                         ;
                                                                   24$-6$,-                         ;
                                                                   24$-6$,-                         ;
                                                                   24$-6$,-                         ;
                                                                   24$-6$                           ;
                                         68 11 0007B       BRB     16$                              ;
                             3F    01    A6 91 0007D 7$:   CMPB    1(XAB), #63                      ; 1611
                                         6A 1B 00081       BLEQU   18$                              ;
                       0C    A8    17    A6 9A 00083       MOVZBL  23(XAB), 12(FAB)                 ; 1615
                       FF    8F    17    A6 91 00088       CMPB    23(XAB), #255                    ; 1620
                                         07 12 0008D       BNEQ    9$                               ;
                             50  875C    8F 3C 0008F       MOVZWL  #34652, R0                       ; 1622
                                         7C 11 00094 8$:   BRB     21$                              ;
                                      0000V 30 00096 9$:   BSBW    GET_KEY_PARMS                    ; 1631
                             6E          51 D0 00099       MOVL    R1, (SP)                         ;
                                      0000V 30 0009C       BSBW    CHECK_KEY_PARMS                  ; 1632
                             70          50 E9 0009F       BLBC    STATUS, 21$                      ;
                             50          6E 9A 000A2       MOVZBL  KEYSIZE, R0                      ; 1637
                     00B4    CA          50 B1 000A5       CMPW    R0, 180(IFAB)                    ;
                                         05 1B 000AA       BLEQU   10$                              ;
                     00B4    CA          6E B0 000AC       MOVW    KEYSIZE, 180(IFAB)               ; 1639
                                   17    A6 95 000B1 10$:  TSTB    23(XAB)                          ; 1644
                                         17 12 000B4       BNEQ    14$                              ;
              05       12    A6          01 E0 000B6       BBS     #1, 18(XAB), 11$                 ; 1648
              07       12    A6          02 E1 000BB       BBC     #2, 18(XAB), 12$                 ;
                             50  851C    8F 3C 000C0 11$:  MOVZWL  #34076, R0                       ; 1650
                                         4B 11 000C5       BRB     21$                              ;
                                      0000V 30 000C7 12$:  BSBW    FIND_PROLOGUE                    ; 1651
                             45          50 E9 000CA 13$:  BLBC    STATUS, 21$                      ;
                                   13    A6 95 000CD 14$:  TSTB    19(XAB)                          ; 1659
                                         08 13 000D0       BEQL    15$                              ;
                             01          57 D1 000D2       CMPL    PROLOGUE, #1                     ; 1660
                                         03 12 000D5       BNEQ    15$                              ;
                             57          02 D0 000D7       MOVL    #2, PROLOGUE                     ; 1662
                                         55 D6 000DA 15$:  INCL    KEYS                             ; 1667
                             50    17    A6 9A 000DC       MOVZBL  23(XAB), R0                      ; 1668
              49       04    BE          50 E2 000E0       BBSS    R0, @KEY_BITS, 24$               ;
                                         47 11 000E5 16$:  BRB     24$                              ; 1602
                             20    01    A6 91 000E7 17$:  CMPB    1(XAB), #32                      ; 1675
                                         07 1E 000EB       BGEQU   19$                              ;
                             50  870C    8F 3C 000ED 18$:  MOVZWL  #34572, R0                       ; 1677
                                         69 11 000F2       BRB     29$                              ;
                       0C    A8    17    A6 9A 000F4 19$:  MOVZBL  23(XAB), 12(FAB)                 ; 1679
                       FF    8F    17    A6 91 000F9       CMPB    23(XAB), #255                    ; 1681
```

```
                                 07  12 000FE          BNEQ    20$
                          50     83F4  8F  3C 00100          MOVZWL  #33780, R0                      ; 1683
                                 56  11 00105          BRB     29$
                          3F       16  A6  91 00107 20$:    CMPB    22(XAB), #63                    ; 1685
                                 07  1B 0010B          BLEQU   22$
                          50     8424  8F  3C 0010D          MOVZWL  #33828, R0                      ; 1687
                                 49  11 00112 21$:    BRB     29$
                                   17  A6  95 00114 22$:    TSTB    23(XAB)                         ; 1692
                                 0A  12 00117          BNEQ    23$
                       10    A8    10  A6  D0 00119          MOVL    16(XAB), 16(FAB)                ; 1695
                       14    A8    14  A6  B0 0011E          MOVW    20(XAB), 20(FAB)                ; 1696
                                 54  D6 00123 23$:    INCL    AREAS                           ; 1699
                          50       17  A6  9A 00125          MOVZBL  23(XAB), R0                     ; 1700
                    00    08    BE  50  E3 00129          BBCS    R0, @AREA_BITS, 24$
                          56       04  A6  D0 0012E 24$:    MOVL    4(XAB), XAB                     ; 1709
                                FF17  31 00132          BRW     4$                              ; 1591
                    05    16    A8  05  E1 00135 25$:    BBC     #5, 22(FAB), 26$                ; 1717
                    04    16    A8  06  E1 0013A          BBC     #6, 22(FAB), 27$
                                 55  D5 0013F 26$:    TSTL    KEYS                            ; 1718
                                 15  13 00141          BEQL    28$
                       0C    A8    24  A8  D0 00143 27$:    MOVL    36(FAB), 12(FAB)                ; 1732
                                 04  AE  DD 00148          PUSHL   KEY_BITS                        ; 1733
                                FE66  30 0014B          BSBW    CHECK_DENSE
                          5E             04  C0 0014E          ADDL2   #4, SP
                          0B             50  E8 00151          BLBS    STATUS, 30$                     ; 1739
                          56       04    BE  E8 00154          BLBS    @KEY_BITS, 34$                  ; 1743
                          50     85FC    8F  3C 00158 28$:    MOVZWL  #34300, STATUS                  ; 1745
                                 4F  11 0015D 29$:    BRB     34$                             ; 1746
                                 54  D5 0015F 30$:    TSTL    AREAS                           ; 1756
                                 05  12 00161          BNEQ    31$
                          54             01  D0 00163          MOVL    #1, AREAS                       ; 1758
                                 0C  11 00166          BRB     32$
                                08    AE  DD 00168 31$:    PUSHL   AREA_BITS                       ; 1760
                                   FE46  30 0016B          BSBW    CHECK_DENSE
                          5E             04  C0 0016E          ADDL2   #4, SP
                          3A             50  E9 00171          BLBC    STATUS, 34$
                    00B1  CA             54  90 00174 32$:    MOVB    AREAS, 177(IFAB)                ; 1762
                    00B2  CA             55  90 00179          MOVB    KEYS, 178(IFAB)                 ; 1763
                                0C    A8  D4 0017E          CLRL    12(FAB)                         ; 1764
                          54             07  C0 00181          ADDL2   #7, R4                          ; 1773
                          54             08  C6 00184          DIVL2   #8, R4
                          55             03  C0 00187          ADDL2   #3, R5
                          55             05  C6 0018A          DIVL2   #5, R5
                          50       01  A544  9E 0018D          MOVAB   1(R5)[R4], MIN_ALLOC
                    00B0  CA             50  90 00192          MOVB    MIN_ALLOC, 176(IFAB)            ; 1778
                      10  A8             50  D1 00197          CMPL    MIN_ALLOC, 16(FAB)              ; 1780
                                 04  1B 0019B          BLEQU   33$
                       10    A8          50  D0 0019D          MOVL    MIN_ALLOC, 16(FAB)              ; 1782
                       62    AA    14    A8  B0 001A1 33$:    MOVW    20(FAB), 98(IFAB)               ; 1789
                       4C    AA    14    A8  B0 001A6          MOVW    20(FAB), 76(IFAB)               ; 1790
                          50             01  D0 001AB          MOVL    #1, R0                          ; 1794
                          5E             0C  C0 001AE 34$:    ADDL2   #12, SP                         ; 1796
                                 3C  BA 001B1          POPR    #^M<R2,R3,R4,R5>
                                 05 001B3          RSB

; Routine Size:  436 bytes,    Routine Base:  RM$RMSMISC + 07FB
```

```
;  1743          1797  1
;  1744          1798  1
```

```
;1746        1799  1 %SBTTL 'CHECK_KEY_PARMS'
;1747        1800  1 ROUTINE CHECK_KEY_PARMS ( KEYSIZE, MINREC, SEGS ) : RL$CHECK_KEY_PARMS =
;1748        1801  1
;1749        1802  1 !++
;1750        1803  1 !
;1751        1804  1 ! CHECK_KEY_PARMS
;1752        1805  1 !       This routine validates key size, and sets the value in the XAB if
;1753        1806  1 !       it was not defined for binary and integer types.
;1754        1807  1 !
;1755        1808  1 ! CALLING SEQUENCE:
;1756        1809  1 !       CHECK_KEY_PARMS ( KEYSIZE, MINREC, SEGS )
;1757        1810  1 !
;1758        1811  1 ! INPUT PARAMETERS:
;1759        1812  1 !       KEYSIZE - number of characters in key
;1760        1813  1 !       MINREC - minimum record size
;1761        1814  1 !       SEGS - number of segments
;1762        1815  1 !
;1763        1816  1 ! IMPLICIT INPUTS:
;1764        1817  1 !       XAB     - address of XAB to process
;1765        1818  1 !       IFAB address
;1766        1819  1 !
;1767        1820  1 ! OUTPUT PARAMETERS:
;1768        1821  1 !       None
;1769        1822  1 !
;1770        1823  1 ! IMPLICIT OUTPUTS:
;1771        1824  1 !       None
;1772        1825  1 !
;1773        1826  1 ! ROUTINE VALUE:
;1774        1827  1 !       R0 :  Success or ( SEG, SIZ, DTP, POS ) error code
;1775        1828  1 !
;1776        1829  1 ! SIDE EFFECTS:
;1777        1830  1 !       XAB [XAB$B_SIZ0] filled in if it was 0.
;1778        1831  1 !
;1779        1832  1 !--
;1780        1833  1
;1781        1834  2     BEGIN
;1782        1835  2
;1783        1836  2     EXTERNAL REGISTER
;1784        1837  2         R_IFAB_STR,
;1785        1838  2         XAB = 6 : REF BBLOCK;
;1786        1839  2
;1787        1840  2     ! If more than one segment, and key data type is not string, then
;1788        1841  2     ! return error.
;1789        1842  2     !
;1790        1843  2
;1791        1844  2     IF .SEGS GTRU 1
;1792        1845  2       AND
;1793        1846  2         .XAB [XAB$B_DTP] NEQ XAB$C_STG
;1794        1847  2     THEN
;1795        1848  2         RETURN RMSERR (SEG);
;1796        1849  2
;1797        1850  2     ! Validate key size depending on data type
;1798        1851  2     !
;1799        1852  2
;1800        1853  2     CASE .XAB [XAB$B_DTP] FROM XAB$C_STG TO XAB$C_MAXDTP OF
;1801        1854  2         SET
;1802        1855  2
```

```
;  1803     1856  2          [XAB$C_STG]:                            ! string data type
;  1804     1857  2              IF .KEYSIZE EQL 0
;  1805     1858  2                 OR  .KEYSIZE GTRU 255
;  1806     1859  2              THEN
;  1807     1860  2                  RETURN RMSERR (SIZ);
;  1808     1861  2
;  1809     1862  2          [XAB$C_IN2, XAB$C_BN2] :                ! 2-byte integer or binary
;  1810     1863  3              BEGIN
;  1811     1864  3              IF .KEYSIZE EQL 0
;  1812     1865  3              THEN
;  1813     1866  4                  BEGIN
;  1814     1867  4                  KEYSIZE = 2;
;  1815     1868  4                  XAB [XAB$B_SIZ0] = .KEYSIZE;
;  1816     1869  3                  END;
;  1817     1870  3
;  1818     1871  3              IF .KEYSIZE NEQ 2
;  1819     1872  3              THEN
;  1820     1873  3                  RETURN RMSERR (SIZ);
;  1821     1874  2              END;
;  1822     1875  2
;  1823     1876  2          [XAB$C_IN4, XAB$C_BN4] :                ! 4-byte integer or binary
;  1824     1877  3              BEGIN
;  1825     1878  3              IF .KEYSIZE EQL 0
;  1826     1879  3              THEN
;  1827     1880  4                  BEGIN
;  1828     1881  4                  KEYSIZE = 4;
;  1829     1882  4                  XAB [XAB$B_SIZ0] = .KEYSIZE;
;  1830     1883  3                  END;
;  1831     1884  3
;  1832     1885  3              IF .KEYSIZE NEQ 4
;  1833     1886  3              THEN
;  1834     1887  3                  RETURN RMSERR (SIZ);
;  1835     1888  2              END;
;  1836     1889  2
;  1837     1890  2          [XAB$C_PAC] :                           ! packed decimal string
;  1838     1891  2              IF .KEYSIZE GTRU 16                 ! max size is 31 nibbles + sign
;  1839     1892  2                OR .KEYSIZE EQL 0
;  1840     1893  2              THEN
;  1841     1894  2                  RETURN RMSERR (SIZ);
;  1842     1895  2
;  1843     1896  2          [XAB$C_IN8, XAB$C_BN8] :                ! 8-byte integer or binary
;  1844     1897  3              BEGIN
;  1845     1898  3              IF .KEYSIZE EQL 0
;  1846     1899  3              THEN
;  1847     1900  4                  BEGIN
;  1848     1901  4                  KEYSIZE = 8;
;  1849     1902  4                  XAB [XAB$B_SIZ0] = .KEYSIZE;
;  1850     1903  3                  END;
;  1851     1904  3
;  1852     1905  3              IF .KEYSIZE NEQ 8
;  1853     1906  3              THEN
;  1854     1907  3                  RETURN RMSERR (SIZ);
;  1855     1908  2              END;
;  1856     1909  2
;  1857     1910  2          [OUTRANGE] :                            ! illegal data type
;  1858     1911  2              RETURN RMSERR (DTP);
;  1859     1912  2
```

```
; 1860          1913  2         TES;
; 1861          1914  2
; 1862          1915  2       ! If the minimum record value calculated by GET_KEY_PARMS equals 0,
; 1863          1916  2       ! then the key must have been a binary or integer with 0 length.
; 1864          1917  2       ! Set MINREC correctly now that the keysize is known.
; 1865          1918  2       !
; 1866          1919  2
; 1867          1920  2       IF .MINREC EQL 0
; 1868          1921  2       THEN
; 1869          1922  2           MINREC = .KEYSIZE + .XAB [XAB$W_POS0];
; 1870          1923  2
; 1871          1924  2       ! Make sure that the maximum record size defined by user is enough to
; 1872          1925  2       ! hold the key.
; 1873          1926  2       !
; 1874          1927  2
; 1875          1928  2       IF .IFAB [IFB$W_MRS] NEQ 0
; 1876          1929  2         AND .MINREC GTRU .IFAB [IFB$W_MRS]
; 1877          1930  2       THEN
; 1878          1931  2           RETURN RMSERR (POS);
; 1879          1932  2
; 1880          1933  2       RETURN RMSSUC();
; 1881          1934  2
; 1882          1935  1       END;                                    ! end of CHECK_KEY_PARMS



                                        52  DD 00000 CHECK_KEY_PARMS:
                                                             PUSHL   R2                     ; 1800
                              01        53  D1 00002         CMPL    SEGS, #1               ; 1844
                                        0C  1B 00005         BLEQU   1$                     ;
                                   13   A6  95 00007         TSTB    19(XAB)                ; 1846
                                        07  13 0000A         BEQL    1$                     ;
                              50  8794  8F  3C 0000C         MOVZWL  #34708, R0             ; 1848
                                        6B  11 00011         BRB     13$                    ;
                    07        00   13   A6  8F 00013 1$:     CASEB   19(XAB), #0, #7        ; 1853
      0036          0026    0026     0017      00018 2$:     .WORD   3$-2$,-                ;
      0051          0051    0046     0036      00020                 4$-2$,-                ;
                                                                     4$-2$,-                ;
                                                                     6$-2$,-                ;
                                                                     6$-2$,-                ;
                                                                     8$-2$,-                ;
                                                                     9$-2$,-                ;
                                                                     9$-2$                  ;
                              50  84E4  8F  3C 00028         MOVZWL  #34020, R0             ; 1911
                                        73  11 0002D         BRB     17$                    ;
                                        51  D5 0002F 3$:     TSTL    KEYSIZE                ; 1857
                                        46  13 00031         BEQL    12$                    ;
               000000FF       8F        51  D1 00033         CMPL    KEYSIZE, #255          ; 1858
                                        44  1B 0003A         BLEQU   14$                    ;
                                        3B  11 0003C         BRB     12$                    ; 1860
                                        51  D5 0003E 4$:     TSTL    KEYSIZE                ; 1864
                                        07  12 00040         BNEQ    5$                     ;
                              51        02  D0 00042         MOVL    #2, KEYSIZE            ; 1867
                         2E   A6        51  90 00045         MOVB    KEYSIZE, 46(XAB)       ; 1868
                              02        51  D1 00049 5$:     CMPL    KEYSIZE, #2            ; 1871
```

```
                                                29  11 0004C          BRB     11$                                     ;
                                                51  D5 0004E 6$:      TSTL    KEYSIZE                                 ; 1878
                                                07  12 00050          BNEQ    7$                                      ;
                                          51    04  D0 00052          MOVL    #4, KEYSIZE                             ; 1881
                                       2E A6    51  90 00055          MOVB    KEYSIZE, 46(XAB)                        ; 1882
                                          04    51  D1 00059 7$:      CMPL    KEYSIZE, #4                             ; 1885
                                                19  11 0005C          BRB     11$                                     ;
                                          10    51  D1 0005E 8$:      CMPL    KEYSIZE, #16                            ; 1891
                                                16  1A 00061          BGTRU   12$                                     ;
                                                51  D5 00063          TSTL    KEYSIZE                                 ; 1892
                                                19  12 00065          BNEQ    14$                                     ;
                                                10  11 00067          BRB     12$                                     ; 1894
                                                51  D5 00069 9$:      TSTL    KEYSIZE                                 ; 1898
                                                07  12 0006B          BNEQ    10$                                     ;
                                          51    08  D0 0006D          MOVL    #8, KEYSIZE                             ; 1901
                                       2E A6    51  90 00070          MOVB    KEYSIZE, 46(XAB)                        ; 1902
                                          08    51  D1 00074 10$:     CMPL    KEYSIZE, #8                             ; 1905
                                                07  13 00077 11$:     BEQL    14$                                     ;
                                          50 86BC 8F  3C 00079 12$:     MOVZWL  #34492, R0                              ; 1907
                                                22  11 0007E 13$:     BRB     17$                                     ;
                                                52  D5 00080 14$:     TSTL    MINREC                                  ; 1920
                                                07  12 00082          BNEQ    15$                                     ;
                                          52    1E  A6  3C 00084          MOVZWL  30(XAB), MINREC                         ; 1922
                                          52    51  C0 00088          ADDL2   KEYSIZE, MINREC                         ;
                                             60 AA  B5 0008B 15$:     TSTW    96(IFAB)                                ; 1928
                                                0F  13 0008E          BEQL    16$                                     ;
                52    60 AA                   10  00 ED 00090          CMPZV   #0, #16, 96(IFAB), MINREC               ; 1929
                                                07  1E 00096          BGEQU   16$                                     ;
                                          50 8624 8F  3C 00098          MOVZWL  #34340, R0                              ; 1931
                                                03  11 0009D          BRB     17$                                     ;
                                          50    01  D0 0009F 16$:     MOVL    #1, R0                                  ; 1933
                                                04  BA 000A2 17$:     POPR    #^M<R2>                                 ; 1935
                                                    05 000A4          RSB                                             ;

; Routine Size:  165 bytes,    Routine Base:  RM$RMSMISC + 09AF
```

```
;1884     1936  1 %SBTTL  'CHECK_MRS'
;1885     1937  1 ROUTINE CHECK_MRS ( PROLOGUE, REC_SIZE ) : RL$CHK_AND_REPROBE =
;1886     1938  1
;1887     1939  1 !++
;1888     1940  1 !
;1889     1941  1 ! CHECK_MRS
;1890     1942  1 !
;1891     1943  1 !       This routine checks that the record size given fits in a bucket,
;1892     1944  1 !       and calculates the minimum bucket size if not specified by
;1893     1945  1 !       the user.
;1894     1946  1 !
;1895     1947  1 ! CALLING SEQUENCE:
;1896     1948  1 !       CHECK_MRS (PROLOGUE, REC_SIZE)
;1897     1949  1 !
;1898     1950  1 ! INPUT PARAMETERS:
;1899     1951  1 !       PROLOGUE :  prologue version
;1900     1952  1 !       REC_SIZE :  Maximum Record Size given by user or
;1901     1953  1 !                   Minimum record size calculated using key size/position
;1902     1954  1 !
;1903     1955  1 ! IMPLICIT INPUTS:
;1904     1956  1 !       XAB       - address of XAB to process
;1905     1957  1 !       FAB       - address of file access block
;1906     1958  1 !       IFAB      - address of internal FAB
;1907     1959  1 !
;1908     1960  1 ! OUTPUT PARAMETERS:
;1909     1961  1 !       None
;1910     1962  1 !
;1911     1963  1 ! IMPLICIT OUTPUTS:
;1912     1964  1 !       IFAB [IFB$B_BKS]
;1913     1965  1 !
;1914     1966  1 ! ROUTINE VALUE:
;1915     1967  1 !       R0 :  MRS error if record does not fit in bucket
;1916     1968  1 !
;1917     1969  1 ! SIDE EFFECTS:
;1918     1970  1 !       none
;1919     1971  1 !
;1920     1972  1 !--
;1921     1973  1
;1922     1974  2     BEGIN
;1923     1975  2
;1924     1976  2     EXTERNAL REGISTER
;1925     1977  2         R_FAB_STR,
;1926     1978  2         R_IFAB_STR,
;1927     1979  2         XAB = 6 : REF BBLOCK;
;1928     1980  2
;1929     1981  2     ! If the user did not specify a bucket size, define a maximum size
;1930     1982  2     ! to make sure MRS will fit.
;1931     1983  2     !
;1932     1984  3     BEGIN
;1933     1985  3
;1934     1986  3     LOCAL
;1935     1987  3         BUCKET_SIZE;
;1936     1988  3
;1937     1989  3     IF .IFAB [IFB$B_BKS] EQL 0
;1938     1990  3     THEN
;1939     1991  3         BUCKET_SIZE = BKT$C_MAXBKTSIZ
;1940     1992  3     ELSE
```

```
;  1941          1993  3          BUCKET_SIZE = .IFAB [IFB$B_BKS];
;  1942          1994  3
;  1943          1995  3      ! Decide if record size will fit in bucket.
;  1944          1996  3      ! For prologue 3 files, we have to consider all possible overhead due to
;  1945          1997  3      ! compression.  Only do these calculations if record size does not equal 0.
;  1946          1998  3      !
;  1947          1999  3
;  1948          2000  3      IF .REC_SIZE NEQU 0
;  1949          2001  3      THEN
;  1950          2002  4          BEGIN
;  1951          2003  4
;  1952          2004  4          LOCAL
;  1953          2005  4              BYTES_IN_BKS;
;  1954          2006  4
;  1955          2007  4          IF .PROLOGUE LSSU PLG$C_VER_3
;  1956          2008  4          THEN
;  1957          2009  5              BEGIN
;  1958          2010  5
;  1959          2011  5              BYTES_IN_BKS = ( .BUCKET_SIZE * 512 ) - BKT$C_OVERHDSZ - 1 -
;  1960          2012  5                             IRC$C_FIXOVHDSZ;                    ! -1 for check byte at end
;  1961          2013  5
;  1962          2014  5              IF .IFAB [IFB$B_RFMORG] EQL FAB$C_VAR
;  1963          2015  5              THEN
;  1964          2016  5                  BYTES_IN_BKS = .BYTES_IN_BKS - (IRC$C_VAROVHDSZ - IRC$C_FIXOVHDSZ);
;  1965          2017  5              END
;  1966          2018  4          ELSE
;  1967          2019  4              BYTES_IN_BKS = ( .BUCKET_SIZE * 512 ) - BKT$C_OVERHDSZ
;  1968          2020  4                             - BKT$C_DATBKTOVH - IRC$C_VAROVHSZ3
;  1969          2021  4                             - IRC$C_KEYCMPOVH - IRC$C_DATCMPOVH;
;  1970          2022  4
;  1971          2023  5          IF .REC_SIZE GTRU WORDMASK(.BYTES_IN_BKS)
;  1972          2024  4          THEN
;  1973          2025  4              RETURN RMSERR (MRS);
;  1974          2026  3          END;                                    ! end of MRS check
;  1975          2027  2      END;                                        ! end of BUCKET_SIZE definition
;  1976          2028  2
;  1977          2029  2      ! If the user did not specify a bucket size, find the minimum size,
;  1978          2030  2      ! based on the record size, which was given.
;  1979          2031  2      !
;  1980          2032  2
;  1981          2033  2      IF .IFAB [IFB$B_BKS] EQL 0
;  1982          2034  2      THEN
;  1983          2035  3          BEGIN
;  1984          2036  3
;  1985          2037  3          LOCAL
;  1986          2038  3              BYTES;                              ! # of bytes to hold one record
;  1987          2039  3
;  1988          2040  3          IF .PROLOGUE LSSU PLG$C_VER_3
;  1989          2041  3          THEN
;  1990          2042  4              BEGIN
;  1991          2043  4              BYTES = .REC_SIZE + BKT$C_OVERHDSZ + IRC$C_FIXOVHDSZ;
;  1992          2044  4
;  1993          2045  4              IF .IFAB [IFB$B_RFMORG] EQL FAB$C_VAR
;  1994          2046  4              THEN
;  1995          2047  4                  BYTES = .BYTES + (IRC$C_VAROVHDSZ - IRC$C_FIXOVHDSZ);
;  1996          2048  4
;  1997          2049  4              END
```

```
; 1998      2050  3          ELSE
; 1999      2051  3
; 2000      2052  3              ! Assume all kinds of compressions on for prologue 3 files...
; 2001      2053  3              !
; 2002      2054  3              BYTES = .REC_SIZE + BKT$C_OVERHDSZ + BKT$C_DATBKTOVH
; 2003      2055  3                      + IRC$C_VAROVHSZ3 + IRC$C_KEYCMPOVH + IRC$C_DATCMPOVH;
; 2004      2056  3
; 2005      2057  3          IFAB [IFB$B_BKS] = ( .BYTES / 512 ) + 1;
; 2006      2058  3
; 2007      2059  3          ! If the bucket size we found is only equal to 1, optimize space
; 2008      2060  3            and performance by setting it to 2.
; 2009      2061  3
; 2010      2062  3          .
; 2011      2063  3          IF .IFAB [IFB$B_BKS] EQL 1
; 2012      2064  3          THEN
; 2013      2065  3              IFAB [IFB$B_BKS] = 2;
; 2014      2066  2          END;                                ! end of bucket size defaulting
; 2015      2067  2
; 2016      2068  2      RETURN RMS$SUC();
; 2017      2069  1      END;



                                0C  BB 00000 CHECK_MRS:
                                                PUSHR   #^M<R2,R3>                              ; 1937
                                53  D4 00002    CLRL    R3                                      ; 1989
                           5E   AA  95 00004    TSTB    94(IFAB)                                ;
                                07  12 00007    BNEQ    1$                                      ;
                                53  D6 00009    INCL    R3                                      ;
                   50           3F  D0 0000B    MOVL    #63, BUCKET_SIZE                        ; 1991
                                04  11 0000E    BRB     2$                                      ;
                   50      5E   AA  9A 00010 1$: MOVZBL 94(IFAB), BUCKET_SIZE                  ; 1993
                   52      10   AE  D0 00014 2$: MOVL   REC_SIZE, R2                            ; 2000
                                2F  13 00018    BEQL    5$                                      ;
           50      50           09  78 0001A    ASHL    #9, R0, R0                              ; 2011
                   51      F2   A0  9E 0001E    MOVAB   -14(R0), R1                             ;
                   03      0C   AE  D1 00022    CMPL    PROLOGUE, #3                            ; 2007
                                0F  1E 00026    BGEQU   3$                                      ;
                   50      F8   A1  9E 00028    MOVAB   -8(R1), BYTES_IN_BKS                    ; 2011
                   02      50   AA  91 0002C    CMPB    80(IFAB), #2                            ; 2014
                                09  12 00030    BNEQ    4$                                      ;
                   50           02  C2 00032    SUBL2   #2, BYTES_IN_BKS                        ; 2016
                                04  11 00035    BRB     4$                                      ; 2007
                   50      EE   A1  9E 00037 3$: MOVAB  -18(R1), BYTES_IN_BKS                   ; 2021
   52      50      10           00  ED 0003B 4$: CMPZV  #0, #16, BYTES_IN_BKS, R2               ; 2023
                                07  1E 00040    BGEQU   5$                                      ;
                   50    85D4   8F  3C 00042    MOVZWL  #34260, R0                              ; 2025
                                35  11 00047    BRB     9$                                      ;
                   2F           53  E9 00049 5$: BLBC   R3, 8$                                  ; 2033
                   03      0C   AE  D1 0004C    CMPL    PROLOGUE, #3                            ; 2040
                                0F  1E 00050    BGEQU   6$                                      ;
                   50      15   A2  9E 00052    MOVAB   21(R2), BYTES                           ; 2043
                   02      50   AA  91 00056    CMPB    80(IFAB), #2                            ; 2045
                                09  12 0005A    BNEQ    7$                                      ;
                   50           02  C0 0005C    ADDL2   #2, BYTES                               ; 2047
```

```
                                    04  11 0005F          BRB     7$                                      ; 2040
                          50  20    A2  9E 00061 6$:      MOVAB   32(R2), BYTES                           ; 2055
                          50  00000200  8F  C6 00065 7$:  DIVL2   #512, R0                               ; 2057
                5E  AA    50        01  81 0006C         ADDB3   #1, R0, 94(IFAB)                        ;
                          01  5E    AA  91 00071         CMPB    94(IFAB), #1                            ; 2063
                                    04  12 00075         BNEQ    8$                                      ;
                      5E  AA        02  90 00077         MOVB    #2, 94(IFAB)                            ; 2065
                          50        01  D0 0007B 8$:     MOVL    #1, R0                                  ; 2068
                                    0C  BA 0007E 9$:     POPR    #^M<R2,R3>                              ; 2069
                                        05 00080         RSB                                             ;

; Routine Size:  129 bytes,    Routine Base:  RM$RMSMISC + 0A54

; 2018          2070  1
```

```
;2020          2071  1 %SBTTL 'CHECK_OVERLAP'
;2021          2072  1 ROUTINE CHECK_OVERLAP ( NO_SEGS ) : RL$CHECK_OVERLAP =
;2022          2073  1
;2023          2074  1 !++
;2024          2075  1 !
;2025          2076  1 ! CHECK_OVERLAP
;2026          2077  1 !
;2027          2078  1 !       This routine checks if the segments of the primary key for
;2028          2079  1 !       prologue 3 files overlap, or not.
;2029          2080  1 !
;2030          2081  1 ! CALLING SEQUENCE:
;2031          2082  1 !       CHECK_OVERLAP(NO_SEGS)
;2032          2083  1 !
;2033          2084  1 ! INPUT PARAMETERS:
;2034          2085  1 !       NO_SEGS :  number of segments in primary key
;2035          2086  1 !
;2036          2087  1 ! IMPLICIT INPUTS:
;2037          2088  1 !       XAB      - address of XAB to process
;2038          2089  1 !
;2039          2090  1 ! OUTPUT PARAMETERS:
;2040          2091  1 !       None
;2041          2092  1 !
;2042          2093  1 ! IMPLICIT OUTPUTS:
;2043          2094  1 !       none
;2044          2095  1 !
;2045          2096  1 ! ROUTINE VALUE:
;2046          2097  1 !       R0 :  SUC - segments do not overlap (success)
;2047          2098  1 !             SEG error code - segments overlap
;2048          2099  1 !
;2049          2100  1 ! SIDE EFFECTS:
;2050          2101  1 !       none
;2051          2102  1 !
;2052          2103  1 !--
;2053          2104  1
;2054          2105  2     BEGIN
;2055          2106  2
;2056          2107  2     EXTERNAL REGISTER
;2057          2108  2         XAB = 6 : REF BBLOCK;
;2058          2109  2
;2059          2110  2     LOCAL
;2060          2111  2         START1,
;2061          2112  2         END1,
;2062          2113  2         START2,
;2063          2114  2         END2,
;2064          2115  2         LEN     : REF VECTOR [, BYTE],
;2065          2116  2         POS     : REF VECTOR [, WORD];
;2066          2117  2
;2067          2118  2     POS = XAB [XAB$W_POS];
;2068          2119  2     LEN = XAB [XAB$B_SIZ];
;2069          2120  2
;2070          2121  2     ! To find out if any segments in the primary key overlap, we must
;2071          2122  2     ! find the starting and ending position of a given segment, and
;2072          2123  2     ! compare these values with the starting and ending position of all
;2073          2124  2     ! other segments in the key.
;2074          2125  2     !
;2075          2126  2
;2076          2127  3     INCR I FROM 0 TO (.NO_SEGS - 2)
```

```
;  2077          2128  2      DO
;  2078          2129  3          BEGIN
;  2079          2130  3          START1 = .POS[.I];
;  2080          2131  3          END1 = .POS[.I] + .LEN[.I] - 1;
;  2081          2132  3
;  2082          2133  4          INCR J FROM .I+1 TO (.NO_SEGS - 1)
;  2083          2134  3          DO
;  2084          2135  4              BEGIN
;  2085          2136  4              START2 = .POS[.J];
;  2086          2137  4              END2 = .POS[.J] + .LEN[.J] - 1;
;  2087          2138  4
;  2088          2139  5              IF ((.START1 GEQ .START2) AND (.START1 LEQ .END2))
;  2089          2140  5                OR ((.END1 GEQ .START2) AND (.END1 LEQ .END2))
;  2090          2141  5                   OR ((.START1 LEQ .START2) AND (.END1 GEQ .END2))
;  2091          2142  4              THEN
;  2092          2143  4                  RETURN RMSERR (SEG);
;  2093          2144  3              END;                            ! end of INCR J loop
;  2094          2145  2          END;                                ! end of INCR I loop
;  2095          2146  2
;  2096          2147  2      RETURN RMSSUC();                        ! return success
;  2097          2148  2
;  2098          2149  1      END;


                        0FBC    8F  BB 00000 CHECK_OVERLAP:
                                             PUSHR   #^M<R2,R3,R4,R5,R7,R8,R9,R10,R11>          ; 2072
                    52    1E    A6  9E 00004         MOVAB   30(R6), POS                        ; 2118
                    57    2E    A6  9E 00008         MOVAB   46(R6), LEN                        ; 2119
                    5B    FE    A3  9E 0000C         MOVAB   -2(R3), R11                        ; 2127
                                53  D7 00010         DECL    R3                                 ; 2133
                    51          01  CE 00012         MNEGL   #1, I                              ;
                                54  10 00015         BSBB    7$
                    5A        6241  3C 00017 1$:     MOVZWL  (POS)[I], START1                   ; 2130
                    58        6241  3C 0001B         MOVZWL  (POS)[I], R8                       ; 2131
                    50        6147  9A 0001F         MOVZBL  (I)[LEN], R0
                    58          50  C0 00023         ADDL2   R0, R8
                    54    FF    A8  9E 00026         MOVAB   -1(R8), END1
                    50          51  D0 0002A         MOVL    I, J                               ; 2139
                                38  11 0002D         BRB     6$
                    59        6240  3C 0002F 2$:     MOVZWL  (POS)[J], START2                   ; 2136
                    58        6240  3C 00033         MOVZWL  (POS)[J], R8                       ; 2137
                    6E        6047  9A 00037         MOVZBL  (J)[LEN], (SP)
                    58          6E  C0 0003B         ADDL2   (SP), R8
                    55    FF    A8  9E 0003E         MOVAB   -1(R8), END2
                    59          5A  D1 00042         CMPL    START1, START2                     ; 2139
                                05  19 00045         BLSS    3$
                    55          5A  D1 00047         CMPL    START1, END2
                                14  15 0004A         BLEQ    5$
                    59          54  D1 0004C 3$:     CMPL    END1, START2                       ; 2140
                                05  19 0004F         BLSS    4$
                    55          54  D1 00051         CMPL    END1, END2
                                0A  15 00054         BLEQ    5$
                    59          5A  D1 00056 4$:     CMPL    START1, START2                     ; 2141
                                0C  14 00059         BGTR    6$
```

```
                      55              54  D1 0005B          CMPL    END1, END2                                          ;
                                      07  19 0005E          BLSS    6$                                                  ;
                      50      8794    8F  3C 00060 5$:      MOVZWL  #34708, R0                                          ; 2143
                                      0B  11 00065          BRB     8$                                                  ;
                  C4  50              53  F3 00067 6$:      AOBLEQ  R3, J, 2$                                           ; 2133
                  A8  51              5B  F3 0006B 7$:      AOBLEQ  R11, I, 1$                                          ; 2127
                      50              01  D0 0006F          MOVL    #1, R0                                              ; 2147
                      5E              04  C0 00072 8$:      ADDL2   #4, SP                                              ; 2149
                              0FBC    8F  BA 00075          POPR    #^M<R2,R3,R4,R5,R7,R8,R9,R10,R11>                   ;
                                          05 00079          RSB                                                         ;

; Routine Size:  122 bytes,    Routine Base:  RM$RMSMISC + 0AD5
```

```
;2100      2150  1 %SBTTL 'CHECK_PROLOG'
;2101      2151  1 ROUTINE CHECK_PROLOG (PROLOG, NO_SEGS; PROLOGUE) : RL$CHECK_PROLOG =
;2102      2152  1
;2103      2153  1 !++
;2104      2154  1 !
;2105      2155  1 ! CHECK_PROLOG
;2106      2156  1 !
;2107      2157  1 !       This routine finds the highest prologue that can
;2108      2158  1 !       handle this key XAB, starting with the value in PROLOG.
;2109      2159  1 !
;2110      2160  1 ! CALLING SEQUENCE:
;2111      2161  1 !       CHECK_PROLOG (PROLOG, NO_SEGS; PROLOGUE)
;2112      2162  1 !
;2113      2163  1 ! INPUT PARAMETERS:
;2114      2164  1 !       PROLOG - prologue number to start with
;2115      2165  1 !       NO_SEGS - number of segments in key
;2116      2166  1 !
;2117      2167  1 ! IMPLICIT INPUTS:
;2118      2168  1 !       XAB - address of XAB to process
;2119      2169  1 !
;2120      2170  1 ! OUTPUT PARAMETERS:
;2121      2171  1 !       PROLOGUE
;2122      2172  1 !
;2123      2173  1 ! IMPLICIT OUTPUTS:
;2124      2174  1 !       None
;2125      2175  1 !
;2126      2176  1 ! ROUTINE VALUE:
;2127      2177  1 !       Error status or success
;2128      2178  1 !
;2129      2179  1 ! SIDE EFFECTS:
;2130      2180  1 !       None
;2131      2181  1 !--
;2132      2182  1
;2133      2183  2 BEGIN
;2134      2184  2
;2135      2185  2 EXTERNAL REGISTER
;2136      2186  2     R_IFAB_STR,
;2137      2187  2     R_FAB,
;2138      2188  2     XAB = 6;
;2139      2189  2
;2140      2190  2 LOCAL
;2141      2191  2     STATUS;
;2142      2192  2
;2143      2193  2 STATUS = RMSERR (PLV);
;2144      2194  2
;2145      2195  2 DECR I FROM .PROLOG TO 1 DO
;2146      2196  2
;2147      2197  3     BEGIN
;2148      2198  3     CASE .I FROM 1 TO 3 OF
;2149      2199  3
;2150      2200  3         SET
;2151      2201  3
;2152      2202  3         [1,2] : ! Set to prologue 1 and return success
;2153      2203  3                 !
;2154      2204  4                 BEGIN
;2155      2205  4                 PROLOGUE = PLG$C_VER_NO;
;2156      2206  4                 RETURN RMSSUC();
```

```
;  2157     2207  3                    END;
;  2158     2208  3
;  2159     2209  3            [3] :   ! Prologue 3:  if all checks succeed, the prologue version
;  2160     2210  3                    ! becomes 3, otherwise, we try a lower version.
;  2161     2211  3                    !
;  2162     2212  4                    BEGIN
;  2163     2213  4                    PROLOGUE = PLG$C_VER_3;
;  2164     2214  4                    STATUS = CHECK_OVERLAP (.NO_SEGS);
;  2165     2215  4
;  2166     2216  4                    ! If the keys do not overlap, we must check for maximum record
;  2167     2217  4                    ! size here too, since we do not want to break old programs
;  2168     2218  4                    ! that do not specify a prologue number, but have big MRS set.
;  2169     2219  4                    !
;  2170     2220  4
;  2171     2221  4                    IF .STATUS
;  2172     2222  4                    THEN
;  2173     2223  5                        BEGIN
;  2174     2224  5                        STATUS = CHECK_MRS (PLG$C_VER_3, .IFAB [IFB$W_MRS]);
;  2175     2225  5
;  2176     2226  5                        IF .STATUS
;  2177     2227  5                        THEN
;  2178     2228  5                            RETURN .STATUS;
;  2179     2229  4                        END;
;  2180     2230  4
;  2181     2231  3                    END;
;  2182     2232  3
;  2183     2233  3            [OUTRANGE] : ! Invalid prologue number
;  2184     2234  3                    RETURN RMSERR (PLV);
;  2185     2235  3
;  2186     2236  3                TES;
;  2187     2237  2            END;
;  2188     2238  2
;  2189     2239  2 ! If we got here, no prologue worked
;  2190     2240  2 !
;  2191     2241  2 RETURN .STATUS;
;  2192     2242  1 END;


                                                52  DD 00000 CHECK_PROLOG:
                                                              PUSHL   R2                              ; 2151
                                      51    872C    8F  3C 00002          MOVZWL  #34604, STATUS    ; 2193
                                      52      01    A0  9E 00007          MOVAB   1(R0), I          ; 2195
                                                    37  11 0000B          BRB     5$                ;
                                02              01    52  CF 0000D 1$:      CASEL   I, #1, #2         ; 2198
                              0015            000D        000D    00011 2$:      .WORD   3$-2$,-    ;
                                                                                    3$-2$,-    ;
                                                                                    4$-2$      ;
                                      50    872C    8F  3C 00017          MOVZWL  #34604, R0        ; 2234
                                                    2C  11 0001C          BRB     7$                ;
                                      57            01  D0 0001E 3$:      MOVL    #1, PROLOGUE      ; 2205
                                      50            01  D0 00021          MOVL    #1, R0            ; 2206
                                                    24  11 00024          BRB     7$                ;
                                      57            03  D0 00026 4$:      MOVL    #3, PROLOGUE      ; 2213
                                                FF5A  30 00029          BSBW    CHECK_OVERLAP     ; 2214
```

```
                51              50  D0 0002C          MOVL    R0, STATUS                              ;
                12              51  E9 0002F          BLBC    STATUS, 5$                              ; 2221
                7E        60    AA  3C 00032          MOVZWL  96(IFAB)  -(SP)                         ; 2224
                                03  DD 00036          PUSHL   #3                                      ;
                              FECA  30 00038          BSBW    CHECK_MRS                               ;
                5E              08  C0 0003B          ADDL2   #8, SP                                  ;
                51              50  D0 0003E          MOVL    R0, STATUS                              ;
                03              51  E8 00041          BLBS    STATUS, 6$                              ; 2226
                C6              52  F5 00044 5$:      SOBGTR  I, 1$                                   ; 2195
                50              51  D0 00047 6$:      MOVL    STATUS, R0                              ; 2241
                                04  BA 0004A 7$:      POPR    #^M<R2>                                 ; 2242
                                    05 0004C          RSB                                             ;

; Routine Size:  77 bytes,    Routine Base:  RM$RMSMISC + 0B4F                                        ; F

; 2193          2243  1
```

```
;  2195          2244  1 %SBTTL  'CLEAR'
;  2196          2245  1 ROUTINE CLEAR ( BUFFER, SIZE ) : RL$CLEAR NOVALUE =
;  2197          2246  1
;  2198          2247  1 !++
;  2199          2248  1 !
;  2200          2249  1 ! CLEAR
;  2201          2250  1 !       This routine fills the buffer with zeros.  It is a separate routine
;  2202          2251  1 !       so as not to clobber R4 and R5.
;  2203          2252  1 !
;  2204          2253  1 ! CALLING SEQUENCE:
;  2205          2254  1 !       CLEAR (BUFFER, SIZE)
;  2206          2255  1 !
;  2207          2256  1 ! INPUT PARAMETERS:
;  2208          2257  1 !       buffer to clear
;  2209          2258  1 !       number of bytes to clear
;  2210          2259  1 !
;  2211          2260  1 ! IMPLICIT INPUTS:
;  2212          2261  1 !       BKT_ADDR - address of buffer to initialize
;  2213          2262  1 !
;  2214          2263  1 ! OUTPUT PARAMETERS:
;  2215          2264  1 !       none
;  2216          2265  1 !
;  2217          2266  1 ! IMPLICIT OUTPUTS:
;  2218          2267  1 !       none
;  2219          2268  1 !
;  2220          2269  1 ! ROUTINE VALUE:
;  2221          2270  1 !       none
;  2222          2271  1 !
;  2223          2272  1 ! SIDE EFFECTS:
;  2224          2273  1 !       none
;  2225          2274  1 !
;  2226          2275  1 !--
;  2227          2276  1
;  2228          2277  2     BEGIN
;  2229          2278  2     CH$FILL ( 0, .SIZE, .BUFFER );
;  2230          2279  1     END;


                                                3C  BB 00000 CLEAR:  PUSHR   #^M<R2,R3,R4,R5>                    ;  2245
      18   AE             00            6E                         MOVC5   #0, (SP), #0, SIZE, @BUFFER          ;  2278
                                                14  BE     00008
                                                3C  BA 0000A         POPR    #^M<R2,R3,R4,R5>                    ;  2279
                                                    05 0000C         RSB                                         ;

; Routine Size:  13 bytes,    Routine Base:  RM$RMSMISC + 0B9C

;  2231          2280  1
```

```
;  2233     2281  1 %SBTTL  'EXTEND'
;  2234     2282  1 ROUTINE EXTEND (BUF,XAB) : RL$EXTEND =
;  2235     2283  1
;  2236     2284  1 !++
;  2237     2285  1 !
;  2238     2286  1 ! EXTEND
;  2239     2287  1 !       This routine is an interface to RM$EXTEND0.  It is called to
;  2240     2288  1 !       extend a file area based on the allocation specified in the
;  2241     2289  1 !       allocation XAB.
;  2242     2290  1 !
;  2243     2291  1 ! CALLING SEQUENCE:
;  2244     2292  1 !       EXTEND (BUF,XAB)
;  2245     2293  1 !
;  2246     2294  1 ! INPUT PARAMETERS:
;  2247     2295  1 !       BUF     - address of area buffer
;  2248     2296  1 !       XAB address
;  2249     2297  1 !
;  2250     2298  1 ! IMPLICIT INPUTS:
;  2251     2299  1 !       FAB address
;  2252     2300  1 !       IFAB address
;  2253     2301  1 !
;  2254     2302  1 ! OUTPUT PARAMETERS:
;  2255     2303  1 !       BUF [AREA$L_CVNB]       - start VBN
;  2256     2304  1 !       BUF [AREA$L_NXTVBN]     - start VBN
;  2257     2305  1 !       BUF [AREA$L_CNBLK]      - number of VBNs in extent
;  2258     2306  1 !       BUF [AREA$L_TOTAL_ALLOC]- total number of VBNs in area
;  2259     2307  1 !
;  2260     2308  1 ! IMPLICIT OUTPUTS:
;  2261     2309  1 !       none
;  2262     2310  1 !
;  2263     2311  1 ! ROUTINE VALUE:
;  2264     2312  1 !       Success or error code (ALN or AOP)
;  2265     2313  1 !
;  2266     2314  1 ! SIDE EFFECTS:
;  2267     2315  1 !       none
;  2268     2316  1 !
;  2269     2317  1 !--
;  2270     2318  1
;  2271     2319  2     BEGIN
;  2272     2320  2
;  2273     2321  2     MAP
;  2274     2322  2         BUF     : REF BBLOCK,
;  2275     2323  2         XAB     : REF BBLOCK;
;  2276     2324  2
;  2277     2325  2     EXTERNAL REGISTER
;  2278     2326  2         COMMON_FAB_STR;
;  2279     2327  2
;  2280     2328  2     LOCAL
;  2281     2329  2         EXTEND_QT;
;  2282     2330  2
;  2283     2331  2     EXTEND_QT = .XAB [XAB$L_ALQ];
;  2284     2332  2
;  2285     2333  3     BEGIN
;  2286     2334  3
;  2287     2335  3     LOCAL
;  2288     2336  3         STARTVBN,
;  2289     2337  3         ENDVBNP1;
```

```
;  2290          2338  3
;  2291          2339  3     RETURN_ON_ERROR ( RMS$EXTEND0 ( .EXTEND_QT, .ENDVBNP1; STARTVBN, ENDVBNP1 ));
;  2292          2340  3
;  2293          2341  3     ! Set up IFAB EBK and HBK data
;  2294          2342  3     !
;  2295          2343  3     IFAB [IFB$L_EBK] = .ENDVBNP1;
;  2296          2344  3     IFAB [IFB$L_HBK] = .ENDVBNP1 - 1;
;  2297          2345  3
;  2298          2346  3     BUF [AREA$L_CVBN]   = .STARTVBN;
;  2299          2347  3     BUF [AREA$L_NXTVBN] = .STARTVBN;
;  2300          2348  3     BUF [AREA$L_CNBLK]  = .ENDVBNP1 - .STARTVBN
;  2301          2349  3
;  2302          2350  2     END;
;  2303          2351  2
;  2304          2352  2     BUF [AREA$L_TOTAL_ALLOC] = .BUF [AREA$L_CNBLK];
;  2305          2353  2     XAB [XAB$L_ALQ] = .BUF [AREA$L_CNBLK];
;  2306          2354  2     FAB [FAB$L_ALQ] = .FAB [FAB$L_ALQ] + .XAB [XAB$L_ALQ];
;  2307          2355  2
;  2308          2356  2     RETURN RMSSUC ();
;  2309          2357  2
;  2310          2358  1     END;
; INFO#250                L1:2339
; Referenced LOCAL symbol ENDVBNP1 is probably not initialized


                                    0078   8F  BB 00000 EXTEND: PUSHR   #^M<R3,R4,R5,R6>                          ;  2282
                                           56  DD 00004         PUSHL   R6
                              50           6E  10 C1 00006         ADDL3   #16, XAB, R0                           ;  2331
                                           55  60 D0 0000A         MOVL    (R0), EXTEND_QT
                                 00000000G     00 16 0000D         JSB     RM$EXTEND0                             ;  2339
                                           2E  50 E9 00013         BLBC    STATUS, 1$
                                    74     AA  56 D0 00016         MOVL    ENDVBNP1, 116(IFAB)                    ;  2343
                                    70     AA  FF A6 9E 0001A      MOVAB   -1(R6), 112(IFAB)                      ;  2344
                                    0C     A7  51 D0 0001F         MOVL    STARTVBN, 12(BUF)                      ;  2346
                                    18     A7  51 D0 00023         MOVL    STARTVBN, 24(BUF)                      ;  2347
                                10  A7         56 C3 00027         SUBL3   STARTVBN, ENDVBNP1, 16(BUF)            ;  2348
                                    32     A7  10 A7 D0 0002C      MOVL    16(BUF), 50(BUF)                       ;  2352
                              50           6E  10 C1 00031         ADDL3   #16, XAB, R0                           ;  2353
                                           60  10 A7 D0 00035      MOVL    16(BUF), (R0)
                              50           6E  10 C1 00039         ADDL3   #16, XAB, R0                           ;  2354
                                    10     A8  60 C0 0003D         ADDL2   (R0), 16(FAB)
                                           50  01 D0 00041         MOVL    #1, R0                                 ;  2356
                                           5E  04 C0 00044 1$:     ADDL2   #4, SP                                 ;  2358
                                    0078   8F  BA 00047         POPR    #^M<R3,R4,R5,R6>
                                               05 0004B         RSB

; Routine Size:  76 bytes,    Routine Base:  RM$RMSMISC + 0BA9
```

```
;2312     2359  1 %SBTTL  'FIND_PROLOGUE'
;2313     2360  1 ROUTINE FIND_PROLOGUE ( NO_SEGS; PROLOGUE ) : RL$FIND_PROLOGUE =
;2314     2361  1
;2315     2362  1 !++
;2316     2363  1 !
;2317     2364  1 ! FIND_PROLOGUE
;2318     2365  1 !       This routine examines the key XAB and determines the prologue
;2319     2366  1 !       version depending on a series of parameters.
;2320     2367  1 !
;2321     2368  1 ! CALLING SEQUENCE:
;2322     2369  1 !       FIND_PROLOGUE ( NO_SEGS; PROLOGUE )
;2323     2370  1 !
;2324     2371  1 ! INPUT PARAMETERS:
;2325     2372  1 !       NO_SEGS - number of segments in key
;2326     2373  1 !
;2327     2374  1 ! IMPLICIT INPUTS:
;2328     2375  1 !       XAB     - address of XAB to process
;2329     2376  1 !
;2330     2377  1 ! OUTPUT PARAMETERS:
;2331     2378  1 !       PROLOGUE version number
;2332     2379  1 !
;2333     2380  1 ! IMPLICIT OUTPUTS:
;2334     2381  1 !       None
;2335     2382  1 !
;2336     2383  1 ! ROUTINE VALUE:
;2337     2384  1 !       R0 :  Success or PLV error code
;2338     2385  1 !
;2339     2386  1 ! SIDE EFFECTS:
;2340     2387  1 !       none
;2341     2388  1 !
;2342     2389  1 !--
;2343     2390  1
;2344     2391  2     BEGIN
;2345     2392  2
;2346     2393  2     EXTERNAL REGISTER
;2347     2394  2         R_IFAB,
;2348     2395  2         R_FAB,
;2349     2396  2         XAB = 6 :  REF BBLOCK;
;2350     2397  2
;2351     2398  2     LOCAL
;2352     2399  2         PROLOG;
;2353     2400  2
;2354     2401  2     ! First check if the user actually specified a prologue number                ;
;2355     2402  2     !
;2356     2403  2
;2357     2404  2     IF .XAB [XAB$B_BLN] GEQU XAB$C_KEYLEN                                        ;
;2358     2405  2     THEN
;2359     2406  2
;2360     2407  2         ! The user has a long key XAB
;2361     2408  2         !
;2362     2409  3         BEGIN
;2363     2410  3
;2364     2411  3         IF .XAB [XAB$B_PROLOG] GTRU 0
;2365     2412  3         THEN
;2366     2413  3
;2367     2414  3             ! We have an explicit prologue number,
;2368     2415  3             ! Check that there is a valid key XAB for that
```

```
;  2369     2416  3              ! prologue.
;  2370     2417  3              !
;  2371     2418  4              BEGIN
;  2372     2419  4
;  2373     2420  4              CASE .XAB [XAB$B_PROLOG] FROM 1 TO 3 OF
;  2374     2421  4                  SET
;  2375     2422  4
;  2376     2423  5                  [1,2] : BEGIN
;  2377     2424  5                          PROLOGUE = PLG$C_VER_NO;
;  2378     2425  5                          RETURN RMSSUC ();
;  2379     2426  4                          END;
;  2380     2427  4
;  2381     2428  5                  [3]   : BEGIN
;  2382     2429  5                          PROLOGUE = PLG$C_VER_3;
;  2383     2430  5                          RETURN CHECK_OVERLAP (.NO_SEGS);
;  2384     2431  4                          END;
;  2385     2432  4
;  2386     2433  4                  [OUTRANGE] : RETURN RMSERR (PLV);
;  2387     2434  4
;  2388     2435  4                  TES;
;  2389     2436  3              END;
;  2390     2437  3
;  2391     2438  2          END;
;  2392     2439  2
;  2393     2440  2      ! Either the user has a short XAB or he didn't specify
;  2394     2441  2      ! prologue number.  Look for process and system defaults.
;  2395     2442  2      !
;  2396     2443  2
;  2397     2444  2      IF .PIO$GB_RMSPROLOG<0,8> NEQU 0
;  2398     2445  2      THEN
;  2399     2446  2          ! There is a default process prologue level
;  2400     2447  2          !
;  2401     2448  2          PROLOG = .PIO$GB_RMSPROLOG<0,8>
;  2402     2449  2      ELSE
;  2403     2450  2
;  2404     2451  2      IF .SYS$GB_RMSPROLOG<0,8> NEQU 0
;  2405     2452  2      THEN
;  2406     2453  2          ! There is a default system prologue level
;  2407     2454  2          !
;  2408     2455  2          PROLOG = .SYS$GB_RMSPROLOG<0,8>
;  2409     2456  2      ELSE
;  2410     2457  2          ! Try for the best there is
;  2411     2458  2          !
;  2412     2459  2          PROLOG = PLG$C_VER_3;
;  2413     2460  2
;  2414     2461  2      RETURN CHECK_PROLOG (.PROLOG, .NO_SEGS; PROLOGUE);
;  2415     2462  2
;  2416     2463  1 END;


                  4C   8F        01    A6  91 00000 FIND_PROLOGUE:
                                                     CMPB    1(XAB), #76                    ; 2404
                                    23    1F 00005   BLSSU   4$                             ;
                                    48    A6  95 00007   TSTB    72(XAB)                    ; 2411
```

```
                            1E   13 0000A        BEQL    4$                          ;
         02       01     48 A6   8F 0000C        CASEB   72(XAB), #1, #2             ; 2428
       0013     000C       000C     00011 1$:    .WORD   2$-1$,-                     ;
                                                         2$-1$,-                     ;
                                                         3$-1$                       ;
                  50   872C 8F   3C 00017        MOVZWL  #34604, R0                  ; 2433
                                 05 0001C        RSB                                 ;
                  57        01   D0 0001D 2$:    MOVL    #1, PROLOGUE                ; 2424
                  50        01   D0 00020        MOVL    #1, R0                      ; 2428
                                 05 00023        RSB                                 ;
                  57        03   D0 00024 3$:    MOVL    #3, PROLOGUE                ; 2429
                          FEB6   31 00027        BRW     CHECK_OVERLAP               ; 2430
                  50 00000000G 00 9A 0002A 4$:   MOVZBL  PIO$GB_RMSPROLOG, R0        ; 2444
                            11   12 00031        BNEQ    6$                          ;
                  51 00000000G 00 9A 00033       MOVZBL  SYS$GB_RMSPROLOG, R1        ; 2451
                            05   13 0003A        BEQL    5$                          ;
                  50        51   D0 0003C        MOVL    R1, PROLOG                  ; 2455
                            03   11 0003F        BRB     6$                          ;
                  50        03   D0 00041 5$:    MOVL    #3, PROLOG                  ; 2459
                          FF13   31 00044 6$:    BRW     CHECK_PROLOG                ; 2461

; Routine Size:  71 bytes,    Routine Base:  RM$RMSMISC + 0BF5
```

```
;  2418          2464  1 %SBTTL  'GET_BKS'
;  2419          2465  1 ROUTINE GET_BKS ( AREA_NO ; BKT_SIZE) : RL$GET_BKS =
;  2420          2466  1
;  2421          2467  1 !++
;  2422          2468  1 !
;  2423          2469  1 ! GET_BKS
;  2424          2470  1 !       This routine returns the bucket size corresponding to the index
;  2425          2471  1 !       or data level area number of the given key of reference.
;  2426          2472  1 !
;  2427          2473  1 ! CALLING SEQUENCE:
;  2428          2474  1 !       GET_BKS ( AREA_NO ; BKT_SIZE )
;  2429          2475  1 !
;  2430          2476  1 ! INPUT PARAMETERS:
;  2431          2477  1 !       AREA_NO - Area number we are looking for
;  2432          2478  1 !
;  2433          2479  1 ! IMPLICIT INPUTS:
;  2434          2480  1 !       IFAB address
;  2435          2481  1 !       FAB address
;  2436          2482  1 !       XAB address
;  2437          2483  1 !
;  2438          2484  1 ! OUTPUT PARAMETERS:
;  2439          2485  1 !       BKT_SIZE = bucket size value for the given area
;  2440          2486  1 !
;  2441          2487  1 ! IMPLICIT OUTPUTS:
;  2442          2488  1 !       none
;  2443          2489  1 !
;  2444          2490  1 ! ROUTINE VALUE:
;  2445          2491  1 !       1 - success
;  2446          2492  1 !       XAB error on REPROBE
;  2447          2493  1 !
;  2448          2494  1 ! SIDE EFFECTS:
;  2449          2495  1 !       none
;  2450          2496  1 !
;  2451          2497  1 !--
;  2452          2498  1
;  2453          2499  2     BEGIN
;  2454          2500  2
;  2455          2501  2     EXTERNAL REGISTER
;  2456          2502  2         R_IFAB_STR,
;  2457          2503  2         R_FAB_STR,
;  2458          2504  2         XAB = 6 :       REF BBLOCK;
;  2459          2505  2
;  2460          2506  2     LOCAL
;  2461          2507  2         SAVE_XAB;                       ! stores address of current XAB
;  2462          2508  2
;  2463          2509  2     SAVE_XAB = .XAB;
;  2464          2510  2     XAB = .FAB [FAB$L_XAB];
;  2465          2511  2
;  2466          2512  2     ! Loop thru the XAB chain
;  2467          2513  2     !
;  2468          2514  2     WHILE .XAB NEQ 0
;  2469          2515  2     DO
;  2470          2516  3         BEGIN
;  2471          2517  3         RETURN_ON_ERROR ( REPROBE () );
;  2472          2518  3
;  2473          2519  3         ! If it is an area allocation XAB, and the area id is the one we
;  2474          2520  3         ! area looking for, get the bucket size and return.
```

```
;  2475          2521  3          !
;  2476          2522  3
;  2477          2523  3          IF .XAB [XAB$B_COD] EQL XAB$C_ALL
;  2478          2524  3            AND .XAB [XAB$B_AID] EQL .AREA_NO <0,8>
;  2479          2525  3          THEN
;  2480          2526  3              EXITLOOP;
;  2481          2527  3          XAB = .XAB [XAB$L_NXT];
;  2482          2528  2          END;
;  2483          2529  2
;  2484          2530  2      ! If no area was found, return IFAB bucket size value, else
;  2485          2531  2      ! use value from XAB.
;  2486          2532  2      !
;  2487          2533  2
;  2488          2534  2      IF .XAB EQL 0
;  2489          2535  2      THEN
;  2490          2536  2          BKT_SIZE = .IFAB [IFB$B_BKS]
;  2491          2537  2      ELSE
;  2492          2538  2          BKT_SIZE = .XAB [XAB$B_BKZ];
;  2493          2539  2      XAB = .SAVE_XAB;                        ! restore XAB addresss
;  2494          2540  2      RETURN RMS$OC ();
;  2495          2541  2
;  2496          2542  1      END;


                                    52  DD 00000 GET_BKS:PUSHL   R2                                  ; 2465
                              52    56  D0 00002          MOVL    XAB, SAVE_XAB                       ; 2509
                              56  24  AB  D0 00005          MOVL    36(FAB), XAB                        ; 2510
                                  18  13  00009 1$:        BEQL    3$                                  ; 2514
                                0000V 30  0000B            BSBW    REPROBE                             ; 2517
                              26    50  E9  0000E          BLBC    STATUS, 6$                          ;
                              14    66  91  00011          CMPB    (XAB), #20                          ; 2523
                                    07  12  00014          BNEQ    2$                                  ;
                        08    AE  17  A6  91  00016          CMPB    23(XAB), AREA_NO                    ; 2524
                                    06  13  0001B          BEQL    3$                                  ;
                              56  04  A6  D0  0001D 2$:      MOVL    4(XAB), XAB                         ; 2527
                                  E6  11  00021          BRB     1$                                  ; 2514
                                  56  D5  00023 3$:       TSTL    XAB                                 ; 2534
                                  06  12  00025          BNEQ    4$                                  ;
                              51  5E  AA  9A  00027          MOVZBL  94(IFAB), BKT_SIZE                  ; 2536
                                  04  11  0002B          BRB     5$                                  ;
                              51  16  A6  9A  0002D 4$:      MOVZBL  22(XAB), BKT_SIZE                   ; 2538
                              56    52  D0  00031 5$:       MOVL    SAVE_XAB, XAB                       ; 2539
                              50    01  D0  00034          MOVL    #1, R0                              ; 2540
                                  04  BA  00037 6$:       POPR    #^M<R2>                             ; 2542
                                      05  00039          RSB                                         ;

; Routine Size:  58 bytes,    Routine Base:  RM$RMSMISC + 0C3C
```

```
;  2498          2543  1 %SBTTL 'GET_KEY_PARMS'
;  2499          2544  1 ROUTINE GET_KEY_PARMS ( ;KEYSIZE, MINREC, SEGS ) : RL$GET_KEY_PARMS NOVALUE =
;  2500          2545  1
;  2501          2546  1 !++
;  2502          2547  1 !
;  2503          2548  1 ! GET_KEY_PARMS
;  2504          2549  1 !       This routine determines the number of segments in a key,
;  2505          2550  1 !       the size of the key, and the minimum record size containing key.
;  2506          2551  1 !
;  2507          2552  1 ! CALLING SEQUENCE:
;  2508          2553  1 !       GET_KEY_PARMS ( ;KEYSIZE, MINREC, SEGS )
;  2509          2554  1 !
;  2510          2555  1 ! INPUT PARAMETERS·
;  2511          2556  1 !       NONE
;  2512          2557  1 !
;  2513          2558  1 ! IMPLICIT INPUTS:
;  2514          2559  1 !       XAB     - address of XAB to process
;  2515          2560  1 !
;  2516          2561  1 ! OUTPUT PARAMETERS:
;  2517          2562  1 !       KEYSIZE - total number of characters in key
;  2518          2563  1 !       MINREC  - minimum size of record necessary to contain key
;  2519          2564  1 !       SEGS - number of segments in key
;  2520          2565  1 !
;  2521          2566  1 ! IMPLICIT OUTPUTS:
;  2522          2567  1 !       None
;  2523          2568  1 !
;  2524          2569  1 ! ROUTINE VALUE:
;  2525          2570  1 !       none
;  2526          2571  1 !
;  2527          2572  1 ! SIDE EFFECTS:
;  2528          2573  1 !       none
;  2529          2574  1 !
;  2530          2575  1 !--
;  2531          2576  1
;  2532          2577  2     BEGIN
;  2533          2578  2
;  2534          2579  2     EXTERNAL REGISTER
;  2535          2580  2         XAB = 6 : REF BBLOCK;
;  2536          2581  2
;  2537          2582  2     LOCAL
;  2538          2583  2         LEN     : REF VECTOR [, BYTE],
;  2539          2584  2         POS     : REF VECTOR [, WORD],
;  2540          2585  2         TYP     : REF VECTOR [, BYTE];
;  2541          2586  2
;  2542          2587  2     POS = XAB [XAB$W_POS];
;  2543          2588  2     LEN = XAB [XAB$B_SIZ];
;  2544          2589  2     SEGS = 0;
;  2545          2590  2     MINREC = 0;
;  2546          2591  2     KEYSIZE = 0;
;  2547          2592  2
;  2548          2593  2     ! Loop thru all key segments adding values to find keysize, number of
;  2549          2594  2     ! segments, and minimum record size to include key.  Stop the loop when
;  2550          2595  2     ! a segment of zero length is found.
;  2551          2596  2     !
;  2552          2597  2
;  2553          2598  2     INCR I FROM 0 TO 7 DO
;  2554          2599  3         BEGIN
```

```
;  2555          2600  3
;  2556          2601  3          IF .LEN[.I] NEQ 0
;  2557          2602  3          THEN
;  2558          2603  4              BEGIN
;  2559          2604  4              SEGS = .SEGS + 1;
;  2560          2605  4              KEYSIZE = .KEYSIZE + .LEN[.I];
;  2561          2606  4
;  2562          2607  4              IF .POS[.I] + .LEN[.I] GTRU .MINREC
;  2563          2608  4              THEN
;  2564          2609  4                  MINREC = .POS[.I] + .LEN[.I];
;  2565          2610  4              END
;  2566          2611  3          ELSE
;  2567          2612  3              EXITLOOP;
;  2568          2613  2          END;
;  2569          2614  2
;  2570          2615  2      ! If the first segment had zero length, assume there is at least one
;  2571          2616  2      ! segment, until we can verify if the key definition is correct.
;  2572          2617  2      !
;  2573          2618  2
;  2574          2619  2      IF .SEGS EQL 0
;  2575          2620  2      THEN
;  2576          2621  2          SEGS = 1;
;  2577          2622  2
;  2578          2623  1      END;


                                    01B0    8F  BB 00000 GET_KEY_PARMS:
                                                PUSHR   #^M<R4,R5,R7,R8>                ;  2544
                                50        1E    A6  9E 00004           MOVAB   30(R6), POS             ;  2587
                                57        2E    A6  9E 00008           MOVAB   46(R6), LEN             ;  2588
                                              52  7C 0000C             CLRQ    MINREC                  ;  2590
                                              51  D4 0000E             CLRL    KEYSIZE                 ;  2591
                                              54  D4 00010             CLRL    I                       ;  2601
                                55          6447  9A 00012 1$:       MOVZBL  (I)[LEN], R5            ;
                                              18  13 00016             BEQL    3$                      ;
                                              53  D6 00018             INCL    SEGS                    ;  2604
                                51            55  C0 0001A           ADDL2   R5, KEYSIZE             ;  2605
                                58          6044  3C 0001D           MOVZWL  (POS)[I], R8            ;  2607
                                55            58  C0 00021           ADDL2   R8, R5                  ;
                                52            55  D1 00024           CMPL    R5, MINREC              ;
                                              03  1B 00027             BLEQU   2$                      ;
                                52            55  D0 00029           MOVL    R5, MINREC              ;  2609
                            E2  54            07  F3 0002C 2$:       AOBLEQ  #7, I, 1$               ;  2598
                                              53  D5 00030 3$:       TSTL    SEGS                    ;  2619
                                              03  12 00032             BNEQ    4$                      ;
                                53            01  D0 00034           MOVL    #1, SEGS                ;  2621
                                        01B0    8F  BA 00037 4$:       POPR    #^M<R4,R5,R7,R8>        ;  2623
                                                05 0003B               RSB                             ;

; Routine Size:  60 bytes,    Routine Base:  RM$RMSMISC + 0C76

;  2579          2624  1
```

```
;  2581          2625  1 %SBTTL  'MOVE'
;  2582          2626  1 ROUTINE MOVE ( SEGMENTS ) : RL$WRITE_KEY_DESC =
;  2583          2627  1
;  2584          2628  1 !++
;  2585          2629  1 !
;  2586          2630  1 ! MOVE
;  2587          2631  1 !
;  2588          2632  1 !       This routine moves key position/length information
;  2589          2633  1 !       from XAB's to key descriptors on disk.
;  2590          2634  1 !       If also moves the key name if non_zero.
;  2591          2635  1 !       It is done as a routine so as not to clobber R4 - R5.
;  2592          2636  1 !
;  2593          2637  1 ! CALLING SEQUENCE:
;  2594          2638  1 !       MOVE ( SEGMENTS )
;  2595          2639  1 !
;  2596          2640  1 ! INPUT PARAMETERS:
;  2597          2641  1 !       SEGMENTS - number of segments in the key
;  2598          2642  1 !
;  2599          2643  1 ! IMPLICIT INPUTS:
;  2600          2644  1 !       BUF     - address of key descriptor buffer
;  2601          2645  1 !       XAB     - address of XAB containing data to be moved
;  2602          2646  1 !
;  2603          2647  1 ! OUTPUT PARAMETERS:
;  2604          2648  1 !       none
;  2605          2649  1 !
;  2606          2650  1 ! IMPLICIT OUTPUTS:
;  2607          2651  1 !       data moved to BUF, an image of the disk key descriptor
;  2608          2652  1 !
;  2609          2653  1 ! ROUTINE VALUE:
;  2610          2654  1 !       KNM error if keyname reprobe fails
;  2611          2655  1 !       SUC - success
;  2612          2656  1 !
;  2613          2657  1 ! SIDE EFFECTS:
;  2614          2658  1 !       none
;  2615          2659  1 !
;  2616          2660  1 !--
;  2617          2661  1
;  2618          2662  2     BEGIN
;  2619          2663  2
;  2620          2664  2     EXTERNAL REGISTER
;  2621          2665  2         R_FAB_STR,
;  2622          2666  2         R_IFAB_STR,
;  2623          2667  2         BUF = 7 : REF BBLOCK,
;  2624          2668  2         XAB = 6 : REF BBLOCK;
;  2625          2669  2
;  2626          2670  2     ASSUME( %QUOTE KEY$W_POSITION,  %QUOTE KEY$W_POSITION1);
;  2627          2671  2     ASSUME( %QUOTE KEY$W_POSITION1, %QUOTE KEY$W_POSITION2);
;  2628          2672  2     ASSUME( %QUOTE KEY$W_POSITION2, %QUOTE KEY$W_POSITION3);
;  2629          2673  2     ASSUME( %QUOTE KEY$W_POSITION3, %QUOTE KEY$W_POSITION4);
;  2630          2674  2     ASSUME( %QUOTE KEY$W_POSITION4, %QUOTE KEY$W_POSITION5);
;  2631          2675  2     ASSUME( %QUOTE KEY$W_POSITION5, %QUOTE KEY$W_POSITION6);
;  2632          2676  2     ASSUME( %QUOTE KEY$W_POSITION6, %QUOTE KEY$W_POSITION7);
;  2633          2677  2     ASSUME( %QUOTE KEY$W_POSITION7, %QUOTE KEY$B_SIZE);
;  2634          2678  2     ASSUME( %QUOTE KEY$B_SIZE,  %QUOTE KEY$B_SIZE1);
;  2635          2679  2     ASSUME( %QUOTE KEY$B_SIZE1, %QUOTE KEY$B_SIZE2);
;  2636          2680  2     ASSUME( %QUOTE KEY$B_SIZE2, %QUOTE KEY$B_SIZE3);
;  2637          2681  2     ASSUME( %QUOTE KEY$B_SIZE3, %QUOTE KEY$B_SIZE4);
```

```
;  2638          2682  2      ASSUME( %QUOTE KEY$B_SIZE4, %QUOTE KEY$B_SIZE5);
;  2639          2683  2      ASSUME( %QUOTE KEY$B_SIZE5, %QUOTE KEY$B_SIZE6);
;  2640          2684  2      ASSUME( %QUOTE KEY$B_SIZE6, %QUOTE KEY$B_SIZE7);
;  2641          2685  2      ASSUME( %QUOTE KEY$B_SIZE7, %QUOTE KEY$T_KEYNAM);
;  2642          2686  2
;  2643        L 2687  2  %IF $BYTEOFFSET(KEY$T_KEYNAM) + KEY$S_KEYNAM NEQ $BYTEOFFSET(KEY$L_LDVBN)
;  2644        U 2688  2  %THEN
;  2645        U 2689  2      %WARN('WARNING CONSTANT HAS CHANGES')
;  2646          2690  2  %FI;
;  2647          2691  2
;  2648          2692  2      ! Pick up the keyname buffer address into a local.  Reprobe for read,
;  2649          2693  2      ! and if valid, move to key descriptor buffer.
;  2650          2694  2      !
;  2651          2695  2
;  2652          2696  2      IF .XAB [XAB$L_KNM] NEQ 0
;  2653          2697  2      THEN
;  2654          2698  3          BEGIN
;  2655          2699  3
;  2656          2700  3          LOCAL
;  2657          2701  3              KEYNAM;
;  2658          2702  3
;  2659          2703  3          KEYNAM = .XAB [XAB$L_KNM];
;  2660          2704  3
;  2661        P 2705  3          IFNORD ( %REF (KEY$S_KEYNAM), .KEYNAM, IFAB [IFB$B_MODE],
;  2662        P 2706  3                   BEGIN
;  2663        P 2707  3                   FAB [FAB$L_STV] = .XAB [XAB$B_REF];
;  2664        P 2708  3                   RETURN RMSERR (KNM);
;  2665          2709  3                   END );
;  2666          2710  3          CH$MOVE(KEY$S_KEYNAM, .KEYNAM, BUF[KEY$T_KEYNAM]);
;  2667          2711  2          END;
;  2668          2712  2
;  2669          2713  2      ! Move the segments position and size.  Use the number of
;  2670          2714  2      ! segments to do the move as a unit.
;  2671          2715  2      !
;  2672          2716  2      CH$MOVE ( .SEGMENTS * 2, XAB [XAB$W_POS], BUF [KEY$W_POSITION] );
;  2673          2717  2      CH$MOVE ( .SEGMENTS, XAB [XAB$B_SIZ], BUF [KEY$B_SIZE] );
;  2674          2718  2
;  2675          2719  2      ! Set data types values.  Propagate XAB$B_DTP to TYP0, and
;  2676          2720  2      ! TYP1-TYP7 to null (STG).
;  2677          2721  2      !
;  2678          2722  2      BUF[KEY$B_TYPE] = .XAB [XAB$B_DTP];
;  2679          2723  2      CH$FILL( 0, 7, BUF[KEY$B_TYPE1] );
;  2680          2724  2
;  2681          2725  2      RETURN RMSSUC();
;  2682          2726  1      END;                                    ! end of routine MOVE


                                        3C  BB 00000 MOVE:   PUSHR   #^M<R2,R3,R4,R5>                ; 2626
                                    38  A6  D5 00002         TSTL    56(XAB)                         ; 2696
                                        1C  13 00005         BEQL    2$                              ;
                                50      38  A6  D0 00007         MOVL    56(XAB), KEYNAM             ; 2703
                            60      20      0A  AA  0C 0000B     PROBER  10(IFAB), #32, (KEYNAM)     ; 2709
                                        0C  12 00010         BNEQ    1$                              ;
                                0C  A8      17  A6  9A 00012     MOVZBL  23(XAB), 12(FAB)            ;
```

```
                                50      8774    8F  3C 00017            MOVZWL  #34676, R0                                ;
                                                26  11 0001C            BRB     3$                                       ;
                    34  A7              60          20  28 0001E 1$:    MOVC3   #32, (KEYNAM), 52(BUF)                   ; 2710
                        50      14      AE          01  78 00023 2$:    ASHL    #1, SEGMENTS, R0                         ; 2716
                    1C  A7      1E      A6          50  28 00028        MOVC3   R0, 30(XAB), 28(BUF)                     ;
                    2C  A7      2E      A6      14  AE  28 0002E        MOVC3   SEGMENTS, 46(XAB), 44(BUF)               ; 2717
                                58      A7      13  A6  90 00035        MOVB    19(XAB), 88(BUF)                         ; 2722
            07              00              6E          00  2C 0003A    MOVC5   #0, (SP), #0, #7, 89(BUF)                ; 2723
                                                59  A7      0003F       
                                50              01  D0 00041        MOVL    #1, R0                                   ; 2725
                                                    3C  BA 00044 3$:    POPR    #^M<R2,R3,R4,R5>                         ; 2726
                                                        05 00046        RSB                                              ;

; Routine Size:  71 bytes,    Routine Base:  RM$RMSMISC + 0CB2
```

```
;2684          2727  1 %SBTTL  'READ_PROLOGUE'
;2685          2728  1 ROUTINE READ_PROLOGUE ( VBN ) : RL$RW_PROLOG =
;2686          2729  1
;2687          2730  1 !++
;2688          2731  1 !
;2689          2732  1 ! READ_PROLOGUE
;2690          2733  1 !
;2691          2734  1 !       This routine locks the VBN.
;2692          2735  1 !
;2693          2736  1 ! CALLING SEQUENCE:
;2694          2737  1 !       READ_PROLOGUE ( VBN )
;2695          2738  1 !
;2696          2739  1 ! INPUT PARAMETERS:
;2697          2740  1 !       VBN     - VBN to lock
;2698          2741  1 !
;2699          2742  1 ! IMPLICIT INPUTS:
;2700          2743  1 !       none
;2701          2744  1 !
;2702          2745  1 ! OUTPUT PARAMETERS:
;2703          2746  1 !       none
;2704          2747  1 !
;2705          2748  1 ! IMPLICIT OUTPUTS:
;2706          2749  1 !       none
;2707          2750  1 !
;2708          2751  1 ! ROUTINE VALUE:
;2709          2752  1 !       RPL error
;2710          2753  1 !       SUC - success
;2711          2754  1 !
;2712          2755  1 ! SIDE EFFECTS:
;2713          2756  1 !       none
;2714          2757  1 !
;2715          2758  1 !--
;2716          2759  1
;2717          2760  2     BEGIN
;2718          2761  2
;2719          2762  2     EXTERNAL REGISTER
;2720          2763  2         COMMON_FAB_STR,
;2721          2764  2         R_BDB_STR;
;2722          2765  2
;2723          2766  2     GLOBAL REGISTER
;2724          2767  2         R_BKT_ADDR;
;2725          2768  2
;2726        P 2769  2     RETURN_ON_ERROR ( RMSCACHE (.VBN, 512, CSH$M_NOREAD  OR  CSH$M_LOCK),
;2727        P 2770  2                       BEGIN
;2728        P 2771  2                       IF .FAB [FAB$L_STV] EQL 0
;2729        P 2772  2                       THEN
;2730        P 2773  2                           FAB [FAB$L_STV] = .STATUS OR 1^16;
;2731        P 2774  2                       STATUS = RMSERR (RPL)
;2732          2775  2                       END );
;2733          2776  2
;2734          2777  2     RETURN RMSSUC();
;2735          2778  2
;2736          2779  1     END;
```

```
                                 2C  BB 00000 READ_PROLOGUE:
                                                 PUSHR   #^M<R2,R3,R5>                           ; 2728
                           53    05  D0 00002             MOVL    #5, R3                          ; 2775
                           52  0200  8F  3C 00005         MOVZWL  #512, R2                        ;
                           51    10  AE  D0 0000A         MOVL    VBN, R1                         ;
                     00000000G   00  16 0000E             JSB     RM$CACHE                        ;
                           15        50  E8 00014         BLBS    STATUS, 2$                      ;
                             0C      A8  D5 00017         TSTL    12(FAB)                         ;
                                     09  12 0001A         BNEQ    1$                              ;
                0C   A8    50 00010000  8F  C9 0001C     BISL3   #65536, STATUS, 12(FAB)         ;
                           50  C104  8F  3C 00025 1$:     MOVZWL  #49412, STATUS                  ;
                                     03  11 0002A         BRB     3$                              ;
                           50        01  D0 0002C 2$:     MOVL    #1, R0                          ; 2777
                                     2C  BA 0002F 3$:     POPR    #^M<R2,R3,R5>                   ; 2779
                                         05 00031         RSB                                     ;

; Routine Size:  50 bytes,    Routine Base:  RM$RMSMISC + 0CF9
```

```
;2738        2780  1 %SBTTL  'REPROBE'
;2739        2781  1 ROUTINE REPROBE : RL$CHK_AND_REPROBE =
;2740        2782  1
;2741        2783  1 !++
;2742        2784  1 !
;2743        2785  1 ! REPROBE
;2744        2786  1 !       This routine reprobes current XAB for read and write.
;2745        2787  1 !
;2746        2788  1 ! CALLING SEQUENCE:
;2747        2789  1 !       REPROBE()
;2748        2790  1 !
;2749        2791  1 ! INPUT PARAMETERS:
;2750        2792  1 !       none
;2751        2793  1 !
;2752        2794  1 ! IMPLICIT INPUTS:
;2753        2795  1 !       XAB - address of user XAB
;2754        2796  1 !       IFAB address
;2755        2797  1 !       FAB address
;2756        2798  1 !
;2757        2799  1 ! OUTPUT PARAMETERS:
;2758        2800  1 !       none
;2759        2801  1 !
;2760        2802  1 ! IMPLICIT OUTPUTS:
;2761        2803  1 !       none
;2762        2804  1 !
;2763        2805  1 ! ROUTINE VALUE:
;2764        2806  1 !       Success or XAB error code
;2765        2807  1 !
;2766        2808  1 ! SIDE EFFECTS:
;2767        2809  1 !       none
;2768        2810  1 !
;2769        2811  1 !--
;2770        2812  1
;2771        2813  2     BEGIN
;2772        2814  2
;2773        2815  2     EXTERNAL REGISTER
;2774        2816  2         R_IFAB_STR,
;2775        2817  2         R_FAB_STR,
;2776        2818  2         XAB = 6 :       REF BBLOCK;
;2777        2819  2
;2778        2820  2     ! Probe current XAB for read and write.
;2779        2821  2     !
;2780        2822  2
;2781      P 2823  2     IFNORD ( %REF($BYTEOFFSET(XAB$L_NXT) + $FIELDWIDTH(XAB$L_NXT)/8),
;2782      P 2824  2                .XAB,
;2783      P 2825  2                IFAB [IFB$B_MODE],
;2784      P 2826  2               (FAB [FAB$L_STV] = .XAB;
;2785        2827  2               RETURN RMSERR (XAB)) );
;2786        2828  2
;2787        2829  2     IF .XAB [XAB$B_COD] EQL XAB$C_ALL
;2788        2830  2     THEN
;2789      P 2831  2         IFNOWRT ( %REF(XAB$C_ALLLEN), .XAB, IFAB [IFB$B_MODE],
;2790      P 2832  2                   (FAB [FAB$L_STV] = .XAB;
;2791        2833  2                   RETURN RMSERR (XAB)) );
;2792        2834  2
;2793        2835  2     IF .XAB [XAB$B_COD] EQL XAB$C_KEY
;2794        2836  2     THEN
```

```
;  2795          2837  3          BEGIN
;  2796          2838  3
;  2797          2839  3          IF .XAB [XAB$B_BLN] GEQ XAB$C_KEYLEN
;  2798          2840  3          THEN
;  2799        P 2841  3              IFNOWRT ( %REF(XAB$C_KEYLEN), .XAB, IFAB [IFB$B_MODE],
;  2800        P 2842  3                        (FAB [FAB$L_STV] = .XAB;
;  2801          2843  4                        RETURN RMSERR (XAB)) )
;  2802          2844  3          ELSE
;  2803        P 2845  3              IFNOWRT ( %REF(XAB$C_KEYLEN_V2), .XAB, IFAB [IFB$B_MODE],
;  2804        P 2846  3                        (FAB [FAB$L_STV] = .XAB;
;  2805          2847  3                        RETURN RMSERR (XAB)) );
;  2806          2848  3
;  2807          2849  2          END;
;  2808          2850  2
;  2809          2851  2      RETURN RMSSUC();
;  2810          2852  2
;  2811          2853  1      END;


                     66            08        0A    AA  0C 00000 REPROBE:PROBER  10(IFAB), #8, (XAB)                ;  2827
                                                   2A  13 00005              BEQL    4$
                                   14             66  91 00007              CMPB    (XAB), #20                      ;  2829
                                                  07  12 0000A              BNEQ    1$
                     66            20        0A    AA  0D 0000C              PROBEW  10(IFAB), #32, (XAB)            ;  2833
                                                   1E  13 00011              BEQL    4$
                                   15             66  91 00013 1$:          CMPB    (XAB), #21                      ;  2835
                                                  23  12 00016              BNEQ    5$
                           4C      8F        01    A6  91 00018              CMPB    1(XAB), #76                     ;  2839
                                                  09  1F 0001D              BLSSU   2$
                     66    004C    8F        0A    AA  0D 0001F              PROBEW  10(IFAB), #76, (XAB)            ;  2843
                                                  07  11 00026              BRB     3$
                     66    0040    8F        0A    AA  0D 00028 2$:          PROBEW  10(IFAB), #64, (XAB)            ;  2847
                                                  0A  12 0002F 3$:          BNEQ    5$
                           0C      A8                56  D0 00031 4$:          MOVL    XAB, 12(FAB)
                                   50      870C      8F  3C 00035              MOVZWL  #34572, R0
                                                         05 0003A              RSB
                                   50                01  D0 0003B 5$:          MOVL    #1, R0                          ;  2851
                                                         05 0003E              RSB                                     ;  2853

; Routine Size:  63 bytes,    Routine Base:  RM$RMSMISC + 0D2B

;  2812          2854  1
```

```
;2814     2855  1 %SBTTL  'WRITE_KEY_DESC'
;2815     2856  1 ROUTINE WRITE_KEY_DESC : RL$WRITE_KEY_DESC =
;2816     2857  1
;2817     2858  1 !++
;2818     2859  1 !
;2819     2860  1 ! WRITE_KEY_DESC
;2820     2861  1 !       This routine fills in the data of the prologue key descriptor.
;2821     2862  1 !
;2822     2863  1 ! CALLING SEQUENCE:
;2823     2864  1 !       WRITE_KEY_DESC
;2824     2865  1 !
;2825     2866  1 ! INPUT PARAMETERS:
;2826     2867  1 !       None
;2827     2868  1 !
;2828     2869  1 ! IMPLICIT INPUTS:
;2829     2870  1 !       IFAB address
;2830     2871  1 !       FAB address
;2831     2872  1 !       XAB to process
;2832     2873  1 !       BUF - address of key descriptor
;2833     2874  1 !
;2834     2875  1 ! IMPLICIT OUTPUTS:
;2835     2876  1 !       None
;2836     2877  1 !
;2837     2878  1 ! ROUTINE VALUE:
;2838     2879  1 !       SUC - success
;2839     2880  1 !       KNM - error from routine MOVE when keyname probing fails
;2840     2881  1 !
;2841     2882  1 ! SIDE EFFECTS:
;2842     2883  1 !       Some fields in user's XAB are written.
;2843     2884  1 !
;2844     2885  1 !--
;2845     2886  1
;2846     2887  2     BEGIN
;2847     2888  2
;2848     2889  2     EXTERNAL REGISTER
;2849     2890  2         R_FAB_STR,
;2850     2891  2         R_IFAB_STR,
;2851     2892  2         BUF = 7 :       REF BBLOCK,
;2852     2893  2         XAB = 6 :       REF BBLOCK;
;2853     2894  2
;2854     2895  2     CLEAR ( .BUF, KEY$C_BLN + KEY$C_SPARE );
;2855     2896  2
;2856     2897  2     ! Fill in key descriptor in prologue VBN
;2857     2898  2     ! key descriptor.
;2858     2899  2     !
;2859     2900  2     BUF [KEY$B_IANUM] = .XAB [XAB$B_IAN];
;2860     2901  2     BUF [KEY$B_LANUM] = .XAB [XAB$B_LAN];
;2861     2902  2     BUF [KEY$B_DANUM] = .XAB [XAB$B_DAN];
;2862     2903  2
;2863     2904  2     ! This is needed to conform with RMS_11 clug.
;2864     2905  2     !
;2865     2906  2     (BUF [KEY$L_ROOTVBN])<0, 8> = .XAB [XAB$B_DAN];
;2866     2907  2     (BUF [KEY$L_ROOTVBN])<8, 8> = .XAB [XAB$B_IAN];
;2867     2908  2     (BUF [KEY$L_ROOTVBN])<16, 8> = .XAB [XAB$B_LAN];
;2868     2909  2     BUF [KEY$B_DATATYPE] = .XAB [XAB$B_DTP];
;2869     2910  2
;2870     2911  2     ! Get the bucket size calculated for the given area number
```

```
;  2871          2912  2      !
;  2872          2913  2
;  2873          2914  3      BEGIN
;  2874          2915  3
;  2875          2916  3      LOCAL
;  2876          2917  3          BKT_SIZE;
;  2877          2918  3
;  2878          2919  3      RETURN_ON_ERROR ( GET_BKS ( .XAB [XAB$B_IAN]; BKT_SIZE ) );
;  2879          2920  3      BUF [KEY$B_IDXBKTSZ] = .BKT_SIZE;
;  2880          2921  3      RETURN_ON_ERROR ( GET_BKS ( .XAB [XAB$B_DAN]; BKT_SIZE ) );
;  2881          2922  3      BUF [KEY$B_DATBKTSZ] = .BKT_SIZE;
;  2882          2923  2      END;                                    ! end of BKT_SIZE definition
;  2883          2924  2
;  2884          2925  3      BEGIN
;  2885          2926  3
;  2886          2927  3      LOCAL
;  2887          2928  3          KEYSIZE,
;  2888          2929  3          MINREC,
;  2889          2930  3          SEGS;
;  2890          2931  3
;  2891          2932  3      GET_KEY_PARMS ( ;KEYSIZE, MINREC, SEGS );
;  2892          2933  3
;  2893          2934  3      BUF [KEY$B_SEGMENTS] = .SEGS;
;  2894          2935  3      BUF [KEY$B_NULLCHAR] = .XAB [XAB$B_NUL];
;  2895          2936  3      BUF [KEY$B_KEYSZ] = .KEYSIZE;
;  2896          2937  3      BUF [KEY$B_KEYREF] = .XAB [XAB$B_REF];
;  2897          2938  3      BUF [KEY$W_MINRECSZ] = .MINREC;
;  2898          2939  3
;  2899          2940  3      ! Now we must make intelligent decisions about the compression
;  2900          2941  3      ! bits for the keys.  If it is not a plg 3 file, then we want
;  2901          2942  3      ! to turn off all compression, since default is zero.
;  2902          2943  3      !
;  2903          2944  3
;  2904          2945  3      IF .IFAB [IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  2905          2946  3      THEN
;  2906          2947  4          BEGIN
;  2907          2948  4          XAB [XAB$V_KEY_NCMPR] = 1;
;  2908          2949  4          XAB [XAB$V_IDX_NCMPR] = 1;
;  2909          2950  4
;  2910          2951  4          IF .XAB [XAB$B_REF] EQL 0
;  2911          2952  4          THEN
;  2912          2953  4              XAB [XAB$V_DAT_NCMPR] = 1;
;  2913          2954  4          END
;  2914          2955  4
;  2915          2956  4      ! In a prologue 3 file, we decide if the compression should
;  2916          2957  4      ! be turned off depending on the key size and type.
;  2917          2958  4      !
;  2918          2959  3      ELSE
;  2919          2960  4          BEGIN
;  2920          2961  4
;  2921          2962  4          ! If the key is less than 6 bytes, or it is not a string key,
;  2922          2963  4          ! do not allow compression.
;  2923          2964  4          !
;  2924          2965  4
;  2925          2966  4          IF .KEYSIZE LSS 6
;  2926          2967  4            OR .XAB [XAB$B_DTP] NEQ XAB$C_STG
;  2927          2968  4          THEN
```

```
;  2928          2969  5              BEGIN
;  2929          2970  5              XAB [XAB$V_KEY_NCMPR] = 1;
;  2930          2971  5              XAB [XAB$V_IDX_NCMPR] = 1;
;  2931          2972  4              END;
;  2932          2973  4
;  2933          2974  4          ! If the data record left after the primary key is
;  2934          2975  4          ! extracted is less than 11 bytes, do not allow data compression.
;  2935          2976  4          !
;  2936          2977  4
;  2937          2978  4          IF .XAB [XAB$B_REF] EQL 0
;  2938          2979  4          THEN
;  2939          2980  4
;  2940          2981  4              IF .IFAB [IFB$W_MRS] NEQ 0
;  2941          2982  4                AND (.IFAB [IFB$W_MRS] - .KEYSIZE<0,16>) LSS 11
;  2942          2983  4              THEN
;  2943          2984  4                  XAB [XAB$V_DAT_NCMPR] = 1;
;  2944          2985  3          END;
;  2945          2986  3
;  2946          2987  3      ! Store flag values.  First those that are valid for all keys, and
;  2947          2988  3      ! then depending on the key of reference.
;  2948          2989  3      !
;  2949          2990  3      BUF [KEY$V_DUPKEYS] = .XAB [XAB$V_DUP];
;  2950          2991  3      BUF [KEY$V_INITIDX] = 1;
;  2951          2992  3      BUF [KEY$V_IDX_COMPR] = NOT .XAB [XAB$V_IDX_NCMPR];
;  2952          2993  3      BUF [KEY$V_KEY_COMPR] = NOT .XAB [XAB$V_KEY_NCMPR];
;  2953          2994  3
;  2954          2995  3      IF .XAB [XAB$B_REF] EQL 0
;  2955          2996  3      THEN
;  2956          2997  3          BUF [KEY$V_REC_COMPR] = NOT .XAB [XAB$V_DAT_NCMPR]
;  2957          2998  3      ELSE
;  2958          2999  4          BEGIN
;  2959          3000  4          BUF [KEY$V_NULKEYS] = .XAB [XAB$V_NUL];
;  2960          3001  4          BUF [KEY$V_CHGKEYS] = .XAB [XAB$V_CHG];
;  2961          3002  3          END;
;  2962          3003  3
;  2963          3004  3      ! Move key position / length information
;  2964          3005  3      !
;  2965          3006  3      RETURN_ON_ERROR ( MOVE ( .SEGS ) );
;  2966          3007  3
;  2967          3008  2      END;                          ! end of KEYSIZE,MINREC,SEGS def
;  2968          3009  2
;  2969          3010  2      BUF [KEY$W_IDXFILL] = .XAB [XAB$W_IFL];
;  2970          3011  2
;  2971          3012  2      IF .BUF [KEY$W_IDXFILL] EQL 0
;  2972          3013  2      THEN
;  2973          3014  2          BUF [KEY$W_IDXFILL] = .BUF [KEY$B_IDXBKTSZ] * 512;
;  2974          3015  2
;  2975          3016  2      IF .BUF [KEY$W_IDXFILL] LSSU
;  2976          3017  3                           WORDMASK( .BUF [KEY$B_IDXBKTSZ] * ( 512/2 ) )
;  2977          3018  2      THEN
;  2978          3019  2          BUF [KEY$W_IDXFILL] = .BUF [KEY$B_IDXBKTSZ] * ( 512/2 );
;  2979          3020  2
;  2980          3021  2      BUF [KEY$W_DATFILL] = .XAB [XAB$W_DFL];
;  2981          3022  2
;  2982          3023  2      IF .BUF [KEY$W_DATFILL] EQL 0
;  2983          3024  2      THEN
;  2984          3025  2          BUF [KEY$W_DATFILL] = .BUF [KEY$B_DATBKTSZ] * 512;
```

```
;  2985          3026  2
;  2986          3027  2      IF .BUF [KEY$W_DATFILL] LSSU
;  2987          3028  3                 WORDMASK( .BUF [KEY$B_DATBKTSZ] * ( 512/2 ) )
;  2988          3029  2      THEN
;  2989          3030  2          BUF [KEY$W_DATFILL] = .BUF [KEY$B_DATBKTSZ] * ( 512/2 );
;  2990          3031  2
;  2991          3032  2      ! Set pointers to next key descriptor:  if this is the last key,
;  2992          3033  2      ! then pointers should be zero.  Otherwise, use key of reference
;  2993          3034  2      ! value to find next position.
;  2994          3035  2      !
;  2995          3036  2
;  2996          3037  3      IF .XAB [XAB$B_REF] EQL (.IFAB [IFB$B_NUM_KEYS] - 1)
;  2997          3038  2      THEN
;  2998          3039  3          BEGIN
;  2999          3040  3          BUF [KEY$L_IDXFL] = 0;
;  3000          3041  3          BUF [KEY$W_NOFF] = 0;
;  3001          3042  3          END
;  3002          3043  2      ELSE
;  3003          3044  3          BEGIN
;  3004          3045  3          BUF [KEY$L_IDXFL] = (.XAB [XAB$B_REF] + 5) / 5 + 1;
;  3005          3046  4          BUF [KEY$W_NOFF] = ( ( .XAB [XAB$B_REF] ) MOD 5 )
;  3006          3047  3                                     * ( KEY$C_BLN + KEY$C_SPARE );
;  3007          3048  2          END;
;  3008          3049  2
;  3009          3050  2      RETURN RMSSUC();                    ! return success
;  3011          3052  1      END;

                        0C  BB 00000 WRITE_KEY_DESC:
                                     PUSHR   #^M<R2,R3>                          ;  2856
           7E       66   8F  9A 00002         MOVZBL  #102, -(SP)                 ;  2895
                         57  DD 00006         PUSHL   BUF
                      FE27  30 00008         BSBW    CLEAR
           5E            04  C0 0000B         ADDL2   #4, SP
06         A7       08   A6  B0 0000E         MOVW    8(XAB), 6(BUF)              ;  2900
08         A7       0A   A6  90 00013         MOVB    10(XAB), 8(BUF)             ;  2902
0C         A7       0A   A6  90 00018         MOVB    10(XAB), 12(BUF)            ;  2906
0D         A7       08   A6  B0 0001D         MOVW    8(XAB), 13(BUF)             ;  2907
11         A7       13   A6  90 00022         MOVB    19(XAB), 17(BUF)            ;  2909
           6E       08   A6  9A 00027         MOVZBL  8(XAB), (SP)                ;  2919
                      FEA4  30 0002B         BSBW    GET_BKS
           5E            04  C0 0002E         ADDL2   #4, SP
           0E            50  E9 00031         BLBC    STATUS, 1$
0A         A7            51  90 00034         MOVB    BKT_SIZE, 10(BUF)           ;  2920
           7E       0A   A6  9A 00038         MOVZBL  10(XAB), -(SP)              ;  2921
                      FE93  30 0003C         BSBW    GET_BKS
           5E            04  C0 0003F         ADDL2   #4, SP
           03            50  E8 00042 1$:     BLBS    STATUS, 2$
                       0140  31 00045         BRW     17$
0B         A7            51  90 00048 2$:     MOVB    BKT_SIZE, 11(BUF)           ;  2922
                      FEBD  30 0004C         BSBW    GET_KEY_PARMS               ;  2932
12         A7            53  90 0004F         MOVB    SEGS, 18(BUF)               ;  2934
13         A7       15   A6  90 00053         MOVB    21(XAB), 19(BUF)            ;  2935
```

```
                                14  A7              51  90 00058          MOVB    KEYSIZE, 20(BUF)                        ; 2936
                                15  A7        17    A6  90 0005C          MOVB    23(XAB), 21(BUF)                        ; 2937
                                16  A7              52  B0 00061          MOVW    MINREC, 22(BUF)                         ; 2938
                                    03      00B7    CA  91 00065          CMPB    183(IFAB), #3                           ; 2945
                                                    0C  1E 0006A          BGEQU   3$                                      ;
                                12  A6        48    8F  88 0006C          BISB2   #72, 18(XAB)                            ; 2949
                                          17  A6  95 00071                TSTB    23(XAB)                                 ; 2951
                                                    2E  12 00074          BNEQ    7$                                      ;
                                                    27  11 00076          BRB     6$                                      ;
                                    06              51  D1 00078  3$:     CMPL    KEYSIZE, #6                             ; 2966
                                                    05  19 0007B          BLSS    4$                                      ;
                                          13  A6  95 0007D                TSTB    19(XAB)                                 ; 2967
                                                    05  13 00080          BEQL    5$                                      ;
                                12  A6        48    8F  88 00082  4$:     BISB2   #72, 18(XAB)                            ; 2971
                                          17  A6  95 00087  5$:     TSTB    23(XAB)                                 ; 2978
                                                    18  12 0008A          BNEQ    7$                                      ;
                                          60  AA  B5 0008C                TSTW    96(IFAB)                                ; 2981
                                                    13  13 0008F          BEQL    7$                                      ;
                                    50              51  3C 00091          MOVZWL  KEYSIZE, R0                             ; 2982
                                    50              0B  C0 00094          ADDL2   #11, R0                                 ;
    50      60  AA                  10              00  ED 00097          CMPZV   #0, #16, 96(IFAB), R0                   ;
                                                    05  18 0009D          BGEQ    7$                                      ;
                                12  A6        80    8F  88 0009F  6$:     BISB2   #128, 18(XAB)                           ; 2984
                                    50  10  A7  9E 000A4  7$:     MOVAB   16(BUF), R0                             ; 2990
                                    51  12  A6  9E 000A8          MOVAB   18(XAB), R1                             ;
    60              01              00          61  F0 000AC          INSV    (R1), #0, #1, (R0)                      ;
                                    60          10  88 000B1          BISB2   #16, (R0)                               ; 2991
    52              61              01          03  EF 000B4          EXTZV   #3, #1, (R1), R2                        ; 2992
                                    52          52  D2 000B9          MCOML   R2, R2                                  ;
    60              01              03          52  F0 000BC          INSV    R2, #3, #1, (R0)                        ;
    52              61              01          06  EF 000C1          EXTZV   #6, #1, (R1), R2                        ; 2993
                                    52          52  D2 000C6          MCOML   R2, R2                                  ;
    60              01              06          52  F0 000C9          INSV    R2, #6, #1, (R0)                        ;
                                          17  A6  95 000CE                TSTB    23(XAB)                                 ; 2995
                                                    0F  12 000D1          BNEQ    8$                                      ;
    52              61              01          07  EF 000D3          EXTZV   #7, #1, (R1), R2                        ; 2997
                                    52          52  D2 000D8          MCOML   R2, R2                                  ;
    60              01              07          52  F0 000DB          INSV    R2, #7, #1, (R0)                        ;
                                                    14  11 000E0          BRB     9$                                      ;
    52              61              01          02  EF 000E2  8$:     EXTZV   #2, #1, (R1), R2                        ; 3000
    60              01              02          52  F0 000E7          INSV    R2, #2, #1, (R0)                        ;
    52              61              01          01  EF 000EC          EXTZV   #1, #1, (R1), R2                        ; 3001
    60              01              01          52  F0 000F1          INSV    R2, #1, #1, (R0)                        ;
                                                    53  DD 000F6  9$:     PUSHL   SEGS                                    ; 3006
                                          FE4D  30  000F8                BSBW    MOVE                                    ;
                                    5E            04  C0 000FB          ADDL2   #4, SP                                  ;
                                    03            50  E8 000FE          BLBS    STATUS, 10$                             ;
                                          0084  31  00101                BRW     17$                                     ;
                                    50      18  A7  9E 00104  10$:     MOVAB   24(BUF), R0                             ; 3010
                                    60      1A  A6  B0 00108          MOVW    26(XAB), (R0)                           ;
                                                    0A  12 0010C          BNEQ    11$                                     ; 3012
                                    51      0A  A7  9A 0010E          MOVZBL  10(BUF), R1                             ; 3014
            60                      51    0200  8F  A5 00112          MULW3   #512, R1, (R0)                          ;
                                    51      0A  A7  9A 00118  11$:     MOVZBL  10(BUF), R1                             ; 3017
            51                      51          08  78 0011C          ASHL    #8, R1, R1                              ;
                                    51          60  B1 00120          CMPW    (R0), R1                                ;
                                                    03  1E 00123          BGEQU   12$                                     ;
```

```
                                 60          51  B0 00125          MOVW    R1, (R0)                                ; 3019
                                 50    1A    A7  9E 00128 12$:     MOVAB   26(BUF), R0                             ; 3021
                                 60    1C    A6  B0 0012C          MOVW    28(XAB), (R0)                           ;
                                             0A  12 00130          BNEQ    13$                                     ; 3023
                                 51    0B    A7  9A 00132          MOVZBL  11(BUF), R1                             ; 3025
                     60          51  0200    8F  A5 00136          MULW3   #512, R1, (R0)                          ;
                                 51    0B    A7  9A 0013C 13$:     MOVZBL  11(BUF), R1                             ; 3028
                     51          51          08  78 00140          ASHL    #8, R1, R1                              ;
                                 51          60  B1 00144          CMPW    (R0), R1                                ;
                                             03  1E 00147          BGEQU   14$                                     ;
                                 60          51  B0 00149          MOVW    R1, (R0)                                ; 3030
                                 50    00B2  CA  9A 0014C 14$:     MOVZBL  178(IFAB), R0                           ; 3037
                                             50  D7 00151          DECL    R0                                      ;
     50          17    A6            08        00 ED 00153          CMPZV   #0, #8, 23(XAB), R0                     ;
                                             07  12 00159          BNEQ    15$                                     ;
                                             67  D4 0015B          CLRL    (BUF)                                   ; 3040
                                       04    A7  B4 0015D          CLRW    4(BUF)                                  ; 3041
                                             23  11 00160          BRB     16$                                     ; 3037
                                 50    17    A6  9A 00162 15$:     MOVZBL  23(XAB), R0                             ; 3045
                                 50          05  C0 00166          ADDL2   #5, R0                                  ;
                                 50          05  C6 00169          DIVL2   #5, R0                                  ;
                                 67    01    A0  9E 0016C          MOVAB   1(R0), (BUF)                            ;
                                 50    17    A6  9A 00170          MOVZBL  23(XAB), R0                             ; 3046
     7E          00              50          01  7A 00174          EMUL    #1, R0, #0, -(SP)                       ;
     50          50              8E          05  7B 00179          EDIV    #5, (SP)+, R0, R0                       ;
           04    A7              50    0066  8F  A5 0017E          MULW3   #102, R0, 4(BUF)                        ; 3047
                                 50          01  D0 00185 16$:     MOVL    #1, R0                                  ; 3050
                                             0C  BA 00188 17$:     POPR    #^M<R2,R3>                              ; 3052
                                                 05 0018A          RSB                                             ;

; Routine Size:  395 bytes,    Routine Base:  RM$RMSMISC + 0D6A
```

```
;3013          3053  1 %SBTTL  'WRITE_PROLOGUE'
;3014          3054  1 ROUTINE WRITE_PROLOGUE : RL$RW_PROLOG =
;3015          3055  1
;3016          3056  1 !++
;3017          3057  1 !
;3018          3058  1 ! WRITE_PROLOGUE
;3019          3059  1 !
;3020          3060  1 !       This routine recalculates the check sum, sets the buffer dirty, and
;3021          3061  1 !       writes it out.
;3022          3062  1 !
;3023          3063  1 !
;3024          3064  1 ! CALLING SEQUENCE:
;3025          3065  1 !       WRITE_PROLOGUE()
;3026          3066  1 !
;3027          3067  1 ! INPUT PARAMETERS:
;3028          3068  1 !       none
;3029          3069  1 !
;3030          3070  1 ! IMPLICIT INPUTS:
;3031          3071  1 !       BDB     - address of BDB describing buffer containing prologue VBN
;3032          3072  1 !
;3033          3073  1 ! OUTPUT PARAMETERS:
;3034          3074  1 !       none
;3035          3075  1 !
;3036          3076  1 ! IMPLICIT OUTPUTS:
;3037          3077  1 !       none
;3038          3078  1 !
;3039          3079  1 ! ROUTINE VALUE:
;3040          3080  1 !       WPL error
;3041          3081  1 !       Success code
;3042          3082  1 !
;3043          3083  1 ! SIDE EFFECTS:
;3044          3084  1 !
;3045          3085  1 !--
;3046          3086  1
;3047          3087  2     BEGIN
;3048          3088  2
;3049          3089  2     EXTERNAL REGISTER
;3050          3090  2         COMMON_FAB_STR,
;3051          3091  2         R_BDB_STR;
;3052          3092  2
;3053          3093  2     RMSMAKSUM(.BDB [BDB$L_ADDR]);
;3054          3094  2     BDB [BDB$V_VAL] = 1;
;3055          3095  2     BDB [BDB$V_DRT] = 1;
;3056          3096  2
;3057          3097  3     BEGIN
;3058          3098  3
;3059          3099  3     LOCAL
;3060          3100  3         FLAGS;
;3061          3101  3
;3062          3102  3     FLAGS = RLS$M_WRT_THRU;
;3063          3103  3
;3064          3104  3     IF .BDB [BDB$L_VBN] EQL 1
;3065          3105  3     THEN
;3066          3106  3         FLAGS = .FLAGS  OR  RLS$M_KEEP_LOCK;
;3067          3107  3
;3068        P 3108  3     RETURN_ON_ERROR (RMSRELEASE(.FLAGS),
;3069        P 3109  3         BEGIN
```

```
; 3070        P 3110  3          IF .FAB [FAB$L_STV] EQL 0
; 3071        P 3111  3          THEN
; 3072        P 3112  3              FAB [FAB$L_STV] = .STATUS OR 1^16;
; 3073        P 3113  3          STATUS = RMSERR (WPL)
; 3074          3114  3          END);
; 3075          3115  2      END;                                    . end of FLAGS declaration
; 3076          3116  2
; 3077          3117  2      RETURN RMSSUC();
; 3078          3118  2
; 3079          3119  1      END;


                                        2C  BB 00000 WRITE_PROLOGUE:
                                                          PUSHR   #^M<R2,R3,R5>                          ; 3054
                          55        18  A4  D0 00002      MOVL    24(BDB), R5                            ; 3093
                            00000000G   00  16 00006      JSB     RMSMAKSUM                              ;
                     0A   A4            03  88 0000C      BISB2   #3, 10(BDB)                            ; 3095
                          53            02  D0 00010      MOVL    #2, FLAGS                              ; 3102
                          01        1C  A4  D1 00013      CMPL    28(BDB), #1                            ; 3104
                                        03  12 00017      BNEQ    1$                                     ;
                          53            04  88 00019      BISB2   #4, FLAGS                              ; 3106
                            00000000G   00  16 0001C 1$:  JSB     RM$RELEASE                             ; 3114
                          15            50  E8 00022      BLBS    STATUS, 3$                             ;
                                    0C  A8  D5 00025      TSTL    12(FAB)                                ;
                                        09  12 00028      BNEQ    2$                                     ;
          0C   A8         50 00010000   8F  C9 0002A      BISL3   #65536, STATUS, 12(FAB)                ;
                          50      C11C  8F  3C 00033 2$:  MOVZWL  #49436, STATUS                         ;
                                        03  11 00038      BRB     4$                                     ;
                          50            01  D0 0003A 3$:  MOVL    #1, R0                                 ; 3117
                                        2C  BA 0003D 4$:  POPR    #^M<R2,R3,R5>                          ; 3119
                                            05 0003F      RSB                                            ;

; Routine Size:  64 bytes,    Routine Base:  RM$RMSMISC + 0EF5


; 3080          3120  1
; 3081          3121  1 END
; 3082          3122  1
; 3083          3123  0 ELUDOM



                                    PSECT SUMMARY
;
;       Name                    Bytes                         Attributes
;
;   RM$RMSMISC                     3893  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       248         7       154         00:00.4



; Information:  1
; Warnings:     0
; Errors:       0


;                                      COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3CREATE/OBJ=OBJ$:RM3CREATE MSRC$:RM3CREATE/UPDATE=(ENH$:RM3CREATE)

; Size:          3893 code + 0 data bytes
; Run Time:         01:27.7
; Elapsed Time:     02:32.2
; Lines/CPU Min:     2135
; Lexemes/CPU-Min: 19119
; Memory Used:  458 pages
; Compilation Complete
```

RM3FACE
LIS

RM3DISCON
LIS

RM3CONN
LIS

RM3DELETE
LIS

RM3CREATE
LIS