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

```
**FILE**ID**RM3MISC                                J 14                                                                                   RM3
                                                                                                                                          V04
```

```
RRRRRRRR    MM      MM    333333    MM      MM    IIIIII      SSSSSSSS    CCCCCCCC
RRRRRRRR    MM      MM    333333    MM      MM    IIIIII      SSSSSSSS    CCCCCCCC
RR      RR  MMMM  MMMM  33      33  MMMM  MMMM      II      SS          CC
RR      RR  MMMM  MMMM  33      33  MMMM  MMMM      II      SS          CC
RR      RR  MM  MM  MM          33  MM  MM  MM      II      SS          CC
RR      RR  MM  MM  MM          33  MM  MM  MM      II      SS          CC
RRRRRRRR    MM      MM        33    MM      MM      II        SSSSSS    CC
RRRRRRRR    MM      MM        33    MM      MM      II        SSSSSS    CC
RR  RR      MM      MM          33  MM      MM      II              SS  CC
RR  RR      MM      MM          33  MM      MM      II              SS  CC
RR    RR    MM      MM  33      33  MM      MM      II              SS  CC              ....
RR    RR    MM      MM  33      33  MM      MM      II              SS  CC              ....
RR      RR  MM      MM    333333    MM      MM    IIIIII    SSSSSSSS      CCCCCCCC      ....
RR      RR  MM      MM    333333    MM      MM    IIIIII    SSSSSSSS      CCCCCCCC      ....

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
```

```
0 MODULE RM3MISC (LANGUAGE (BLISS32) ,
0                 IDENT = 'V04-000'
0                 ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSAChUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 !
1 ! FACILITY:     RMS32 INDEX SEQUENTIAL FILE ORGANIZATION
1 !
1 ! ABSTRACT:
1 !               MISCELLANEOUS ROUTINES
1 !
1 !
1 ! ENVIRONMENT:
1 !
1 !               VAX/VMS OPERATING SYSTEM
1 !
1 !--
1
1 !
1 ! AUTHOR:       Wendy Koenig     CREATION DATE:         17-APR-78  9:57
1 !
1 ! MODIFIED BY:
1 !
1 !     V03-010   JWT0151         Jim Teague             31-Jan-1984
1 !               Under certain conditions, RM$RECORD_KEY can start
1 !               searching what it thinks is a SIDR bucket beginning
1 !               at LST_NCMP, but LST_NCMP happens to point to a
1 !               record in a primary data bucket.
1 !
1 !     V03-009   JWT0147         Jim Teague             12-Dec-1983
1 !               Correct insane sanity check on index buckets: on an
1 !               EXACTLY full index bucket it is not an error to have
1 !               the back freespace pointer point to a byte 1 less than
```

```
1 !                     the front freespace pointer.
1 !
1 !     V03-008         MCN0013         Maria del C. Nasr       15-Mar-1983
1 !                     More linkages reorganization
1 !
1 !     V03-007         MCN0012         Maria del C. Nasr       01-Mar-1983
1 !                     Reorganize linkages
1 !
1 !     V03-006         TMK0004         Todd M. Katz            13-Sep-1982
1 !                     Add support for prologue 3 SIDRs. This involved rewriting
1 !                     RM$RECORD_KEY and RM$CNTRL_ADDR, and making changes to
1 !                     RM$RECORD_VBN.
1 !
1 !                     Eliminate the routine RM$KEY_TYPE_CONV, a routine that is
1 !                     never used, and all calls to RM$CONV_TO_ASCII and
1 !                     RM$CONV_FROM_ASCII.
1 !
1 !     V03-005         KBT0221         Keith B. Thompson       23-Aug-1982
1 !                     Reorganize psects
1 !
1 !     V03-004         MCN0011         Maria del C. Nasr       29-Jun-1982
1 !                     Reverse parameters in call to RM$CONV_TO_ASCII in
1 !                     RM$RECORD_KEY.
1 !
1 !     V03-003         TMK0003         Todd M. Katz            28-Jun-1982
1 !                     I added subtitles in TMK0001 but I spelled the lexical
1 !                     function SBTTL incorrectly.
1 !
1 !     V03-002         TMK0002         Todd M. Katz            28-Jun-1982
1 !                     Add linakge for RM$RECORD_ID forgotten in TMK0001.
1 !
1 !     V03-001         TMK0001         Todd M. Katz            28-Jun-1982
1 !                     Add the new routine RM$RECORD_ID which extracts from the
1 !                     RRV field of the given primary data record the ID.
1 !
1 !     V02-017         PSK0005         Paulina S. Knibbe       02-Sep-1981
1 !                     Only add truncated character when the length of the
1 !                     currently expanded key is less than the total length
1 !
1 !     V02-016         MCN0010         Maria del C. Nasr       04-Aug-1981
1 !                     Modify RM$RECORD_KEY to do type conversion when extracting
1 !                     key segments from an expanded prologue 3 data record.
1 !                     Also, add RM$KEY_TYPE_CONV routine.
1 !
1 !     V02-015         PSK0005         Paulina S. Knibbe       30-Jul-1981
1 !                     Remove support for truncated index keys from RM$RECORD_KEY
1 !
1 !     V02-014         PSK0004         Paulina S. Knibbe       15-Jun-1981
1 !                     Change RM$RECORD_KEY to work for prologue three index
1 !                     and SIDR records, too.
1 !                     Change RM$CNTRL_ADDR to work for prologue three index buckets
1 !
1 !     V02-013         MCN0009         Maria del C. Nasr       07-May-1981
1 !                     Add support for front end compressed keys in RM$RECORD_KEY.
1 !
1 !     V02-012         MCN0008         Maria del C. Nasr       22-Apr-1981
1 !                     Fix some bugs with prologue 3 changes.
```

```
1 !
1 !     V02-011     PSK0003         Paulina S. Knibbe       17-Apr-1981
1 !                 Fix some problems w/RM$CNTRL_ADDR
1 !
1 !     V02-010     MCN0007         Maria del C. Nasr       13-Apr-1981
1 !                 Add RM$CHECK_SEGMENT routine.
1 !
1 !     V02-009     PSK0002         Paulina S. Knibbe       08-Apr-1981
1 !                 Add RM$CNTRL_ADDR to return the address of the control
1 !                 byte for the current record in any data bucket
1 !
1 !     V02-008     MCN0006         Maria del C. Nasr       23-Mar-1981
1 !                 Modify these routines to be able to process prologue 3
1 !                 data level structure changes (base level 1).
1 !
1 !     V02-007     PSK0001         Paulina S. Knibbe       12-Mar-1981
1 !                 Change the reference to segment length to a byte
1 !
1 !     V02-006     REFORMAT        Paulina S. Knibbe       23-Jul-1980
1 !
1 !
1 ! REVISION HISTORY:
1 !
1 !   Christian Saether,  28-SEP-78  8:52
1 !   X0002 - add RM$MOVE routine to avoid CH$MOVE problems
1 !
1 !   Christian Saether,  9-OCT-78  11:09
1 !   X0003 - modify RECORD_KEY to use routine REC_OVHD
1 !
1 !   Christian Saether,  10-OCT-78  10:14
1 !   X0004 - change RECORD_KEY to use REC_OVHD routine
1 !
1 !   Wendy Koenig,       24-OCT-78  14:02
1 !   X0005 - MAKE CHANGES CAUSED BY SHARING CONVENTIONS
1 !
1 !*****
1
1 LIBRARY 'RMSLIB:RMS';
1
1 REQUIRE 'RMSSRC:RMSIDXDEF';
1
1 ! define default psects for code
1 !
1
1 PSECT
1     CODE = RM$RMS3(PSECT_ATTR),
1     PLIT = RM$RMS3(PSECT_ATTR);
1
1 ! Linkages
1 !
1
1 LINKAGE
1     L_CHECK_SEGMENT,
1     L_PRESERVE1,
1     L_RABREG_567,
1     L_RABREG_67,
1     L_REC_OVHD,
;  172          0236  1       L_SIDR_FIRST;
;  173          0237  1
;  174          0238  1
;  175          0239  1  ! External Routines
;  176          0240  1  !
;  177          0241  1  EXTERNAL ROUTINE
;  178          0242  1       RM$SIDR_FIRST          : RL$SIDR_FIRST,
;  179          0243  1       RM$REC_OVHD            : RL$REC_OVHD;
```

```
1 %SBTTL 'RM$CHECK_SEGMENT'
1 GLOBAL ROUTINE RM$CHECK_SEGMENT( START_BUF, CURR_BYTE, ADDR_LEN ) : RL$CHECK_SEGMENT =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !       This routine determines if a given byte belongs to a segment
1 !       in the primary key.
1 !
1 ! CALLING SEQUENCE:
1 !       RM$CHECK_SEGMENT(PAR1,PAR2,PAR3)
1 !
1 ! INPUT PARAMETERS:
1 !       START_BUF - start address of input buffer if packing records
1 !                   or output buffer if unpacking
1 !       CURR_BYTE - address of current byte in buffer
1 !
1 ! IMPLICIT INPUT:
1 !       IDX descriptor (R7)
1 !
1 ! OUTPUT PARAMETER:
1 !       If not key segment:
1 !               ADDR_LEN = address of next segment
1 !       If key segment:
1 !               ADDR_LEN = length of key segment
1 !
1 ! ROUTINE VALUE:
1 !       0 - if not key segment
1 !       1 - if key segment
1 !
1 ! SIDE EFFECTS:
1 !       Unknown
1 !
1 !--
1
2    BEGIN
2
2    EXTERNAL REGISTER
2        R_IDX_DFN_STR;
2
2    LOCAL
2        X,
2        SEG_ADDR,
2        S_SEG_ADDR,
2        SEG_LEN :       BYTE,
2        SEG_DATA_ADDR;
2
2
2    SEG_DATA_ADDR = IDX_DFN[IDX$W_POSITION];
2    X = .IDX_DFN[IDX$B_SEGMENTS];
2
2    ! Determine the highest possible segment
2    !
2    S_SEG_ADDR = .I[X_DFN[IDX$B_DATBKTSZ] * 512 + .START_BUF;
2
2    WHILE .X NEQU 0
2    DO
```

```
;  238          0301  3          BEGIN
;  239          0302  3
;  240          0303  3          ! Get segment address and length
;  241          0304  3          !
;  242          0305  3          SEG_ADDR = .(.SEG_DATA_ADDR)<0,16> + .START_BUF;
;  243          0306  3          SEG_DATA_ADDR = .SEG_DATA_ADDR + 2;
;  244          0307  3          SEG_LEN = .(.SEG_DATA_ADDR)<0,8>;
;  245          0308  3          SEG_DATA_ADDR = .SEG_DATA_ADDR + 2;
;  246          0309  3          X = .X - 1;
;  247          0310  3
;  248          0311  3          IF .CURR_BYTE GEQU .SEG_ADDR
;  249          0312  3          THEN
;  250          0313  4              BEGIN
;  251          0314  4
;  252          0315  4              ! If the byte belongs to the primary key, return length between
;  253          0316  4              ! current byte and end of segment, and success.
;  254          0317  4              !
;  255          0318  5              IF .CURR_BYTE LSSU (.SEG_ADDR + .SEG_LEN)
;  256          0319  4              THEN
;  257          0320  5                  BEGIN
;  258          0321  5                  ADDR_LEN = (.SEG_ADDR + .SEG_LEN) - .CURR_BYTE;
;  259          0322  5                  RETURN 1
;  260          0323  5                  END
;  261          0324  4              END
;  262          0325  3          ELSE
;  263          0326  3
;  264          0327  3              ! If this segment is closer to current byte than previous segment
;  265          0328  3              ! but not before, note address
;  266          0329  3              !
;  267          0330  3              IF .SEG_ADDR LSSU .S_SEG_ADDR
;  268          0331  4                AND (.CURR_BYTE LSSU .SEG_ADDR)
;  269          0332  3              THEN
;  270          0333  3                  S_SEG_ADDR = .SEG_ADDR;
;  271          0334  2          END;                    ! end of while loop
;  272          0335  2
;  273          0336  2      ! Return address of closest segment to current byte
;  274          0337  2      !
;  275          0338  2      ADDR_LEN = .S_SEG_ADDR;
;  276          0339  2      RETURN 0;
;  277          0340  1      END;


                                                              .TITLE  RM3MISC
                                                              .IDENT  \V04-000\

                                                              .EXTRN  RM$SIDR_FIRST, RM$REC_OVHD

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

                                   034A   8F  BB 00000 RM$CHECK_SEGMENT::
                                                              PUSHR   #^M<R1,R3,R6,R8,R9>                       ; 0245
                           55       2C   A7  9E 00004          MOVAB   44(R7), SEG_DATA_ADDR                     ; 0292
                           58       1E   A7  9A 00008          MOVZBL  30(IDX_DFN), X                            ; 0293
                           51       17   A7  9A 0000C          MOVZBL  23(IDX_DFN), R1                           ; 0297
                  51       51            09  78 00010          ASHL    #9, R1, R1                                ;
                  56       51            50  C1 00014          ADDL3   START_BUF, R1, S_SEG_ADDR                 ;
                                         58  D5 00018 1$:      TSTL    X                                         ; 0299
                                   35  13 0001A             BEQL    3$                                         ;
                      51           85  3C 0001C             MOVZWL  (SEG_DATA_ADDR)+, SEG_ADDR                 ; 0305
                      51           50  C0 0001F             ADDL2   START_BUF, SEG_ADDR                        ;
                      59           85  90 00022             MOVB    (SEG_DATA_ADDR)+, SEG_LEN                  ; 0307
                                   55  D6 00025             INCL    SEG_DATA_ADDR                              ; 0308
                                   58  D7 00027             DECL    X                                          ; 0309
                      51           54  D1 00029             CMPL    CURR_BYTE, SEG_ADDR                        ; 0311
                                   14  1F 0002C             BLSSU   2$                                         ;
                      53           59  9A 0002E             MOVZBL  SEG_LEN, R3                                ; 0318
                      53           51  C0 00031             ADDL2   SEG_ADDR, R3                               ;
                      53           54  D1 00034             CMPL    CURR_BYTE, R3                              ;
                                   DF  1E 00037             BGEQU   1$                                         ;
          52          53           54  C3 00039             SUBL3   CURR_BYTE, R3, ADDR_LEN                    ; 0321
                      50           01  D0 0003D             MOVL    #1, R0                                     ; 0322
                                   14  11 00040             BRB     4$                                         ;
                      56           51  D1 00042 2$:         CMPL    SEG_ADDR, S_SEG_ADDR                       ; 0330
                                   D1  1E 00045             BGEQU   1$                                         ;
                      51           54  D1 00047             CMPL    CURR_BYTE, SEG_ADDR                        ; 0331
                                   CC  1E 0004A             BGEQU   1$                                         ;
                      56           51  D0 0004C             MOVL    SEG_ADDR, S_SEG_ADDR                       ; 0333
                                   C7  11 0004F             BRB     1$                                         ; 0299
                      52           56  D0 00051 3$:         MOVL    S_SEG_ADDR, ADDR_LEN                       ; 0338
                                   50  D4 00054             CLRL    R0                                         ; 0339
                          034A     8F  BA 00056 4$:         POPR    #^M<R1,R3,R6,R8,R9>                        ; 0340
                                       05 0005A             RSB                                                ;

; Routine Size:  91 bytes,    Routine Base:  RM$RMS3 + 0000
```

```
1 %SBTTL 'RM$CNTRL_ADDR'
1 GLOBAL ROUTINE RM$CNTRL_ADDR: RL$RABREG_567 =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine returns the address of the control byte for the current
1 !       record. For all prologue 1 and 2 records, and prologue 3 primary data
1 !       records, the control byte is associated with the rest of the record
1 !       overhead. For prologue 3 index records, the control byte is associated
1 !       with the VBN downpointer, and all VBN downpointers are found at the rear
1 !       of the bucket. For prologue 3 SIDRs, the control byte is associated with
1 !       the RRV pointer of the SIDR array's first element.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       RM$CNTRL_ADDR()
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       BKT_ADDR                        - address of bucket
1 !           BKT$W_FREESPACE             - offset to first free byte in bucket
1 !           BKT$B_INDEXNO               - index of bucket
1 !           BKT$B_LEVEL                 - level of bucket
1 !           BKT$V_PTR_SZ                - size of all VBN downpointers in bucket
1 !
1 !       IDX_DFN                         - address of index descriptor
1 !           IDX$B_IDXBKTSZ              - size of the index bucket
1 !
1 !       IFAB                            - address of IFAB
1 !           IFB$B_PLG_VER               - prologue version of the file
1 !
1 !       IRAB                            - address of IRAB
1 !           IRB$L_REC_COUNT             - number of preceeding records
1 !
1 !       REC_ADDR                        - address of the record
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       Address of the control byte.
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 !--
2   BEGIN
2
```

```
;  336        0398  2      EXTERNAL REGISTER
;  337        0399  2          R_BKT_ADDR_STR,
;  338        0400  2          R_IDX_DFN_STR,
;  339        0401  2          R_IFAB_STR,
;  340        0402  2          R_IRAB_STR,
;  341        0403  2          R_REC_ADDR_STR;
;  342        0404  2
;  343        0405  2      MACRO
;  344        0406  2          FREESPACE       = 0,0,16,0 %;
;  345        0407  2
;  346        0408  2      ! If this is a prologue 2 file, or a prologue 3 primary data record then the
;  347        0409  2      ! address of the record is the address of the record's control byte.
;  348        0410  2      !
;  349        0411  2      IF  .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  350        0412  2           OR
;  351        0413  3          (.BKT_ADDR[BKT$B_LEVEL] EQLU 0
;  352        0414  3               AND
;  353        0415  3              .BKT_ADDR[BKT$B_INDEXNO] EQLU 0)
;  354        0416  2      THEN
;  355        0417  2          RETURN .REC_ADDR
;  356        0418  2
;  357        0419  2      ! If this is a prologue 3 SIDR, then the address of the control byte of the
;  358        0420  2      ! first SIDR array element is returned.
;  359        0421  2      !
;  360        0422  2      ELSE
;  361        0423  2          IF .BKT_ADDR[BKT$B_LEVEL] EQLU 0
;  362        0424  2          THEN
;  363        0425  3              BEGIN
;  364        0426  3
;  365        0427  3              GLOBAL REGISTER
;  366        0428  3                  R_IMPURE,
;  367        0429  3                  R_RAB;
;  368        0430  3
;  369        0431  3              RETURN RM$SIDR_FIRST(0)
;  370        0432  3              END
;  371        0433  3
;  372        0434  3          ! This is a prologue 3 index record. The VBN downpointers are stored at
;  373        0435  3          ! the end of the bucket, and the address of the VBN downpointer
;  374        0436  3          ! corresponding to the current record is returned.
;  375        0437  3          !
;  376        0438  2          ELSE
;  377        0439  3              BEGIN
;  378        0440  3
;  379        0441  3              LOCAL
;  380        0442  3                  CONTROL         : REF BBLOCK,
;  381        0443  3                  VBN_SIZE;
;  382        0444  3
;  383        0445  3              ! Position to the back freespace pointer in th. index bucket, and
;  384        0446  3              ! verify that its value makes sense ( ie - it is no more than one
;  385        0447  3              ! byte less than the front freespace pointer, and not past the end
;  386        0448  3              ! of the bucket).
;  387        0449  3              !
;  388        0450  3              ! NOTE: On an EXACTLY full bucket, the back freespace pointer will
;  389        0451  3              !  be (correctly) one byte less than the front freespace pointer.
;  390        0452  3              !  Any further overlapping will be an error.
;  391        0453  3              !
;  392        0454  4              CONTROL = .BKT_ADDR + (.IDX_DFN[IDX$B_IDXBKTSZ] * 512)         ;
```

```
:  393        0455  3                            - BKT$C_ENDOVHD;
:  394        0456  3
:  395        0457  4             IF  .CONTROL[FREESPACE] LSSU (.BKT_ADDR[BKT$W_FREESPACE] - 1)
:  396        0458  3                   OR
:  397        0459  3                 .BKT_ADDR + .CONTROL[FREESPACE] GTRU .CONTROL
:  398        0460  3             THEN
:  399        0461  3                 BUG_CHECK;
:  400        0462  3
:  401        0463  3             ! Position to the VBN downpointer associated with this index record
:  402        0464  3             ! and return its address in the bucket.
:  403        0465  3             !
:  404        0466  3             VBN_SIZE = .BKT_ADDR[BKT$V_PTR_SZ] + 2;
:  405        0467  3             CONTROL  = .CONTROL - .VBN_SIZE;
:  406        0468  3             CONTROL  = .CONTROL - (.VBN_SIZE * .IRAB[IRB$L_REC_COUNT]);
:  407        0469  3
:  408        0470  3             RETURN .CONTROL;
:  409        0471  2             END;
:  410        0472  2
:  411        0473  1      END;



                                                              .EXTRN  RMS$BUG3

                             0904   8F  BB 00000 RM$CNTRL_ADDR::
                                                              PUSHR   #^M<R2,R8,R11>                       ; 0342
                     03      00B7   CA  91 00004              CMPB    183(IFAB), #3                        ; 0411
                                    0A  1F 00009              BLSSU   1$                                   ;
                               0C   A5  95 0000B              TSTB    12(BKT_ADDR)                         ; 0413
                                    0A  12 0000E              BNEQ    2$                                   ;
                               01   A5  95 00010              TSTB    1(BKT_ADDR)                          ; 0415
                                    05  12 00013              BNEQ    2$                                   ;
                     50             56  D0 00015 1$:          MOVL    REC_ADDR, R0                         ; 0423
                                    4E  11 00018              BRB     6$                                   ;
                               0C   A5  95 0001A 2$:          TSTB    12(BKT_ADDR)                         ;
                                    0A  12 0001D              BNEQ    3$                                   ;
                                    7E  D4 0001F              CLRL    -(SP)                                ; 0431
                                 0000G  30 00021              BSBW    RM$SIDR_FIRST                        ;
                     5E             04  C0 00024              ADDL2   #4, SP                               ;
                                    3F  11 00027              BRB     6$                                   ;
                     50        16   A7  9A 00029 3$:          MOVZBL  22(IDX_DFN), R0                      ; 0454
            50       50             09  78 0002D              ASHL    #9, R0, R0                           ;
                     52      FC A045    9E 00031              MOVAB   -4(R0)[BKT_ADDR], CONTROL            ; 0455
                     50        04   A5  3C 00036              MOVZWL  4(BKT_ADDR), R0                      ; 0457
                                    50  D7 0003A              DECL    R0                                   ;
    50      62       10             00  ED 0003C              CMPZV   #0, #16, (CONTROL), R0               ;
                                    0B  1F 00041              BLSSU   4$                                   ;
                     50             62  3C 00043              MOVZWL  (CONTROL), R0                        ; 0459
                     50             55  C0 00046              ADDL2   BKT_ADDR, R0                         ;
                     52             50  D1 00049              CMPL    R0, CONTROL                          ;
                                    03  1B 0004C              BLEQU   5$                                   ;
                                 0000G  30 0004E 4$:          BSBW    RM$BUG3                              ; 0460
    50     0D    A5  02             03  EF 00051 5$:          EXTZV   #3, #2, 13(BKT_ADDR), VBN_SIZE       ; 0466
                     50             02  C0 00057              ADDL2   #2, VBN_SIZE                         ;
                     52             50  C2 0005A              SUBL2   VBN_SIZE, CONTROL                    ; 0467
                     50      0094   C9  C4 0005D              MULL2   148(IRAB), R0                        ; 0468
                     52             50  C2 00062              SUBL2   R0, CONTROL                          ;
```

```
                                            50           52  D0 00065          MOVL      CONTROL, R0                              ; 0470
                                                  0904   8F  BA 00068 6$:      POPR      #^M<R2,R8,R11>                           ; 0473
                                                             05 0006C          RSB                                                ;

; Routine Size:  109 bytes,    Routine Base:  RM$RMS3 + 005B
;  413          0474  1 %SBTTL 'RM$MOVE'
;  414          0475  1 GLOBAL ROUTINE RM$MOVE (LENGTH, FROM_ADDR, TO_ADDR) : RL$PRESERVE1 =
;  415          0476  1
;  416          0477  1 !++
;  417          0478  1 !
;  418          0479  1 ! FUNCTIONAL DESCRIPTION:
;  419          0480  1 !
;  420          0481  1 !       The purpose of this routine is to move a block of characters from a
;  421          0482  1 !       source to a destination buffer. Its existance is do to the necessity
;  422          0483  1 !       of save registers R1 through R5 before doing a CH$MOVE, which is
;  423          0484  1 !       basically what this routine does.
;  424          0485  1 !
;  425          0486  1 ! CALLING SEQUENCE:
;  426          0487  1 !
;  427          0488  1 !       RM$MOVE ()
;  428          0489  1 !
;  429          0490  1 ! INPUT PARAMETERS:
;  430          0491  1 !
;  431          0492  1 !       LENGTH          - length of block to be moved
;  432          0493  1 !       FROM_ADDR       - address to move from
;  433          0494  1 !       TO_ADDR         - address to move to
;  434          0495  1 !
;  435          0496  1 ! IMPLICIT INPUTS:
;  436          0497  1 !       NONE
;  437          0498  1 !
;  438          0499  1 ! OUTPUT PARAMETERS:
;  439          0500  1 !       NONE
;  440          0501  1 !
;  441          0502  1 ! IMPLICIT OUTPUTS:
;  442          0503  1 !       NONE
;  443          0504  1 !
;  444          0505  1 ! ROUTINE VALUE:
;  445          0506  1 !
;  446          0507  1 !       The address of the first byte in the destination buffer past the
;  447          0508  1 !       block of characters moved.
;  448          0509  1 !
;  449          0510  1 ! SIDE EFFECTS:
;  450          0511  1 !       NONE
;  451          0512  1 !
;  452          0513  1 !--
;  453          0514  1
;  454          0515  2     BEGIN
;  455          0516  2     RETURN CH$MOVE(.LENGTH, .FROM_ADDR, .TO_ADDR);
;  456          0517  1     END;



                                                     3E  BB 00000 RM$MOVE::
                                                                       PUSHR   #^M<R1,R2,R3,R4,R5>                         ; 0475
                 20   BE       1C   BE       18   AE  28 00002         MOVC3   LENGTH, @FROM_ADDR, @TO_ADDR                ; 0516
                                    50            53  D0 00009         MOVL    R3, R0                                      ;
                                                  3E  BA 0000C         POPR    #^M<R1,R2,R3,R4,R5>                         ; 0517
                                                      05 0000E         RSB                                                 ;

; Routine Size:  15 bytes,    Routine Base:  RM$RMS3 + 00C8
```

```
1 %SBTTL 'RM$RECORD_ID'
1 GLOBAL ROUTINE RM$RECORD_ID : RL$RABREG_67 =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine extracts the ID from the primary data record's RRV field.
1 !
1 ! CALLING SEQUENCE:
1 !       BSBW RM$RECORD_ID()
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       IFAB                    - address of the IFAB
1 !           IFB$B_PLG_VER       - prologue version of the file
1 !
1 !       REC_ADDR                - address of the record
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       The ID of the given record
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 !--
1
2     BEGIN
2
2     EXTERNAL REGISTER
2         R_IFAB_STR,
2         R_REC_ADDR_STR;
2
2     BUILTIN
2         AP;
2
2     IF .IFAB[IFB$B_PLG_VER] EQLU 3
2     THEN
2         RETURN .(.REC_ADDR + 3)<0,16>
2     ELSE
2         RETURN .(.REC_ADDR + 2)<0,8>;
2
1     END;
```

```
           03      00B7   CA  91 00000 RM$RECORD_ID::
                                                 CMPB    183(IFAB), #3                 ; 0564
                          05  12 00005           BNEQ    1$                            ;
           50        03   A6  3C 00007           MOVZWL  3(REC_ADDR), R0               ; 0568
                              05 0000B           RSB                                   ;
           50        02   A6  9A 0000C 1$:       MOVZBL  2(REC_ADDR), R0               ;
                              05 00010           RSB                                   ; 0570

; Routine Size:  17 bytes,    Routine Base:  RM$RMS3 + 00D7
```

```
1 %SBTTL 'RMSRECORD_KEY'
1 GLOBAL ROUTINE RMSRECORD_KEY (OUTBUF) : RL$PRESERVE1 =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine extracts a key from a record and places it the output
1 !       buffer, the address of which is passed to it as an arguement. The
1 !       key that is extracted is an index key, if the record is an index record,
1 !       an alternate key, if the record is a SIDR, or either the primary key
1 !       or an alternate key, if the record is a primary data record. In the
1 !       latter case, which key is extracted depends upon the index descriptor
1 !       this routine recieves as implicit input. If the index descriptor is for
1 !       the primary key of reference then it will be the primary key that is
1 !       extracted from the primary data record; otherwise, it will be an
1 !       alternate key.
1 !
1 !       This routine maybe called indicating either that the record has overhead
1 !       data associated with it, or that REC_ADDR points directly to the record
1 !       itself. In the former case, RMS will always first position past the
1 !       record overhead to the record itself, before joining the common code to
1 !       extract the appropriate key. This routine also maybe called indicating
1 !       either that the record is compressed format (prolgoue 3 only), or is
1 !       not, and the routine takes the appropriate action in each case.
1 !
1 !       This routine makes one very important assumption. If the record is a
1 !       primary data record and the index descriptor is for a secondary key,
1 !       inotherwards RMS is to extract a secondary key from a primary data
1 !       record, then the primary data record can not be in compressed format
1 !       because it would then be impossible to find let alone extract out the
1 !       alternate key.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       RMSRECORD_KEY()
1 !
1 ! INPUT PARAMETERS:
1 !
1 !       OUTBUF          - address of the buffer to contain extracted key
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       AP                      - used to control information to the` routine
1 !           bit 0               - if 0, record overhead
1 !                               - if 1, no record overhead
1 !           bit 1               - if 0, compressed form (prologue 3 only)
1 !                               - if 1, expanded form
1 !
1 !       BKT_ADDR                - address of bucket
1 !           BKT$B_INDEXNO       - index bucket is in
1 !           BKT$B_LEVEL         - level of bucket
1 !
1 !       IDX_DFN                 - address of index descriptor
1 !           IDX$V_IDX_COMPR     - if set, index key compression is enabled
1 !           IDX$V_KEY_COMPR     - if set, key compression is enabled
1 !           IDX$B_KEYSZ         - size of key
```

```
;  569        0628  1 !                 IDX$W_POSITION      - table of segment positions
;  570        0629  1 !                 IDX$B_SEGMENTS      - number of segement
;  571        0630  1 !                 IDX$B_SIZE          - table of segment sizes
;  572        0631  1 !                 IDX$B_TYPE          - table of segment types
;  573        0632  1 !
;  574        0633  1 !             IFAB                    - address of IFAB
;  575        0634  1 !                 IFB$B_PLG_VER       - prologue version of file
;  576        0635  1 !
;  577        0636  1 !             IRAB                    - address of IRAB
;  578        0637  1 !                 IRB$L_CURBDB        - address of BDB for current record's buffer
;  579        0638  1 !                 IRB$L_LST_NCMP      - address of last noncompressed key in bucket
;  580        0639  1 !
;  581        0640  1 !             REC_ADDR                - address of current record
;  582        0641  1 !
;  583        0642  1 ! OUTPUT PARAMETERS:
;  584        0643  1 !             NONE
;  585        0644  1 !
;  586        0645  1 ! IMPLICIT OUTPUTS:
;  587        0646  1 !             NONE
;  588        0647  1 !
;  589        0648  1 ! ROUTINE VALUE:
;  590        0649  1 !
;  591        0650  1 !             Address of first byte in output buffer past extracted key.
;  592        0651  1 !
;  593        0652  1 ! SIDE EFFECTS:
;  594        0653  1 !
;  595        0654  1 !             AP is trashed.
;  596        0655  1 !
;  597        0656  1 !--
;  598        0657  1
;  599        0658  2       BEGIN
;  600        0659  2
;  601        0660  2       BUILTIN
;  602        0661  2           AP;
;  603        0662  2
;  604        0663  2       EXTERNAL REGISTER
;  605        0664  2           R_IDX_DFN_STR,
;  606        0665  2           R_IFAB_STR,
;  607        0666  2           R_IRAB_STR,
;  608        0667  2           R_REC_ADDR_STR;
;  609        0668  2
;  610        0669  2       LOCAL
;  611        0670  2           START_ADDR :    REF BBLOCK;
;  612        0671  2
;  613        0672  2       ! Define macros to identify compressed key overhead.
;  614        0673  2       !
;  615        0674  2       MACRO
;  616        0675  2           KEY_LEN = 0,0,8,0 %,
;  617        0676  2           CMP_CNT = 1,0,8,0 %;
;  618        0677  2
;  619        0678  2       START_ADDR = .REC_ADDR;
;  620        0679  2
;  621        0680  2       ! If record overhead is indicated, position past it to the record proper.
;  622        0681  2       !
;  623        0682  2       IF NOT .AP<0,1>
;  624        0683  2       THEN
;  625        0684  3           BEGIN
;  626          0685  3
;  627          0686  3             LOCAL
;  628          0687  3                 REC_SIZE;
;  629          0688  3
;  630          0689  3             REC_SIZE = .BBLOCK [.BBLOCK [.IRAB[IRB$L_CURBDB], BDB$L_ADDR],
;  631          0690  3                                 BKT$B_LEVEL];
;  632          0691  3
;  633          0692  3             IF .REC_SIZE EQLU 0
;  634          0693  3                 AND
;  635          0694  3                 .IDX_DFN[IDX$B_KEYREF] NEQU 0
;  636          0695  3             THEN
;  637          0696  3                 REC_SIZE = -1;
;  638          0697  3
;  639          0698  3             START_ADDR = .START_ADDR + RM$REC_OVHD(.REC_SIZE; REC_SIZE);
;  640          0699  2             END;
;  641          0700  2
;  642          0701  2         ! The file is a prologue 1 or 2 file; or, the file is a prologue 3 file,
;  643          0702  2         ! but the record is not in compressed form. RMS only has to simple extract
;  644          0703  2         ! each segment from the appropriate position in the record, and move it
;  645          0704  2         ! into the keybuffer.
;  646          0705  2         !
;  647          0706  2         IF   .IFAB[IFB$B_PLG_VER] LSSU PLG$C_VER_3
;  648          0707  2              OR
;  649          0708  2              .AP<1,1>
;  650          0709  2         THEN
;  651          0710  2             INCR SEG_DATA_ADDR
;  652          0711  2                     FROM IDX_DFN[IDX$W_POSITION]
;  653          0712  3                     TO   (IDX_DFN[IDX$W_POSITION]
;  654          0713  4                           + (4 * .IDX_DFN[IDX$B_SEGMENTS])
;  655          0714  3                           - 4)
;  656          0715  2                     BY 4
;  657          0716  2             DO
;  658          0717  3                 BEGIN
;  659          0718  3
;  660          0719  3                 GLOBAL REGISTER
;  661          0720  3                     R_RAB,
;  662          0721  3                     R_IMPURE,
;  663          0722  3                     R_BDB;
;  664          0723  3
;  665          0724  3                 OUTBUF = RM$MOVE (.(.SEG_DATA_ADDR)<16,8>,
;  666          0725  3                                   .START_ADDR + .(.SEG_DATA_ADDR)<0,16>,
;  667          0726  3                                   .OUTBUF)
;  668          0727  3                 END
;  669          0728  3
;  670          0729  3         ! The record is in compressed format. It can either be a primary data
;  671          0730  3         ! record, ar index record or a SIDR. The desired key is extracted as a
;  672          0731  3         ! whole with expansion of the key being done, if the key is compressed.
;  673          0732  3         !
;  674          0733  2         ELSE
;  675          0734  3             BEGIN
;  676          0735  3
;  677          0736  3             LOCAL
;  678          0737  3                 BUCKET : REF BBLOCK;
;  679          0738  3
;  680          0739  3             BUCKET = .BBLOCK[.IRAB[IRB$L_CURBDB], BDB$L_ADDR];
;  681          0740  3
;  682          0741  3             ! If the key has been compressed, then extraction of the key must be
;  683          0742  3             ! accompanied by the addition of the front compressed and rear-end
;  684          0743  3             ! truncated characters.
;  685          0744  3             !
;  686          0745  4             IF (.BUCKET[BKT$B_LEVEL] EQLU 0
;  687          0746  4                     AND
;  688          0747  4                   .IDX_DFN[IDX$V_KEY_COMPR])
;  689          0748  3                 OR
;  690          0749  4                (.BUCKET[BKT$B_LEVEL] NEQU 0
;  691          0750  4                     AND
;  692          0751  4                   .IDX_DFN[IDX$V_IDX_COMPR])
;  693          0752  3             THEN
;  694          0753  4                 BEGIN
;  695          0754  4
;  696          0755  4                 LOCAL
;  697          0756  4                     LENGTH,
;  698          0757  4                     SAVE_REC_ADDR;
;  699          0758  4
;  700          0759  4                 SAVE_REC_ADDR = .REC_ADDR;
;  701          0760  4
;  702          0761  4                 ! Position to the first byte in the output buffer past the
;  703          0762  4                 ! number of bytes front compressed in the key that is to be
;  704          0763  4                 ! returned.
;  705          0764  4                 !
;  706          0765  4                 OUTBUF = .OUTBUF + .START_ADDR[CMP_CNT];
;  707          0766  4
;  708          0767  4                 ! Scan the bucket until the desired record is reached, moving the
;  709          0768  4                 ! the characters front compressed off the key of the desired
;  710          0769  4                 ! record into the output buffer as they are encountered. The bucket
;  711          0770  4                 ! scan starts with the first record in the bucket
;  712          0771  4                 !
;  713          0772  4                 REC_ADDR = .BUCKET + BKT$C_OVERHDSZ;
;  714          0773  4
;  715          0774  4                 WHILE 1 DO
;  716          0775  5                 BEGIN
;  717          0776  5
;  718          0777  5                     LOCAL
;  719          0778  5                         RECORD_OVHD,
;  720          0779  5                         RECORD_SIZE;
;  721          0780  5
;  722          0781  5                     ! Position to the key of the current record. This will involve
;  723          0782  5                     ! determining the number of bytes of record overhead.
;  724          0783  5                     !
;  725          0784  6                     BEGIN
;  726          0785  6
;  727          0786  6                     LOCAL
;  728          0787  6                         REC_SIZE;
;  729          0788  6
;  730          0789  6                     ! Set REC_SIZE according to the bucket type.
;  731          0790  6                     !
;  732          0791  6                     IF (REC_SIZE = .BUCKET[BKT$B_LEVEL]) EQLU 0
;  733          0792  6                     THEN
;  734          0793  6                         IF .BUCKET[BKT$B_INDEXNO] NEQU 0
;  735          0794  6                         THEN
;  736          0795  6                             REC_SIZE = -1;
;  737          0796  6
;  738          0797  6                     RECORD_OVHD = RM$REC_OVHD(.REC_SIZE; REC_SIZE);
;  739          0798  6                     RECORD_SIZE = .REC_SIZE;
;  740          0799  5                 END;
;  741          0800  5
;  742          0801  5                 ! If the desired record has been reached in the bucket scan,
;  743          0802  5                 ! then terminate the scan.
;  744          0803  5                 !
;  745          0804  5                 IF (REC_ADDR = .REC_ADDR + .RECORD_OVHD) GEQU .START_ADDR
;  746          0805  5                 THEN
;  747          0806  5                     EXITLOOP;
;  748          0807  5
;  749          0808  5                 ! If the front compression count of the key of the current
;  750          0809  5                 ! record is less than the compression count of the key of
;  751          0810  5                 ! the desired record, then the former has characters that
;  752          0811  5                 ! the latter requires in its expansion.
;  753          0812  5                 !
;  754          0813  5                 IF .REC_ADDR[CMP_CNT] LSSU .START_ADDR[CMP_CNT]
;  755          0814  5                 THEN
;  756          0815  6                     BEGIN
;  757          0816  6
;  758          0817  6                     ! If the compression count is equal to zero, move all
;  759          0818  6                     ! the characters.  Otherwise, RMS had previously moved
;  760          0819  6                     ! characters that now appear to be incorrect, so
;  761          0820  6                     ! overlay them with what RMS hopes are the correct ones.
;  762          0821  6                     !
;  763          0822  6                     LENGTH = .START_ADDR[CMP_CNT] - .REC_ADDR[CMP_CNT];
;  764          0823  6                     OUTBUF = .OUTBUF - .LENGTH;
;  765          0824  6
;  766          0825  6                     ! Move all of the front compressed characters needed by
;  767          0826  6                     ! the key of the desired record that can be supplied by
;  768          0827  6                     ! the key of the current record into the outbuf buffer,
;  769          0828  6                     ! utilizing the truncated character of the key of the
;  770          0829  6                     ! current record to supply any of these characters as
;  771          0830  6                     ! needed.
;  772          0831  6                     !
;  773          0832  6                     OUTBUF = CH$COPY (.REC_ADDR[KEY_LEN],
;  774          0833  6                                       .REC_ADDR + 2,
;  775          0834  6                                       .(.REC_ADDR + .REC_ADDR[KEY_LEN] + 1),
;  776          0835  6                                       .LENGTH,
;  777          0836  6                                       .OUTBUF);
;  778          0837  5                     END;
;  779          0838  5
;  780          0839  5                 ! Position to the next record in the bucket;
;  781          0840  5                 !
;  782          0841  5                 REC_ADDR = .REC_ADDR + .RECORD_SIZE;
;  783          0842  4                 END;                                    ! end of WHILE loop
;  784          0843  4
;  785          0844  4             ! Complete the key of the desired record with those characters
;  786          0845  4             ! not front compressed - extending the key out to its full size
;  787          0846  4             ! using its rear-end truncated character, if it is required.
;  788          0847  4             !
;  789          0848  4             LENGTH = .IDX_DFN[IDX$B_KEYSZ] - .START_ADDR[CMP_CNT];
;  790          0849  4
;  791          0850  4             IF .LENGTH GTR 0
;  792          0851  4             THEN
;  793          0852  4                 OUTBUF = CH$COPY (.START_ADDR[KEY_LEN],
;  794          0853  4                                   .START_ADDR + 2,
;  795          0854  4                                   .(.START_ADDR + .START_ADDR[KEY_LEN] + 1),
;  796          0855  4                                   .LENGTH,
;  797          0856  4                                   .OUTBUF);
;  798          0857  4
;  799          0858  4             REC_ADDR = .SAVE_REC_ADDR;
;  800          0859  4             END
;  801          0860  3         ELSE
;  802          0861  3
;  803          0862  3             ! The record is in compressed form, but the key which is to be
;  804          0863  3             ! extracted is not itself compressed. Therefore, it maybe moved
;  805          0864  3             ! as a single entity into the output buffer.
;  806          0865  3             !
;  807          0866  3             OUTBUF = CH$MOVE( .IDX_DFN[IDX$B_KEYSZ], .START_ADDR, .OUTBUF);
;  808          0867  3
;  809          0868  2         END;
;  810          0869  2
;  811          0870  2     ! Return the address of the first byte in the output buffer, past the
;  812          0871  2     ! the key which has been extracted from the current record, and placed
;  813          0872  2     ! there.
;  814          0873  2     !
;  815          0874  2     RETURN .OUTBUF;
;  816          0875  1     END.
```

```
                                             093E    8F  BB 00000 RM$RECORD_KEY::
                                                                          PUSHR   #^M<R1,R2,R3,R4,R5,R8,R11>                   ; 0572
                                       5E            0C  C2 00004         SUBL2   #12, SP                                      ;
                                                     56  DD 00007         PUSHL   REC_ADDR                                     ; 0678
                                       1C            5C  E8 00009         BLBS    AP, 2$                                       ; 0682
                                       50      20    A9  D0 0000C         MOVL    32(IRAB), R0                                 ; 0689
                                       50      18    A0  D0 00010         MOVL    24(R0), R0                                   ; 0690
                                       51      0C    A0  9A 00014         MOVZBL  12(R0), REC_SIZE                             ; 0689
                                                     08  12 00018         BNEQ    1$                                           ; 0692
                                               21    A7  95 0001A         TSTB    33(IDX_DFN)                                  ; 0694
                                                     03  13 0001D         BEQL    1$                                           ;
                                       51            01  CE 0001F         MNEGL   #1, REC_SIZE                                 ; 0696
                                                  0000G  30 00022 1$:     BSBW    RM$REC_OVHD                                  ; 0698
                                       6E            50  C0 00025         ADDL2   R0, START_ADDR                               ;
                                       03    00B7    CA  91 00028 2$:     CMPB    183(IFAB), #3                                ; 0706
                                                     04  1F 0002D         BLSSU   3$                                           ;
                         2F            5C            01  E1 0002F         BBC     #1, AP, 6$                                   ; 0708
                                       50      1E    A7  9A 00033 3$:     MOVZBL  30(IDX_DFN), R0                              ; 0713
                                       52      28  A740  DE 00037         MOVAL   40(IDX_DFN)[R0], R2                          ; 0712
                                       51      28    A7  9E 0003C         MOVAB   40(R7), SEG_DATA_ADDR                        ; 0711
                                                     17  11 00040         BRB     5$                                           ;
                                               30    AE  DD 00042 4$:     PUSHL   OUTBUF                                       ; 0726
                                       50            61  3C 00045         MOVZWL  (SEG_DATA_ADDR), R0                          ; 0725
                                               04  BE40  9F 00048         PUSHAB  @START_ADDR[R0]                              ;
                                       7E      02    A1  9A 0004C         MOVZBL  2(SEG_DATA_ADDR), -(SP)                      ; 0724
                                                     8E  10 00050         BSBB    RM$MOVE                                      ;
                                       5E            0C  C0 00052         ADDL2   #12, SP                                      ;
                                  30   AE            50  D0 00055         MOVL    R0, OUTBUF                                   ;
          FFE3           51            04            52  F1 00059 5$:     ACBL    R2, #4, SEG_DATA_ADDR, 4$                    ;
                                                   00BE  31 0005F         BRW     16$                                          ; 0711
                                       50      20    A9  D0 00062 6$:     MOVL    32(IRAB), R0                                 ; 0739
                                       58      18    A0  D0 00066         MOVL    24(R0), BUCKET                               ;
```

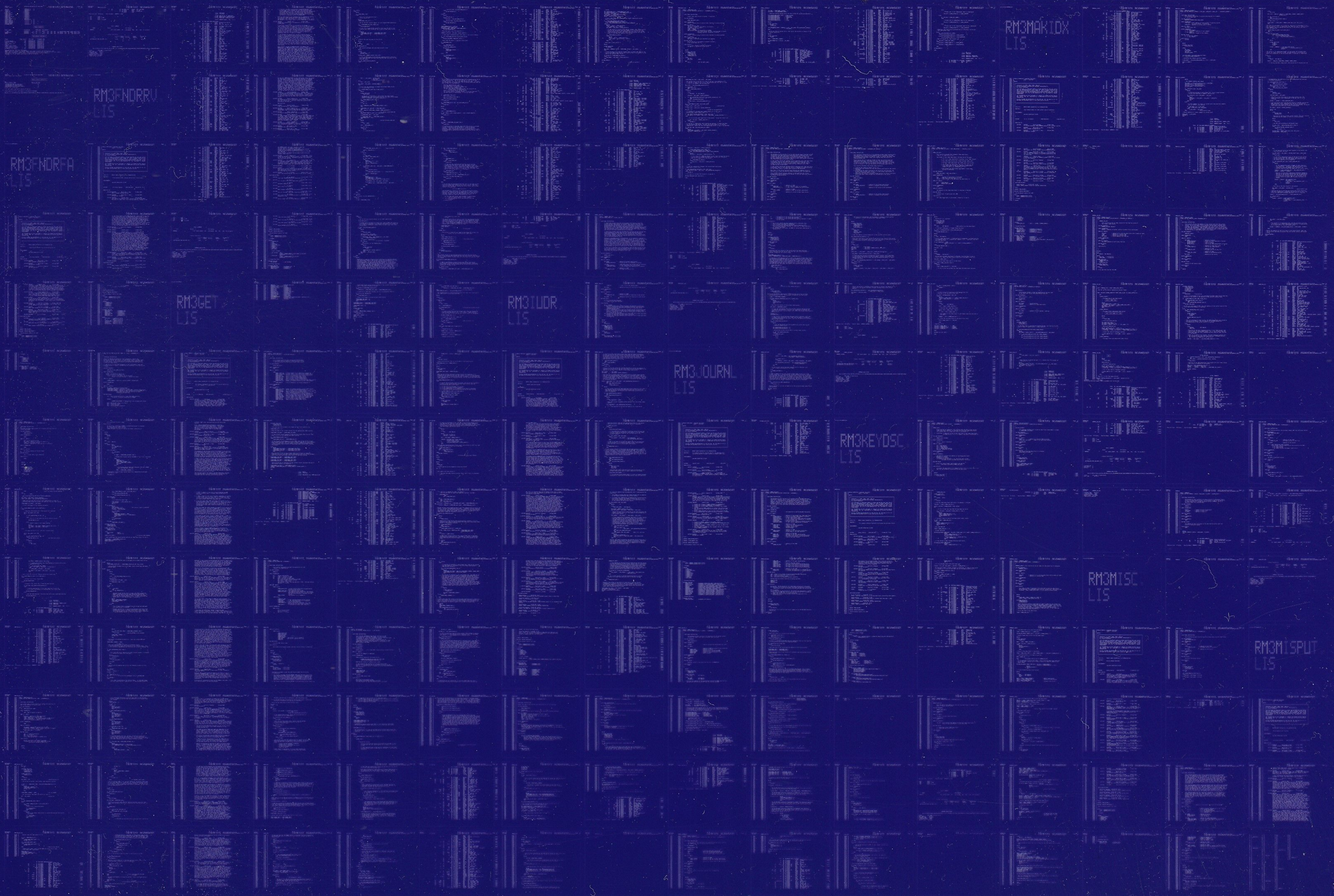
```
                             04    AE       0C     A8 9A 0006A         MOVZBL  12(BUCKET), 4(SP)                                 ; 0745
                                                   0A 12 0006F         BNEQ    8$                                                ;
                      0A     1C    A7              06 E0 00071         BBS     #6, 28(IDX_DFN), 9$                               ; 0747
                                                   03 12 00076         BNEQ    8$                                                ; 0749
                                                 0097 31 00078 7$:     BRW     15$                                               ;
                      F8     1C    A7              03 E1 0007B 8$:     BBC     #3, 28(IDX_DFN), 7$                               ; 0751
                             0C    AE              56 D0 00080 9$:     MOVL    REC_ADDR, SAVE_REC_ADDR                           ; 0759
                      51           6E              01 C1 00084         ADDL3   #1, START_ADDR, R1                                ; 0765
                                   50              61 9A 00088         MOVZBL  (R1), R0                                          ;
                             30    AE              50 C0 0008B         ADDL2   R0, OUTBUF                                        ;
                                   56       0E     A8 9E 0008F         MOVAB   14(R8), REC_ADDR                                  ; 0772
                                   51       04     AE D0 00093 10$:    MOVL    4(SP), REC_SIZE                                   ; 0791
                                                   08 12 00097         BNEQ    11$                                               ;
                                            01     A8 95 00099         TSTB    1(BUCKET)                                         ; 0793
                                                   03 13 0009C         BEQL    11$                                               ;
                                   51              01 CE 0009E         MNEGL   #1, REC_SIZE                                      ; 0795
                                                0000G 30 000A1 11$:    BSBW    RM$REC_OVHD                                       ; 0797
                             08    AE              51 D0 000A4         MOVL    REC_SIZE, RECORD_SIZE                             ; 0798
                                   56              50 C0 000A8         ADDL2   RECORD_OVHD, REC_ADDR                             ; 0804
                                   6E              56 D1 000AB         CMPL    REC_ADDh, START_ADDR                              ;
                                                   33 1E 000AE         BGEQU   13$                                               ;
                      50           6E              01 C1 000B0         ADDL3   #1, START_ADDR, R0                                ; 0813
                                   60       01     A6 91 000B4         CMPB    1(REC_ADDR), (R0)                                 ;
                                                   23 1E 000B8         BGEQU   12$                                               ;
                      50           6E              01 C1 000BA         ADDL3   #1, START_ADDR, R0                                ; 0822
                                   5B              60 9A 000BE         MOVZBL  (R0), LENGTH                                      ;
                                   51       01     A6 9A 000C1         MOVZBL  1(REC_ADDR), R1                                   ;
                                   5B              51 C2 000C5         SUBL2   R1, LENGTH                                        ;
                             30    AE              5B C2 000C8         SUBL2   LENGTH, OUTBUF                                    ; 0823
                                   50              66 9A 000CC         MOVZBL  (REC_ADDR), R0                                    ; 0832
        5B       01 A046     02    A6              50 2C 000CF         MOVC5   R0, 2(REC_ADDR), 1(R0)[REC_ADDR], LENGTH, -       ; 0836
                                            30     BE    000D7                 @OUTBUF                                           ;
                             30    AE              53 D0 000D9         MOVL    R3, OUTBUF                                        ;
                                   56       08     AE C0 000DD 12$:    ADDL2   RECORD_SIZE, REC_ADDR                             ; 0841
                                                   B0 11 000E1         BRB     10$                                               ; 0774
                                   50       20     A7 9A 000E3 13$:    MOVZBL  32(IDX_DFN), R0                                   ; 0848
                      51           6E              01 C1 000E7         ADDL3   #1, START_ADDR, R1                                ;
                                   5B              61 9A 000EB         MOVZBL  (R1), LENGTH                                      ;
                      5B           50              5B C3 000EE         SUBL3   LENGTH, R0, LENGTH                                ;
                                                   18 15 000F2         BLEQ    14$                                               ; 0850
                                   50       00     BE 9A 000F4         MOVZBL  @START_ADDR, R0                                   ; 0852
                                   58              6E D0 000F8         MOVL    START_ADDR, R8                                    ; 0856
                      7E           6E              02 C1 000FB         ADDL3   #2, START_ADDR, -(SP)                             ;
        5B       01 A048           9E              50 2C 000FF         MOVC5   R0, @(SP)+, 1(R0)[R8], LENGTH, @OUTBUF            ;
                                            30     BE    00106                                                                   ;
                             30    AE              53 D0 00108         MOVL    R3, OUTBUF                                        ;
                                   56       0C     AE D0 0010C 14$:    MOVL    SAVE_REC_ADDR, REC_ADDR                           ; 0858
                                                   0E 11 00110         BRB     16$                                               ; 0745
                                   50       20     A7 9A 00112 15$:    MOVZBL  32(IDX_DFN), R0                                   ; 0866
                 30   BE     00    BE              50 28 00116         MOVC3   R0, @START_ADDR, @OUTBUF                          ;
                             30    AE              53 D0 0011C         MOVL    R3, OUTBUF                                        ;
                                   50       30     AE D0 00120 16$:    MOVL    OUTBUF, R0                                        ; 0874
                                   5E              10 C0 00124         ADDL2   #16, SP                                           ; 0875
                                          093E     8F BA 00127         POPR    #^M<R1,R2,R3,R4,R5,R8,R11>                        ;
                                                      05 0012B         RSB                                                       ;

; Routine Size:  300 bytes,    Routine Base:  RM$RMS3 + 00E8
```

```
1 %SBTTL 'RMS$RECORD_VBN'
1 GLOBAL ROUTINE RMS$RECORD_VBN : RL$PRESERVE1 =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine extracts the variable length VBN from the given record.
1 !
1 ! CALLING SEQUENCE:
1 !       BSBW RMS$RECORD_VBN()
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       IFAB                    - address of the IFAB
1           IFB$B_PLG_VER       - prologue version of the file
1 !
1 !       REC_ADDR                - address of the record
1 !
1 !       AP -- code indicating type of bucket
1 !             (also offset from the beginning of the record to the VBN)
1 !               3 for DATA records
1 !               2 for SIDR records
1 !               1 for INDEX records (Prologue 1 and 3 only)
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       The VBN of the given record.
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 !--
1
2    BEGIN
2
2    EXTERNAL REGISTER
2       R_IFAB_STR,
2       R_REC_ADDR_STR;
2
2    BUILTIN
2       AP;
2
2    IF .IFAB[IFB$B_PLG_VER] EQLU 3
2    THEN
2       IF .AP EQLU 3
2       THEN
2           RETURN .(.REC_ADDR + 5)<0,8*(2 + .REC_ADDR[IRC$V_PTRSZ])>
```

```
;  875          0933  2          ELSE
;  876          0934  2              RETURN .(.REC_ADDR + 3)<0,8*(2 + .REC_ADDR[IRC$V_PTRSZ])>
;  877          0935  2      ELSE
;  878          0936  2          RETURN .(.REC_ADDR + .AP)<0,8*(2 + .REC_ADDR[IRC$V_PTRSZ])>;
;  879          0937  2
;  880          0938  1      END;



                                               51  DD 00000 RM$RECORD_VBN::
                                                                   PUSHL   R1                                          ; 0877
                                  03     00B7  CA  91 00002        CMPB    183(IFAB), #3                               ; 0928
                                               2E  12 00007        BNEQ    3$                                          ;
                                  03           5C  D1 00009        CMPL    AP, #3                                      ; 0930
                                               13  12 0000C        BNEQ    1$                                          ;
         50             66        02           00  EF 0000E        EXTZV   #0, #2, (REC_ADDR), R0                      ; 0932
                                  50           08  C4 00013        MULL2   #8, R0                                      ;
                                  50           10  C0 00016        ADDL2   #16, R0                                     ;
         51       05    A6        50           00  EF 00019        EXTZV   #0, R0, 5(REC_ADDR), R1                     ;
                                               11  11 0001F        BRB     2$                                          ;
         50             66        02           00  EF 00021 1$:    EXTZV   #0, #2, (REC_ADDR), R0                      ; 0934
                                  50           08  C4 00026        MULL2   #8, R0                                      ;
                                  50           10  C0 00029        ADDL2   #16, R0                                     ;
         51       03    A6        50           00  EF 0002C        EXTZV   #0, R0, 3(REC_ADDR), R1                     ;
                                  50           51  D0 00032 2$:    MOVL    R1, R0                                      ;
                                               11  11 00035        BRB     4$                                          ; 0936
         50             66        02           00  EF 00037 3$:    EXTZV   #0, #2, (REC_ADDR), R0                      ;
                                  50           08  C4 0003C        MULL2   #8, R0                                      ;
                                  50           10  C0 0003F        ADDL2   #16, R0                                     ;
         51           6C46        50           00  EF 00042        EXTZV   #0, R0, (AP)[REC_ADDR], R1                  ;
                                  50           51  D0 00048 4$:    MOVL    R1, R0                                      ;
                                               02  BA 0004B        POPR    #^M<R1>                                     ; 0938
                                                   05 0004D        RSB                                                 ;

; Routine Size:  78 bytes,    Routine Base:  RM$RMS3 + 0214


;  881          0939  1
;  882          0940  1
;  883          0941  1 END
;  884          0942  0 ELUDOM




;                                            PSECT SUMMARY
;
;        Name                           Bytes                        Attributes
;
;  RM$RMS3                                   610  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       44         1        154         00:00.4




;                                    COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3MISC/OBJ=OBJ$:RM3MISC MSRC$:RM3MISC/UPDATE=(ENH$:RM3MISC)

; Size:          610 code + 0 data bytes
; Run Time:         00:15.0
; Elapsed Time:     00:42.0
; Lines/CPU Min:     3775
; Lexemes/CPU-Min: 13478
; Memory Used:  128 pages
; Compilation Complete
```

0325 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



RM3MAKIDX
LIS

RM3FNDRRV
LIS

RM3FNDRFA
LIS

RM3GET
LIS

RM3IUDR
LIS

RM3JOURNL
LIS

RM3KEYDSC
LIS

RM3MISC
LIS

RM3MISPUT
LIS