```
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    XX      XX  KK      KK  EEEEEEEEEE  YY      YY    000000
RRRRRRRR    MM      MM    333333    XX      XX  KK      KK  EEEEEEEEEE  YY      YY    000000
RR      RR  MMMM  MMMM  33      33  XX      XX  KK      KK  EE          YY      YY  00      00
RR      RR  MMMM  MMMM  33      33  XX      XX  KK      KK  EE          YY      YY  00      00
RR      RR  MM  MM  MM          33    XX  XX    KK    KK    EE            YY  YY    00      00
RR      RR  MM  MM  MM          33    XX  XX    KK    KK    EE            YY  YY    00      00
RRRRRRRR    MM      MM        33        XX      KKKKKK      EEEEEEEE        YY      00      00
RRRRRRRR    MM      MM        33        XX      KKKKKK      EEEEEEEE        YY      00      00
RR  RR      MM      MM          33    XX  XX    KK    KK    EE              YY      00      00
RR  RR      MM      MM          33    XX  XX    KK    KK    EE              YY      00      00
RR    RR    MM      MM  33      33  XX      XX  KK      KK  EE              YY      00      00      ....
RR    RR    MM      MM  33      33  XX      XX  KK      KK  EE              YY      00      00      ....
RR      RR  MM      MM    333333    XX      XX  KK      KK  EEEEEEEEEE      YY        000000        ....
RR      RR  MM      MM    333333    XX      XX  KK      KK  EEEEEEEEEE      YY        000000        ....

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 RM3XKEYO (LANGUAGE (BLISS32) ,
;     2          0002  0                   IDENT = 'V04-000'
;     3          0003  0                   ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !**************************************************************************
;     7          0007  1 !*                                                                        *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                              *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                  *
;    11          0011  1 !*                                                                        *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
;    17          0017  1 !*  TRANSFERRED.                                                           *
;    18          0018  1 !*                                                                        *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
;    21          0021  1 !*  CORPORATION.                                                           *
;    22          0022  1 !*                                                                        *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    25          0025  1 !*                                                                        *
;    26          0026  1 !*                                                                        *
;    27          0027  1 !**************************************************************************
;    28          0028  1
;    29          0029  1 !++
;    30          0030  1 !
;    31          0031  1 ! FACILITY:     RMS32 index sequential file organization
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !                This routine fills in the KEY XAB from the disk
;    35          0035  1 !
;    36          0036  1 !
;    37          0037  1 ! ENVIRONMENT:
;    38          0038  1 !
;    39          0039  1 !               VAX/VMS operating system
;    40          0040  1 !
;    41          0041  1 !--
;    42          0042  1
;    43          0043  1 !
;    44          0044  1 ! AUTHOR:         D. M. BOUSQUET
;    45          0045  1 ! CREATION DATE: 18-AUG-78  14:19
;    46          0046  1 !
;    47          0047  1 ! MODIFIED BY:
;    48          0048  1 !
;    49          0049  1 !       V03-008 DAS0001         David Solomon           25-Mar-1984
;    50          0050  1 !               Fix broken branches.
;    51          0051  1 !
;    52          0052  1 !       V03-007 LJA0099         Laurie J. Anderson      26-Sep-1983
;    53          0053  1 !               Fix bugcheck, where someone will do a $DISPLAY on a
;    54          0054  1 !               file opened with BRO.  In this case, the index descriptors
;    55          0055  1 !               are not allocated.
;    56          0056  1 !
;    57          0057  1 !       V03-006 MCN0002         Maria del C. Nasr       31-Mar-1983
```

```
;    58          0058  1 !               More linkages reorganization.
;    59          0059  1 !
;    60          0060  1 !       V03-005 MCN0001         Maria del C. Nasr       01-Mar-1983
;    61          0061  1 !               Reorganize linkages
;    62          0062  1 !
;    63          0063  1 !       V03-004 KBT0289         Keith B. Thompson       23-Aug-1982
;    64          0064  1 !               Reorganize psects
;    65          0065  1 !
;    66          0066  1 !       V03-003 KBT0074         Keith B. Thompson       29-Jun-1982
;    67          0067  1 !               Enable the processing of no-contiguous key descriptors
;    68          0068  1 !
;    69          0069  1 !       V03-002 KBT0017         Keith Thompson          19-Mar-1982
;    70          0070  1 !               Ignore compression bits when filling in xab of a prologue
;    71          0071  1 !               1,2 file
;    72          0072  1 !
;    73          0073  1 !       V03-001 JWH0001         Jeffrey W. Horn         16-March-1982
;    74          0074  1 !               Fix writing into space beyond the old end of XABKEY.
;    75          0075  1 !
;    76          0076  1 !       V02-006 DJD0001         Darrell Duffy           1-March-1982
;    77          0077  1 !               Fix probing in RM$XKEY03.
;    78          0078  1 !
;    79          0079  1 !       V02-005 KBT0004         K B Thompson            8-Feb-1982
;    80          0080  1 !               Correct compression bits when filling in key xab from
;    81          0081  1 !               prologue and stuff the prologue version if long key=0 xab
;    82          0082  1 !
;    83          0083  1 !       V02-004 CDS0001         C Saether               9-Aug-1981
;    84          0084  1 !               Use alternate linkage declaration for RELEASE.
;    85          0085  1 !
;    86          0086  1 !       V02-003 REFORMAT        D M WALP                24-JUL-1980
;    87          0087  1 !
;    88          0088  1 !
;    89          0089  1 ! REVISION HISTORY:
;    90          0090  1 !
;    91          0091  1 !   Wendy Koenig,       24-OCT-78  14:03
;    92          0092  1 !   X0002 - make changes caused by sharing conventions
;    93          0093  1 !
;    94          0094  1 !*****
;    95          0095  1
;    96          0096  1 LIBRARY 'RMSLIB:RMS';
;    97          0097  1 REQUIRE 'RMSSRC:RMSIDXDEF';
;    98          0162  1
;    99          0163  1 ! Define default psects for code
;   100          0164  1 !
;   101          0165  1 PSECT
;   102          0166  1     CODE = RM$RMS3(PSECT_ATTR),
;   103          0167  1     PLIT = RM$RMS3(PSECT_ATTR);
;   104          0168  1
;   105          0169  1
;   106          0170  1 LINKAGE
;   107          0171  1     L_CACHE,
;   108          0172  1     L_CHKSUM,
;   109          0173  1     L_FABREG_7,
;   110          0174  1     L_LINK_7_10_11,
;   111          0175  1     L_RELEASE_FAB;
;   112          0176  1
;   113          0177  1 ! External Routines
;   114          0178  1 !
```

```
;  115          0179  1 EXTERNAL ROUTINE
;  116          0180  1     RM$CACHE            : RL$CACHE ADDRESSING_MODE( LONG_RELATIVE ),
;  117          0181  1     RM$CHKSUM           : RL$CHKSUM,
;  118          0182  1     PM$GET_NEXT_KEY     : RL$LINK_7_10_11,
;  119          0183  1     RM$RELEASE          : RL$RELEASE_FAB ADDRESSING_MODE( LONG_RELATIVE );
;  120          0184  1
```

```
;  122          0185  1 %SBTTL  'RM$XKEYO3'
;  123          0186  1 GLOBAL ROUTINE RM$XKEYO3 ( XA3 : REF BBLOCK ) : RL$FABREG_7 =
;  124          0187  1
;  125          0188  1 !++
;  126          0189  1 !
;  127          0190  1 ! FUNCTIONAL DESCRIPTION:
;  128          0191  1 !
;  129          0192  1 !       The KEY XAB is filled in from the KEY descriptor
;  130          0193  1 !
;  131          0194  1 ! CALLING SEQUENCE:
;  132          0195  1 !
;  133          0196  1 !       RM$XKEYO3 ( XAB )
;  134          0197  1 !
;  135          0198  1 ! INPUT PARAMETERS:
;  136          0199  1 !
;  137          0200  1 !       XAB             -  Pointer to the KEY XAB we are filling in
;  138          0201  1 !
;  139          0202  1 ! IMPLICIT INPUTS:
;  140          0203  1 !
;  141          0204  1 !       $XAB
;  142          0205  1 !           [REF]               -  Key of reference
;  143          0206  1 !           [KNM]               -  Pointer to key name buffer
;  144          0207  1 !
;  145          0208  1 !       $KEYDEF
;  146          0209  1 !           [KEY$S_KEYNAM]      -  Size of the keyname buffer, 32
;  147          0210  1 !           [KEY$T_KEYNAM]      -  Buffer of the keyname (table)
;  148          0211  1 !
;  149          0212  1 !       $IFAB
;  150          0213  1 !           [IFB$B_ORGCASE]     -  File organization
;  151          0214  1 !           [IFB$C_IDX]         -  Indexed file organization constant
;  152          0215  1 !           [IFB$B_MODE]        -  Mode of the key name buffer
;  153          0216  1 !           [IFB$B_PLG_VER]     -  Index file prologue version
;  154          0217  1 !
;  155          0218  1 !       $IDX_DFN
;  156          0219  1 !           [all fields in the index descriptor are input,
;  157          0220  1 !               but specifically reference a few]
;  158          0221  1 !           [IANUM]             -  Index area number
;  159          0222  1 !           [SEGMENTS]          -  Number of segments
;  160          0223  1 !           [FIXED_BLN]         -  Fixed block length analogous to the XAB
;  161          0224  1 !           [POSITION]          -  Position of first segment
;  162          0225  1 !           [SIZE]              -  Size of first segment
;  163          0226  1 !
;  164          0227  1 ! OUTPUT PARAMETERS:
;  165          0228  1 !
;  166          0229  1 !       XAB                     -  Untouched by this routine
;  167          0230  1 !
;  168          0231  1 ! IMPLICIT OUTPUTS:
;  169          0232  1 !
;  170          0233  1 !       $XAB
;  171          0234  1 !           [all fields are filled in from the index descriptor]
;  172          0235  1 !
;  173          0236  1 ! ROUTINE VALUE:
;  174          0237  1 !
;  175          0238  1 !       RMSERR
;  176          0239  1 !           (SUC)  - Success code
;  177          0240  1 !
;  178          0241  1 ! SIDE EFFECTS:
```

```
;  179     0242  1 !       none
;  180     0243  1 !
;  181     0244  1 !--
;  182     0245  1
;  183     0246  2     BEGIN
;  184     0247  2
;  185     0248  2     EXTERNAL REGISTER
;  186     0249  2         R_IDX_DFN_STR,
;  187     0250  2         COMMON_FAB_STR;
;  188     0251  2
;  189     0252  2     LOCAL
;  190     0253  2         SAV_BDB,
;  191     0254  2         KEY_DESC        : REF BBLOCK;
;  192     0255  2
;  193     0256  2     ! If BIO was set in the FAB, then we only want to make sure user knows that
;  194     0257  2     ! the NUM_KEYS is not filled in
;  195     0258  2     !
;  196     0259  2     IF .IFAB [ IFB$B_NUM_KEYS ] EQL 0
;  197     0260  2     THEN
;  198     0261  2         RETURN RMSSUC( OK_NOP );
;  199     0262  2
;  200     0263  2     ! Let's first check to see if this is a indexed file
;  201     0264  2     !
;  202     0265  2     IF .IFAB [ IFB$B_ORGCASE ] EQL IFB$C_IDX
;  203     0266  2     THEN
;  204     0267  3         BEGIN
;  205     0268  3
;  206     0269  3         ! Now to make sure that the reference input in valid find the internal
;  207     0270  3         ! index descriptor
;  208     0271  3         !
;  209     0272  3         ! Start with the primary descriptor.  NOTE: We cannot call rm$key_desc
;  210     0273  3         ! here because of register problems but we do know that the first
;  211     0274  3         ! index descriptor off the ifab is the one for key 0 so get it that way
;  212     0275  3         !
;  213     0276  3         IDX_DFN = .IFAB [ IFB$L_IDX_PTR ];
;  214     0277  3
;  215     0278  3         ! Make sure that the index descriptors have been allocated before
;  216     0279  3         ! you use it.  Return success if it's not there, that's okay.
;  217     0280  3         IF .IDX_DFN EQLU 0
;  218     0281  3         THEN
;  219     0282  3             RETURN RMSSUC( SUC );
;  220     0283  3
;  221     0284  3         ! Loop until we find it
;  222     0285  3         !
;  223     0286  3         WHILE .XAB [ XAB$B_REF ] NEQ .IDX_DFN [ IDX$B_KEYREF ]
;  224     0287  3         DO
;  225     0288  3
;  226     0289  3             ! If we ran out of keys there is a problem
;  227     0290  3             !
;  228     0291  3             IF NOT RMS$GET_NEXT_KEY()
;  229     0292  3             THEN
;  230     0293  3                 RETURN RMSERR( REF );
;  231     0294  3
;  232     0295  3         ! Now to read in prologue descriptor
;  233     0296  3         !
;  234     0297  4         BEGIN
;  235     0298  4         GLOBAL REGISTER
```

```
;  236     0299  4          COMMON_IO_STR;
;  237     0300  4
;  238     0301  4      LOCAL
;  239     0302  4          STATUS;
;  240     0303  4
;  241     0304  4      STATUS = RM$CACHE( .IDX_DFN [ IDX$L_VBN ], 512, 0);
;  242     0305  4
;  243     0306  4      ! If error then return with error code in status
;  244     0307  4      !
;  245     0308  4      IF NOT .STATUS
;  246     0309  4      THEN
;  247     0310  4          RETURN .STATUS;
;  248     0311  4
;  249     0312  4      ! Now to release the bucket and check it
;  250     0313  4      !
;  251   P 0314  4      RETURN_ON_ERROR( RM$CHKSUM(),
;  252   P 0315  4                       RM$RELEASE(0)
;  253     0316  4                       );
;  254     0317  4
;  255     0318  4      ! Now point to the key descriptor in the prologue
;  256     0319  4      !
;  257     0320  4      KEY_DESC = .BKT_ADDR + .IDX_DFN [ IDX$W_OFFSET ];
;  258     0321  4
;  259     0322  4      ! Now to save the BDB before CH$MOVE clobbers it
;  260     0323  4      !
;  261     0324  4      SAV_BDB = .BDB
;  262     0325  4
;  263     0326  3      END;
;  264     0327  3
;  265     0328  3      !       Now to do a straight move from the key descriptor to the XAB
;  266     0329  3      !
;  267     0330  3      CH$MOVE( $BYTEOFFSET( KEY$T_KEYNAM ) - $BYTEOFFSET ( KEY$B_IANUM ),
;  268     0331  3               KEY_DESC [ KEY$B_IANUM ],
;  269     0332  3               XAB [ XAB$B_IAN ]);
;  270     0333  3
;  271     0334  3      ! If this is a prologue 3 file correct the compression bits
;  272     0335  3      !
;  273     0336  3      IF .IFAB [ IFB$B_PLG_VER ] GEQU 3
;  274     0337  3      THEN
;  275     0338  3
;  276     0339  3          ! SET in the prologue = CLEAR in the xab
;  277     0340  3          !
;  278     0341  4          XAB [ XAB$B_FLG ] = .XAB [ XAB$B_FLG ] XOR ( XAB$M_IDX_NCMPR OR
;  279     0342  4                                                       XAB$M_KEY_NCMPR OR
;  280     0343  3                                                       XAB$M_DAT_NCMPR );
;  281     0344  3
;  282     0345  3      ! If this is a long key xab and it is key-0 (primary)
;  283     0346  3      ! then fill in the prologue version number
;  284     0347  3      !
;  285     0348  3      IF ( .XAB [ XAB$B_BLN ] EQLU XAB$K_KEYLEN ) AND
;  286     0349  4         ( .XAB [ XAB$B_REF ] EQLU 0 )
;  287     0350  3      THEN
;  288     0351  3          XAB [ XAB$B_PROLOG ] = .IFAB [ IFB$B_PLG_VER ];
;  289     0352  3
;  290     0353  3      ! If the user has a key name buffer fill it in
;  291     0354  3      !
;  292     0355  3      IF .XAB [ XAB$L_KNM ] NEQ 0
```

```
;  293          0356  3          THEN
;  294          0357  4              BEGIN
;  295          0358  4              LOCAL
;  296          0359  4                  KNM_ADDR;
;  297          0360  4
;  298          0361  4              KNM_ADDR = .XAB [ XAB$L_KNM ];
;  299          0362  4
;  300          0363  4              ! Probe it
;  301          0364  4              !
;  302        P 0365  4              IFNOWRT( %REF( KEY$S_KEYNAM ),.KNM_ADDR,IFAB [ IFB$B_MODE ],
;  303        P 0366  4                       BEGIN
;  304        P 0367  4                       GLOBAL REGISTER
;  305        P 0368  4                           R_BDB_STR;
;  306        P 0369  4
;  307        P 0370  4                       BDB = .SAV_BDB;
;  308        P 0371  4                       RM$RELEASE(0);
;  309        P 0372  4                       RETURN RMSERR(KNM)
;  310          0373  4                       END);
;  311          0374  4
;  312          0375  4
;  313          0376  4              ! Now to move the buffer
;  314          0377  4              !
;  315          0378  4              CH$MOVE( KEY$S_KEYNAM,KEY_DESC [ KEY$T_KEYNAM ],.KNM_ADDR )
;  316          0379  4
;  317          0380  3              END;
;  318          0381  3
;  319          0382  3          ! Now move last long word.
;  320          0383  3          !
;  321          0384  3          XAB [ XAB$L_DVB ] = .KEY_DESC [ KEY$L_LDVBN ];
;  322          0385  3
;  323          0386  4          BEGIN
;  324          0387  4          GLOBAL REGISTER
;  325          0388  4              R_BDB_STR;
;  326          0389  4
;  327          0390  4              BDB = .SAV_BDB;
;  328          0391  4              RM$RELEASE(0)
;  329          0392  4              END
;  330          0393  4
;  331          0394  2              END;
;  332          0395  2
;  333          0396  2          ! Now to return the success code if all went well
;  334          0397  2          !
;  335          0398  3          RETURN RMSSUC( SUC )
;  336          0399  3
;  337          0400  1          END;


                                                    .TITLE  RM3XKEY0
                                                    .IDENT  \V04-000\

                                                    .EXTRN  RM$CACHE, RM$CHKSUM
                                                    .EXTRN  RM$GET_NEXT_KEY
                                                    .EXTRN  RM$RELEASE

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

                                007C    8F  BB 00000 RM$XKEY03::
```

```
                                                    PUSHR   #^M<R2,R3,R4,R5,R6>                ; 0186
                5E            08  C2 00004          SUBL2   #8, SP                             ;
                      00B2    CA  95 00007          TSTB    178(IFAB)                          ; 0259
                              07  12 0000B          BNEQ    1$                                 ;
                50    8059    8F  3C 0000D          MOVZWL  #32857, R0                         ; 0261
                              53  11 00012          BRB     6$                                 ;
                02      23    AA  91 00014 1$:      CMPB    35(IFAB), #2                       ; 0265
                              03  13 00018          BEQL    3$                                 ;
                            00C2  31 0001A 2$:      BRW     13$                                ;
                57    00AC    CA  D0 0001C 3$:      MOVL    172(IFAB), IDX_DFN                 ; 0276
                              F6  13 00022          BEQL    2$                                 ; 0280
                56      20    AE  D0 00024          MOVL    XAB, R6                            ; 0286
           21   A7      17    A6  91 00028 4$:      CMPB    23(R6), 33(IDX_DFN)                ;
                              0D  13 0002D          BEQL    5$                                 ;
                           0000G  30 0002F          BSBW    RM$GET_NEXT_KEY                    ; 0291
                F3            50  E8 00032          BLBS    R0, 4$                             ;
                50    875C    8F  3C 00035          MOVZWL  #34652, R0                         ; 0293
                              2B  11 0003A          BRB     6$                                 ;
                              53  D4 0003C 5$:      CLRL    R3                                 ; 0304
                52    0200    8F  3C 0003E          MOVZWL  #512, R2                           ;
                51      0A    A7  D0 00043          MOVL    10(IDX_DFN), R1                    ;
                   00000000G  EF  16 00047          JSB     RM$CACHE                           ;
                6D            50  E9 0004D          BLBC    STATUS, 10$                        ; 0308
                           0000G  30 00050          BSBW    RM$CHKSUM                          ; 0316
           04   AE            50  D0 00053          MOVL    R0, STATUS                         ;
                0E      04    AE  E8 00057          BLBS    STATUS, 7$                         ;
                              53  D4 0005B          CLRL    R3                                 ;
                   00000000G  EF  16 0005D          JSB     RM$RELEASE                         ;
                50      04    AE  D0 00063          MOVL    STATUS, R0                         ;
                              79  11 00067 6$:      BRB     14$                                ;
                50      0E    A7  3C 00069 7$:      MOVZWL  14(IDX_DFN), R0                    ; 0320
      6E        55            50  C1 0006D          ADDL3   R0, BKT_ADDR, KEY_DESC             ;
           04   AE            54  D0 00071          MOVL    BDB, SAV_BDB                       ; 0324
      7E        6E            06  C1 00075          ADDL3   #6, KEY_DESC, -(SP)                ; 0332
 08   A6        9E            2E  28 00079          MOVC3   #46, @(SP)+, 8(R6)                 ;
                03    00B7    CA  91 0007E          CMPB    183(IFAB), #3                      ; 0336
                              05  1F 00083          BLSSU   8$                                 ;
           12   A6      C8    8F  8C 00085          XORB2   #200, 18(R6)                       ; 0341
           4C   8F      01    A6  91 0008A 8$:      CMPB    1(R6), #76                         ; 0348
                              0B  12 0008F          BNEQ    9$                                 ;
                        17    A6  95 00091          TSTB    23(R6)                             ; 0349
                              06  12 00094          BNEQ    9$                                 ;
           48   A6    00B7    CA  90 00096          MOVB    183(IFAB), 72(R6)                  ; 0351
                        38    A6  D5 0009C 9$:      TSTL    56(R6)                             ; 0355
                              26  13 0009F          BEQL    12$                                ;
                55      38    A6  D0 000A1          MOVL    56(R6), KNM_ADDR                   ; 0361
      65        20      0A    AA  0D 000A5          PROBEW  10(IFAB), #32, (KNM_ADDR)          ; 0373
                              13  12 000AA          BNEQ    11$                                ;
                54      04    AE  D0 000AC          MOVL    SAV_BDB, BDB                       ;
                              53  D4 000B0          CLRL    R3                                 ;
                   00000000G  EF  16 000B2          JSB     RM$RELEASE                         ;
                50    8774    8F  3C 000B8          MOVZWL  #34676, R0                         ;
                              23  11 000BD 10$:     BRB     14$                                ;
      7E        6E            34  C1 000BF 11$:     ADDL3   #52, KEY_DESC, -(SP)               ; 0378
      65        9E            20  28 000C3          MOVC3   #32, @(SP)+, (KNM_ADDR)            ;
      50        6E 00000054   8F  C1 000C7 12$:     ADDL3   #84, KEY_DESC, R0                  ; 0384
           3C   A6            60  D0 000CF          MOVL    (R0), 60(R6)                       ;
```

```
                          54        04    AE  D0 000D3           MOVL    SAV_BDB, BDB                            ; 0390
                                          53  D4 000D7           CLRL    R3                                      ; 0391
                             00000000G    EF  16 000D9           JSB     RM$RELEASE                              ;
                          50              01  D0 000DF 13$:      MOVL    #1, R0                                  ; 0398
                          5E              08  C0 000E2 14$:      ADDL2   #8, SP                                  ; 0400
                                  007C    8F  BA 000E5           POPR    #^M<R2,R3,R4,R5,R6>                     ;
                                              05 000E9           RSB                                             ;

; Routine Size:  234 bytes,    Routine Base:  RM$RMS3 + 0000


;  338           0401  1
;  339           0402  1 END
;  340           0403  1
;  341           0404  0 ELUDOM




;                                 PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  RM$RMS3                       234  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        58         1       154         00:00.4



;                                    COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3XKEYO/OBJ=OBJ$:RM3XKEYO MSRC$:RM3XKEYO/UPDATE=(ENH$:RM3XKEYO)

; Size:         234 code + 0 data bytes
; Run Time:        00:07.0
; Elapsed Time:    00:16.7
; Lines/CPU Min:    3472
; Lexemes/CPU-Min: 15292
; Memory Used:  98 pages
; Compilation Complete
```

RMS0FSCN
LIS

RM3XALLO
LIS

RMS0CONN
LIS

RMS0ENTER
LIS

RMS0ERASE
LIS

RMS0CREAT
LIS

RMS0DELET
LIS

RMS0FIND
LIS

RM3XSUMO
LIS

RMS0DISPL
LIS

RM3XKEYO
LIS

RMS0GET
LIS

RMS0CLOSE
LIS

RMS0DISC
LIS

RMS0EXTEN
LIS

RMS0BLKIO
LIS