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

```
RRRRRRRR    MM       MM    333333      CCCCCCCC    000000    NN       NN  NN       NN
RRRRRRRR    MM       MM    333333      CCCCCCCC    000000    NN       NN  NN       NN
RR      RR  MMMM   MMMM  33      33  CC          00      00  NN       NN  NN       NN
RR      RR  MMMM   MMMM  33      33  CC          00      00  NN       NN  NN       NN
RR      RR  MM MM MM MM        33  CC          00      00  NNNN     NN  NNNN     NN
RR      RR  MM  MMM  MM        33  CC          00      00  NNNN     NN  NNNN     NN
RRRRRRRR    MM       MM      33    CC          00      00  NN NN    NN  NN NN    NN
RRRRRRRR    MM       MM      33    CC          00      00  NN NN    NN  NN NN    NN
RR  RR      MM       MM        33  CC          00      00  NN  NNNN NN      NNNN
RR  RR      MM       MM        33  CC          00      00  NN  NNNN NN      NNNN
RR    RR    MM       MM  33    33  CC          00      00  NN    NN NN        NN    ....
RR    RR    MM       MM  33    33  CC          00      00  NN    NN NN        NN    ....
RR      RR  MM       MM    333333      CCCCCCCC    000000    NN       NN  NN       NN    ....
RR      RR  MM       MM    333333      CCCCCCCC    000000    NN       NN  NN       NN    ....

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 RM3CONN (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 !                INDEXED SPECIFIC CODE FOR $CONNECT
;    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:       Wendy Koenig     CREATION DATE:         6-APR-78  10:11
;    45          0045  1 !
;    46          0046  1 !
;    47          0047  1 ! MODIFIED BY:
;    48          0048  1 !
;    49          0049  1 !       V03-017 JWT0187         Jim Teague              19-Jul-1984
;    50          0050  1 !               Make more appropriate error exit when user attempts
;    51          0051  1 !               to connect with EOF and BIO.
;    52          0052  1 !
;    53          0053  1 !       V03-016 DAS0001         David Solomon           25-Mar-1984
;    54          0054  1 !               Fix broken branches.
;    55          0055  1 !
;    56          0056  1 !       V03-015 TMK0006         Todd M. Katz            02-Apr-1983
;    57          0057  1 !               When the new BDB stuff was implemented (RAS0134), the
```

```
;    58          0058  1 !               implementer neglected to put the addresses of the AI and BI
;    59          0059  1 !               journalling BDBs within the data BDB. This corrects the problem.
;    60          0060  1 !               The fields within the BDBs were also being set up incorrectly
;    61          0061  1 !               because the constant RJR$C_BLKLEN was being used as the size of
;    62          0062  1 !               the RMS Journalling record overhead instead of RJR$C_BKTLEN.
;    63          0063  1 !
;    64          0064  1 !       V03-014 MCN0013         Maria del C. Nasr       24-Mar-1983
;    65          0065  1 !               Reorganize GETSPC and RETSPC linkages
;    66          0066  1 !
;    67          0067  1 !       V03-013 RAS0134         Ron Schaefer            16-Mar-1983
;    68          0068  1 !               Implement an alternate BDB strategy for ISAM AI/BI journaling.
;    69          0069  1 !               The technique is to allocate one huge area for each
;    70          0070  1 !               BDB.  This area has room for the data buffer, a BDB and RJR
;    71          0071  1 !               for AI journaling; a data buffer, BDB and RJR for BI
;    72          0072  1 !               journaling.  The normal BDB points to the AI buffer.
;    73          0073  1 !               The AI and BI journaling BDBs are not allocated on the
;    74          0074  1 !               normal BDB chain but instead are only found from the
;    75          0075  1 !               data bucket BDB.
;    76          0076  1 !
;    77          0077  1 !       V03-012 MCN0012         Maria del C. Nasr       24-Feb-1983
;    78          0078  1 !               Reorganize linkages
;    79          0079  1 !
;    80          0080  1 !       V03-011 TMK0005         Todd M. Katz            22-Dec-1982
;    81          0081  1 !               If the file allows updates, and is either a prologue 3 file or
;    82          0082  1 !               defines alternate keys, then there is a new record buffer which
;    83          0083  1 !               must be allocatec  The address of this buffer is stored in
;    84          0084  1 !               IRB$L_OLDBUF, and  t is the same size as the internal record
;    85          0085  1 !               buffer. I have increased the size of the record buffers by two
;    86          0086  1 !               in some cases, to allow for both the size of the record and the
;    87          0087  1 !               record itself to be stored side by side within the same buffer.
;    88          0088  1 !
;    89          0089  1 !       V03-010 TMK0004         Todd M. Katz            06-Dec-1982
;    90          0090  1 !               If the file is a prologue 1 or 2 file, contains alternate
;    91          0091  1 !               keys, and is open for write access then allocate a record
;    92          0092  1 !               buffer and store its address in IRB$L_RECBUF. This record
;    93          0093  1 !               buffer will be used to store the primary data record being
;    94          0094  1 !               deleted during a $DELETE. The use of this buffer has become
;    95          0095  1 !               necessary because of the changes made in the $DELETE algorithm.
;    96          0096  1 !               The user data record is now deleted before the alternate keys,
;    97          0097  1 !               instead of the other way around which is how it has been done in
;    98          0098  1 !               the past. Thus, the primary data record must be saved off to
;    99          0099  1 !               the side so that after the user data record has been deleted it
;   100          0100  1 !               is still available for the extraction of the alternate keys as
;   101          0101  1 !               part of their deletion. This record buffer is used to save the
;   102          0102  1 !               primary data record.
;   103          0103  1 !
;   104          0104  1 !       V03-009 TMK0003         Todd M. Katz            02-Dec-1982
;   105          0105  1 !               The number of keybuffers is now represented by the constant
;   106          0106  1 !               IFB$C_KBUFNUM.
;   107          0107  1 !
;   108          0108  1 !       V03-008 TMK0002         Todd M. Katz            24-Nov-1982
;   109          0109  1 !               Allocate a seventh keybuffer.
;   110          0110  1 !
;   111          0111  1 !       V03-007 KBT0336         Keith B. Thompson       10-Sep-1982
;   112          0112  1 !               Remove ref. to RM$GETSOSPC routine
;   113          0113  1 !
;   114          0114  1 !       V03-006 KBT0160         Keith B. Thompson       21-Aug-1982
```

```
;  115          0115  1 !          Reorganize psects
;  116          0116  1 !
;  117          0117  1 !      V03-005 TMK0001         Todd M. Katz            22-Jun-1982
;  118          0118  1 !              !mplement the RMS cluster solution for next record positioning.
;  119          0119  1 !              Instead of keeping the NRP context in system space, RMS now
;  120          0120  1 !              keeps it locally in the IRAB. As a result it no longer has
;  121          0121  1 !              to allocate space for a NRP cell, and so that maybe deleted
;  122          0122  1 !              from this routine. Also, allocate a sixth keybuffer. Finally,
;  123          0123  1 !              the function of the bit IRB$V_EOF has been replaced by the bit
;  124          0124  1 !              IRB$V_CON_EOF.
;  125          0125  1 !
;  126          0126  1 !      V03-004 KBT0055         Keith B. Thompson       8-Jun-1982
;  127          0127  1 !              No lr nger need to allocate any index descriptors on connect
;  128          0128  1 !
;  129          0129  1 !      V03-003 KPL003          P Lieberwirth           25-May-1982
;  130          0130  1 !              Fix bug intoduced by KPL0002.  Can't call RMS$DISCOMMON
;  131          0131  1 !              on errors allocating BDBs or global buffers.
;  132          0132  1 !
;  133          0133  1 !      V03-002 LJA0007         Laurie Anderson         25-Mar-1982
;  134          0134  1 !              Change KBUFSZ to reference a macro when computing buffer
;  135          0135  1 !              size and make IFB$B_KBUFSZ a word, now: IFB$W_KBUFSZ.
;  136          0136  1 !
;  137          0137  1 !      V03-001 KPL0002         P Lieberwirth           17-Mar-1982
;  138          0138  1 !              Don't allow both PUT to EOF and Block IO to be specified.
;  139          0139  1 !              Call RMS$DISCOMMON on this error and errors allocating BDBs.
;  140          0140  1 !              Add subtitle.
;  141          0141  1 !
;  142          0142  1 !      V02-020 KPL0001         P Lieberwirth            5-Nov-1981
;  143          0143  1 !              Add support for PUT to EOF.
;  144          0144  1 !
;  145          0145  1 !      V02-019 CDS0002         C Saether               29-Sep-1981
;  146          0146  1 !              Always allocate 5 keybuffers for all plgs.
;  147          0147  1 !
;  148          0148  1 !      V02-018 PSK0002         Paulina S. Knibbe       08-Aug-1981
;  149          0149  1 !              Remove support for truncated keys in INDEX on
;  150          0150  1 !              prologue three files.
;  151          0151  1 !
;  152          0152  1 !      V02-017 CDS0001         C Saether               06-Aug-1981
;  153          0153  1 !              Deallocate index descriptors if BRO create
;  154          0154  1 !              and BIO connect.
;  155          0155  1 !
;  156          0156  1 !      V02-016 MCN0011         Maria del C. Nasr       19-Jun-1981
;  157          0157  1 !              Add a fifth key buffer for prologue 3 files.
;  158          0158  1 !
;  159          0159  1 !      V02-015 PSK0001         Paulina S. Knibbe       12-Jun-1981
;  160          0160  1 !              Allocate a 'split context' block whenever we do
;  161          0161  1 !              a connect on a prologue three file.
;  162          0162  1 !
;  163          0163  1 !      V02-014 MCN0008         Maria del C. Nasr       15-May-1981
;  164          0164  1 !              Add a fourth key buffer for prologue 3 files.
;  165          0165  1 !
;  166          0166  1 !      V02-013 MCN0007         Maria del C. Nasr       20-Apr-1981
;  167          0167  1 !              Add code to allocate record output buffer for prologue 3
;  168          0168  1 !              files.
;  169          0169  1 !
;  170          0170  1 !      V02-012 REFORMAT        Ken Henderson           29-Jul-1980
;  171          0171  1 !              The code was reformatted.
```

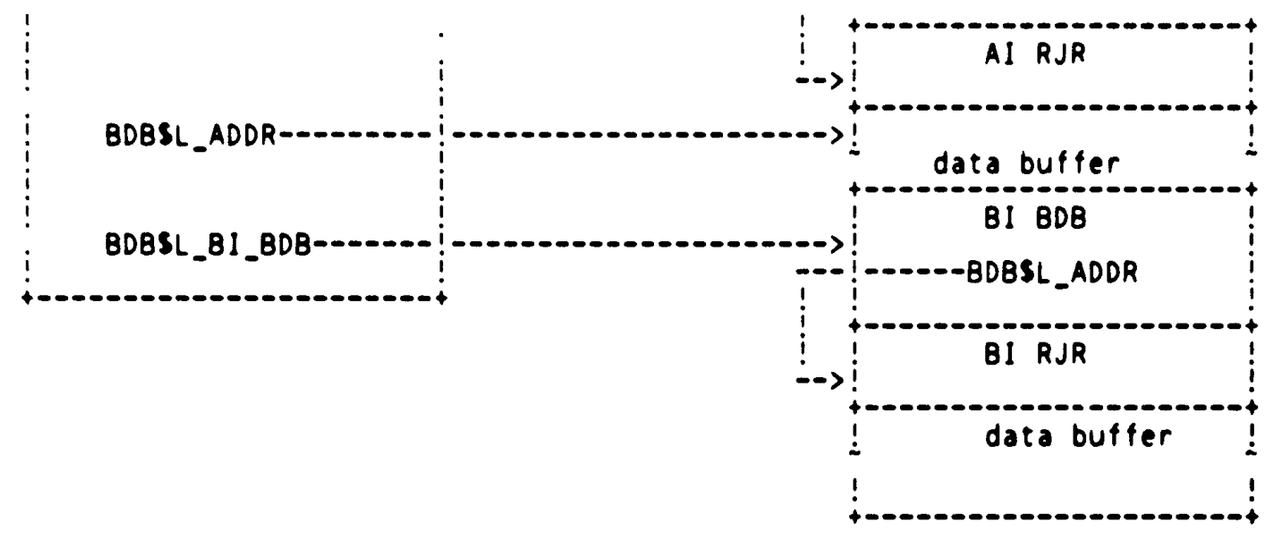
```
;  172     0172  1 !
;  173     0173  1 ! REVISION HISTORY:
;  174     0174  1 !
;  175     0175  1 !   Wendy Koenig,       24-JUL-78  13:34
;  176     0176  1 !   X0002 - allocate 3rd keybuffer to store primary key on put
;  177     0177  1 !
;  178     0178  1 !   Wendy Koenig,       24-OCT-78  14:01
;  179     0179  1 !   X0003 - make changes caused by sharing conventions
;  180     0180  1 !
;  181     0181  1 !   Wendy Koenig,       25-OCT-78  13:05
;  182     0182  1 !   X0004 - increment nrp_vbn in case it's bio
;  183     0183  1 !
;  184     0184  1 !   Wendy Koenig,       3-NOV-78  8:48
;  185     0185  1 !   X0005 - allocate update buffer for all keys,
;  186     0186  1 !        if bro turns to record i/o and put accessed, allocate the key desc
;  187     0187  1 !
;  188     0188  1 !   Wendy Koenig,       6-NOV-78  13:05
;  189     0189  1 !   X0006 - check for eof bit in rop, if found, give error
;  190     0190  1 !
;  191     0191  1 !   Wendy Koenig,       13-NOV-78  13:28
;  192     0192  1 !   X0007 - sharing implementation changes
;  193     0193  1 !
;  194     0194  1 !   Wendy Koenig,       20-NOV-78  10:54
;  195     0195  1 !   X0008 - allocate bcb's and other changes for sharing
;  196     0196  1 !
;  197     0197  1 !   Wendy Koenig,       28-NOV-78  15:22
;  198     0198  1 !   X0009 - bdballoc will now allocate the minimum # for isam, so don't call
;  199     0199  1 ! bdballoc_alt
;  200     0200  1 !
;  201     0201  1 !   Wendy Koenig,       22-JAN-79  13:05
;  202     0202  1 !   X0010 - allocate a lock bdb per stream, not per file ( needed for update
;  203     0203  1 ! RRV)
;  204     0204  1 !
;  205     0205  1 !   Christian Saether,  26-Oct-79  23:45
;  206     0206  1 !   0011 - rm$bdballoc now allocates lock bdb and does block i/o right
;  207     0207  1 !
;  208     0208  1 !*****
;  209     0209  1
;  210     0210  1 LIBRARY 'RMSLIB:RMS';
;  211     0211  1
;  212     0212  1 REQUIRE 'RMSSRC:RMSIDXDEF';
;  213     0277  1
;  214     0278  1 ! Define default psects for code
;  215     0279  1 !
;  216     0280  1 PSECT
;  217     0281  1     CODE = RM$RMS3(PSECT_ATTR),
;  218     0282  1     PLIT = RM$RMS3(PSECT_ATTR);
;  219     0283  1
;  220     0284  1 ! Linkages
;  221     0285  1 !
;  222     0286  1 LINKAGE
;  223     0287  1     L_BDBALLOC,
;  224     0288  1     L_ERROR_LINK1,
;  225     0289  1     L_GETSPC,
;  226     0290  1     L_LINK_7_10_11,
;  227     0291  1     L_RABREG,
;  228     0292  1     L_RABREG_7,
```

```
;  229          0293  1      L_RETSPC;
;  230          0294  1
;  231          0295  1  ! External Routines
;  232          0296  1  !
;  233          0297  1  EXTERNAL ROUTINE
;  234          0298  1      RM$BDBALLOC         : RL$BDBALLOC ADDRESSING_MODE( LONG_RELATIVE ),
;  235          0299  1      RM$CCLN1            : RL$ERROR_LINK1 ADDRESSING_MODE( LONG_RELATIVE ),
;  236          0300  1      RM$CLOSE3           : RL$LINK_7_10_11,
;  237          0301  1      RM$DISCOMMON        : RL$ERROR_LINK1 ADDRESSING_MODE( LONG_RELATIVE ),
;  238          0302  1      RM$KEY_DESC         : RL$RABREG_7,
;  239          0303  1      RM$RETSPC1          : RL$RETSPC_NOVALUE ADDRESSING_MODE( LONG_RELATIVE ),
;  240          0304  1      RM$GETSPC1          : RL$GETSPC ADDRESSING_MODE( LONG_RELATIVE );
;  241          0305  1
;  242          0306  1
```

```
;  244     0307  1 %SBTTL 'RM$CONNECT3B'
;  245     0308  1 GLOBAL ROUTINE RM$CONNECT3B : RL$RABREG =
;  246     0309  1
;  247     0310  1 !++
;  248     0311  1 !
;  249     0312  1 ! FUNCTIONAL DESCRIPTION:
;  250     0313  1 !
;  251     0314  1 !       This module performs the following functions required for
;  252     0315  1 !       connecting indexed files:
;  253     0316  1 !       1 -- performs various validity checks
;  254     0317  1 !       2 -- if connect for block i/o allocate a BDB and exit after
;  255     0318  1 !            first deallocating index descriptors if BRO create
;  256     0319  1 !       3 -- allocate BDB's and buffer's depending on access mode
;  257     0320  1 !            -- if write accessed allocate a lock BDB
;  258     0321  1 !            -- and if shared, allocate a BCB
;  259     0322  1 !       4 -- allocate key buffers and an update buffer
;  260     0323  1 !            -- allocates 7 key buffers of keybufsize
;  261     0324  1 !       5 -- allocate record output buffer
;  262     0325  1 !       6 -- allocate old record buffer if update accesses allowed
;  263     0326  1 !       7 -- initialize various fields in the IRAB
;  264     0327  1 !       8 -- initialize the next record positioning information kept within
;  265     0328  1 !            the IRAB
;  266     0329  1 !
;  267     0330  1 ! CALLING SEQUENCE:
;  268     0331  1 !
;  269     0332  1 !       BSBW RM$CONNECT3B
;  270     0333  1 !       Entered via case branch from RMS$CONNECT and BSBW from RM$CONNECT3;
;  271     0334  1 !       returns to RM$CONNECT3 and then exits from RMS directly
;  272     0335  1 !       (i.e  - on success, it branches to RM$EXSUC
;  273     0336  1 !             - on error, it calls either RM$CCLN1 to deallocate the IRAB or
;  274     0337  1 !               RM$COMCLNUP to deallocate the BDB's, buffer's, BCB's, and the
;  275     0338  1 !               IRAB, and then branches to RM$EX_NOSTR)
;  276     0339  1 !
;  277     0340  1 ! INPUT PARAMETERS:
;  278     0341  1 !       none
;  279     0342  1 !
;  280     0343  1 ! IMPLICIT INPUTS:
;  281     0344  1 !
;  282     0345  1 !       R8 -- RAB address
;  283     0346  1 !       R9 -- IRAB address
;  284     0347  1 !       R10 -- IFAB address
;  285     0348  1 !       R11 -- IMPURE AREA address
;  286     0349  1 !       contents of RAB and IRAB
;  287     0350  1 !
;  288     0351  1 ! OUTPUT PARAMETERS:
;  289     0352  1 !       none
;  290     0353  1 !
;  291     0354  1 ! IMPLICIT OUTPUTS:
;  292     0355  1 !
;  293     0356  1 !       Set various fields in IRAB and IFAB
;  294     0357  1 !
;  295     0358  1 ! ROUTINE VALUE:
;  296     0359  1 !
;  297     0360  1 !       Usual RMS status codes
;  298     0361  1 !
;  299     0362  1 ! SIDE EFFECTS:
;  300     0363  1 !
```

```
;  301          0364  1 !       Allocate BDB's and buffer's, key buffers and update buffer, record
;  302          0365  1 !       output buffer and old record buffer, and zero out all next record
;  303          0366  1 !       positioning context variables kept within the  IRAB except for the
;  304          0367  1 !       current key of reference which is initialized to that specified
;  305          0368  1 !       by the user in the RAB.
;  306          0369  1 !
;  307          0370  1 !--
;  308          0371  1
;  309          0372  2     BEGIN
;  310          0373  2
;  311          0374  2     MACRO
;  312        M 0375  2         DEALLOC_KEYBUF =
;  313        M 0376  2             BEGIN
;  314        M 0377  2
;  315        M 0378  2             SIZE = .IFAB[IFB$W_KBUFSZ]*IFB$C_KBUFNUM;
;  316        M 0379  2
;  317        M 0380  2             IF .SIZE LSS 12
;  318        M 0381  2             THEN
;  319        M 0382  2                 SIZE = 12;
;  320        M 0383  2
;  321        M 0384  2             RMSRETSPC1 ( .SIZE, 0, KEYBUF_ADDR(1) );
;  322        M 0385  2             RMSDISCOMMON();
;  323        M 0386  2             RETURN .STATUS;
;  324          0387  2             END; %,
;  325          0388  2
;  326        M 0389  2         DEALLOC_UPDBUF =
;  327        M 0390  2             BEGIN
;  328        M 0391  2             IF .IFAB[IFB$V_UPD]
;  329        M 0392  2             THEN
;  330        M 0393  2                 BEGIN
;  331        M 0394  2                 SIZE = .IFAB[IFB$B_NUM_KEYS];
;  332        M 0395  2
;  333        M 0396  2                 IF .SIZE LSS 12
;  334        M 0397  2                 THEN
;  335        M 0398  2                     SIZE = 12;
;  336        M 0399  2
;  337        M 0400  2                 RMSRETSPC1 ( .SIZE, 0, .IRAB[IRB$L_UPDBUF] );
;  338        M 0401  2                 END;
;  339          0402  2             END; %;
;  340          0403  2
;  341          0404  2     BUILTIN
;  342          0405  2         TESTBITSC;
;  343          0406  2
;  344          0407  2     EXTERNAL REGISTER
;  345          0408  2         COMMON_RAB_STR;
;  346          0409  2
;  347          0410  2     GLOBAL REGISTER
;  348          0411  2         R_IDX_DFN_STR;
;  349          0412  2
;  350          0413  2     LOCAL
;  351          0414  2         SIZE;
;  352          0415  2
;  353          0416  2     ! Support PUT to EOF by translating ROP bit into IRAB bit for use on first
;  354          0417  2     ! PUT.
;  355          0418  2     !
;  356          0419  2     IF .RAB[RAB$V_EOF]
;  357          0420  2     THEN
```

```
;  358     0421  2
;  359     0422  2          !  Don't allow EOF and BIO to both be specified.
;  360     0423  2          !
;  361     0424  2          IF .RAB[RAB$V_BIO]
;  362     0425  2          THEN
;  363     0426  3              BEGIN
;  364     0427  3              RM$CCLN1();
;  365     0428  3              RETURN RMSERR(ROP);
;  366     0429  3              END
;  367     0430  2          ELSE
;  368     0431  2              IRAB[IRB$V_CON_EOF] = 1;
;  369     0432  2
;  370     0433  2
;  371     0434  2      ! In case we're in block i/o mode, have to do this -- same field as NRP_PTR.
;  372     0435  2      !
;  373     0436  2      IRAB[IRB$L_NRP_VBN] = .IRAB[IRB$L_NRP_VBN] + 1;
;  374     0437  2
;  375     0438  2      ! If $OPEN or $CREATE was done with bro specified, use the bio rop bit
;  376     0439  2      ! to determine whether to connect for block or record operations. note:
;  377     0440  2      ! any subsequent connects are necessarily of the same type.
;  378     0441  2      !
;  379     0442  2      IF TESTBITSC (IFAB[IFB$V_BRO])
;  380     0443  2      THEN
;  381     0444  2
;  382     0445  2          ! If block io deallocate the index descriptors and set the bio bit in
;  383     0446  2          ! the ifab to disallow record operations
;  384     0447  2          !
;  385     0448  2          IF .RAB[RAB$V_BIO]
;  386     0449  2          THEN
;  387     0450  3              BEGIN
;  388     0451  3              RM$CLOSE3();
;  389     0452  3              IFAB[IFB$V_BIO] = 1
;  390     0453  2              END;
;  391     0454  2
;  392     0455  2        IRAB[IRB$L_CURBDB] = 0;
;  393     0456  2
;  394     0457  2        ! Allocate all of the BDB's and buffer's and BCB's ( 1 BCB per BDB).
;  395     0458  2        ! RM$BDBALLOC handles all defaulting such that a minimum of 2 buffers
;  396     0459  2        ! are allocated plus a lock BDB (no buffer) if write accessed. For block
;  397     0460  2        ! i/o a BDB only (no buffer) is allocated. Exit after BDBALLOC if block
;  398     0461  2        ! i/o. Also, count all 'extra' buffers (number in excess of 2 buffers)
;  399     0462  2
;  400     0463  2  ! If either AI or BI journaling is enabled then a more complicated BDB
;  401     0464  2  ! and buffer strategy is used.   Basically the data and AI buffer is
;  402     0465  2  ! shared, there is a separate BI buffer and there are separate AI
;  403     0466  2  ! and BI BDBs and RJRs allocated out of this one huge area.
;  404     0467  2  ! The reason for the one huge area is to make sure that the RM0x
;  405     0468  2  ! level of the system sees only normal BDBs and buffers.
;  406     0469  2  ! The addressing of the other portions is as follows:
;  407     0470  2  !
;  408     0471  2  !
;  409     0472  2  !  ...    +-----------------------+                      +-----------------------+
;  410     0473  2  !         !       data BDB        !                      !        AI BDB         !
;  411     0474  2  !         !                       !                      !                       !
;  412     0475  2  !         !    BDB$L_AI_BDB-------!----------------      !                       !
;  413     0476  2  !         !                       !                ---   !------BDB$L_ADDR       !
;  414     0477  2  !         !                       !                  !   !                       !
```




```
;  415     0478  2 !
;  416     0479  2 !
;  417     0480  2 !
;  418     0481  2 !
;  419     0482  2 !       BDB$L_ADDR--------!---------------------->!
;  420     0483  2 !                         !                       !
;  421     0484  2 !                         !                       !
;  422     0485  2 !                         !                       !
;  423     0486  2 !       BDB$L_BI_BDB------!---------------------->!
;  424     0487  2 !                         !                       !
;  425     0488  2 !       +-----------------!-----+                 !
;  426     0489  2 !                         !                       !
;  427     0490  2 !                         !                       !
;  428     0491  2 !                         !                       !
;  429     0492  2 !                         !                       !
;  430     0493  2 !                         !                       !
;  431     0494  2 !                         !                       !
;  432     0495  2 !                         !                       !
;  433     0496  2 !                         !                       !
;  434     0497  2
;  435     0498  3     BEGIN
;  436     0499  3
;  437     0500  3     LOCAL
;  438     0501  3         BKS,                            ! max bucket size
;  439     0502  3         BUFSIZ,                         ! size of buffer to allocate
;  440     0503  3         JBDB:   REF BLOCK[,BYTE],       ! pointer to jnlbdb
;  441     0504  3         DBDB:   REF BLOCK[,BYTE];       ! pointer to data bdb
;  442     0505  3
;  443     0506  3     BUFSIZ = BKS = .IFAB[IFB$B_BKS] ^ 9;
;  444     0507  3
;  445     0508  3     IF .IFAB[IFB$V_AI]
;  446     0509  3     THEN
;  447     0510  3         BUFSIZ = .BUFSIZ + RJR$C_BKTLEN + BDB$C_BLN;
;  448     0511  3
;  449     0512  3     IF .IFAB[IFB$V_BI]
;  450     0513  3     THEN
;  451     0514  3         BUFSIZ = .BUFSIZ + .BKS + RJR$C_BKTLEN + BDB$C_BLN;
;  452     0515  3
;  453     0516  3     RETURN_ON_ERROR (RM$BDBALLOC (4, .BUFSIZ));
;  454     0517  3
;  455     0518  3     ! The new BDBs are allocated onto the end of the BDB chain
;  456     0519  3     ! so now go backwards over the BDBs, carving them up as appropriate
;  457     0520  3     ! into the journaling data structures as needed.
;  458     0521  3     !
;  459     0522  3     DBDB = .IFAB[IFB$L_BDB_BLNK];
;  460     0523  3
;  461     0524  3     INCR I FROM 1 TO .IRAB[IRB$B_BCNT] DO
;  462     0525  4         BEGIN
;  463     0526  4         DBDB[BDB$W_SIZE] = .BKS;
;  464     0527  4
;  465     0528  4         IF .IFAB[IFB$V_AI]
;  466     0529  4         THEN
;  467     0530  5             BEGIN
;  468     0531  5             JBDB = .DBDB[BDB$L_ADDR];
;  469     0532  5             DBDB[BDB$L_AI_BDB] = .JBDB;
;  470     0533  5
;  471     0534  5             JBDB[BDB$B_BID] = BDB$C_BID;
```

```
;  472          0535  5              JBDB[BDB$B_BLN] = BDB$C_BLN/4;
;  473          0536  5              JBDB[BDB$L_FLINK] = .JBDB;
;  474          0537  5              JBDB[BDB$L_BLINK] = .JBDB;
;  475          0538  5              JBDB[BDB$W_SIZE] = .BKS + RJR$C_BKTLEN;
;  476          0539  5              JBDB[BDB$L_ADDR] = .JBDB + BDB$C_BLN;
;  477          0540  5              DBDB[BDB$L_ADDR] = .JBDB + BDB$C_BLN + RJR$C_BKTLEN;
;  478          0541  4              END;
;  479          0542  4
;  480          0543  4          IF .IFAB[IFB$V_BI]
;  481          0544  4          THEN
;  482          0545  5              BEGIN
;  483          0546  5              JBDB = .DBDB[BDB$L_ADDR] + .BKS;
;  484          0547  5              DBDB[BDB$L_BI_BDB] = .JBDB;
;  485          0548  5
;  486          0549  5              JBDB[BDB$B_BID] = BDB$C_BID;
;  487          0550  5              JBDB[BDB$B_BLN] = BDB$C_BLN/4;
;  488          0551  5              JBDB[BDB$L_FLINK] = .JBDB;
;  489          0552  5              JBDB[BDB$L_BLINK] = .JBDB;
;  490          0553  5              JBDB[BDB$W_SIZE] = .BKS + RJR$C_BKTLEN;
;  491          0554  5              JBDB[BDB$L_ADDR] = .JBDB + BDB$C_BLN;
;  492          0555  4              END;
;  493          0556  4          DBDB = .DBDB[BDB$L_BLINK];
;  494          0557  3          END;
;  495          0558  3
;  496          0559  2      END;
;  497          0560  2
;  498          0561  2      ! If open for block io return
;  499          0562  2      !
;  500          0563  2      IF .IFAB[IFB$V_BIO]
;  501          0564  2      THEN
;  502          0565  2          RETURN RMSSUC();
;  503          0566  2
;  504          0567  2      IF .IRAB[IRB$B_BCNT] GTR 2
;  505          0568  2      THEN
;  506          0569  2          IFAB[IFB$B_EXTRABUF] = .IFAB[IFB$B_EXTRABUF] +
;  507          0570  2                                 (.IRAB[IRB$B_BCNT] - 2);
;  508          0571  2
;  509          0572  2      ! Set up the key descriptor for the primary key
;  510          0573  2      !
;  511          0574  2      RETURN_ON_ERROR (RM$KEY_DESC(0), RM$DISCOMMON());
;  512          0575  2
;  513          0576  2      ! The following code is made of the sections that allocate the remaining
;  514          0577  2      ! structures:
;  515          0578  2      !   - key buffers
;  516          0579  2      !   - update buffer, if update set
;  517          0580  2      !   - old record buffer if update set
;  518          0581  2      !   - record buffer
;  519          0582  2      ! In each case, if it fails to allocate any of the structures, it
;  520          0583  2      ! deallocates any of those already allocated, and returns.
;  521          0584  2      !
;  522          0585  3      BEGIN
;  523          0586  3
;  524          0587  3      LOCAL
;  525          0588  3          STATUS;
;  526          0589  3
;  527          0590  3      ! Allocate the keystring buffer (IFB$C_KBUFNUM * .IFB$W_KBUFSZ).
;  528          0591  3      !
```

```
;  529     0592  4     BEGIN
;  530     0593  4
;  531     0594  4     LOCAL
;  532     0595  4         BLK : REF BBLOCK;
;  533     0596  4
;  534     0597  4     SIZE = .IFAB[IFB$W_KBUFSZ] * IFB$C_KBUFNUM;
;  535     0598  4
;  536     0599  4     IF .SIZE LSS 12
;  537     0600  4     THEN
;  538     0601  4         SIZE = 12;
;  539     0602  4
;  540     0603  4     IF RM$GETSPC1( 0, .SIZE; BLK )
;  541     0604  4     THEN
;  542     0605  4         IRAB[IRB$L_KEYBUF] = .BLK
;  543     0606  4     ELSE
;  544     0607  4         RETURN RM$DISCOMMON();
;  545     0608  3     END;
;  546     0609  3
;  547     0610  3     ! If update is set, then allocate an update buffer.
;  548     0611  3     !
;  549     0612  3     IF .IFAB[IFB$V_UPD]
;  550     0613  3     THEN
;  551     0614  4         BEGIN
;  552     0615  4
;  553     0616  4         LOCAL
;  554     0617  4             BLK : REF BBLOCK;
;  555     0618  4
;  556     0619  4         SIZE = .IFAB[IFB$B_NUM_KEYS];
;  557     0620  4
;  558     0621  4         IF .SIZE LSS 12
;  559     0622  4         THEN
;  560     0623  4             SIZE = 12;
;  561     0624  4
;  562     0625  5         IF (STATUS = RM$GETSPC1 ( 0, .SIZE; BLK ) )
;  563     0626  4         THEN
;  564     0627  4             IRAB[IRB$L_UPDBUF] = .BLK
;  565     0628  4         ELSE
;  566     0629  4             DEALLOC_KEYBUF;
;  567     0630  3         END;                                ! end of update buffer allocation
;  568     0631  3
;  569     0632  3     ! Determine the number of bytes to allocate for a record buffer depending
;  570     0633  3     ! on maximum record size.
;  571     0634  3     !
;  572     0635  3     IF .IFAB[IFB$W_MRS] EQLU 0
;  573     0636  3     THEN
;  574     0637  3         SIZE = .IDX_DFN[IDX$B_DATBKTSZ] * 512
;  575     0638  3     ELSE
;  576     0639  3         IF .IFAB[IFB$B_PLG_VER] GEQU PLG$C_VER_3
;  577     0640  3         THEN
;  578     0641  3             SIZE = .IFAB[IFB$W_MRS] + IRC$C_KEYCMPOVH
;  579     0642  3                                     + IRC$C_DATCMPOVH
;  580     0643  3                                     + 2
;  581     0644  3         ELSE
;  582     0645  3             SIZE = .IFAB[IFB$W_MRS] + 2;
;  583     0646  3
;  584     0647  3     IF .SIZE LSS 12
;  585     0648  3     THEN
```

```
;  586     0649  3          SIZE = 12;
;  587     0650  3
;  588     0651  3      ! Allocate record output buffer for unpacking of prologue 3 data records.
;  589     0652  3      ! Also allocate a record buffer if the file is a prologue 1 or 2 file, is
;  590     0653  3      ! opened for write access, and defines secondary keys.
;  591     0654  3      !
;  592     0655  4      IF  (.IFAB[IFB$B_PLG_VER] EQLU PLG$C_VER_3)
;  593     0656  3           OR
;  594     0657  4           (.IFAB[IFB$B_NUM_KEYS] GTRU 1
;  595     0658  4                   AND
;  596     0659  4                  .IFAB[IFB$V_WRTACC])
;  597     0660  3      THEN
;  598     0661  4          BEGIN
;  599     0662  4
;  600     0663  4          LOCAL
;  601     0664  4              BLK : REF BBLOCK;
;  602     0665  4
;  603     0666  5          IF (STATUS = RMSGETSPC1 ( 0, .SIZE; BLK ) )
;  604     0667  4          THEN
;  605     0668  4              IRAB[IRB$L_RECBUF] = .BLK
;  606     0669  4
;  607     0670  4          ! Deallocate update buffer, and key buffer on errors.
;  608     0671  4          !
;  609     0672  4          ELSE
;  610     0673  5              BEGIN
;  611     0674  5              DEALLOC_UPDBUF;
;  612     0675  5              DEALLOC_KEYBUF;
;  613     0676  4              END;
;  614     0677  4
;  615     0678  4          ! If update access has been specified and either the file is a prologue
;  616     0679  4          ! 3 file or alternate keys have been defined, allocate the old record
;  617     0680  4          ! buffer. On errors, return all internal structures that have been
;  618     0681  4          ! allocated.
;  619     0682  4          !
;  620     0683  4          IF  .IFAB[IFB$V_UPD]
;  621     0684  4          THEN
;  622     0685  5              BEGIN
;  623     0686  5
;  624     0687  5              LOCAL
;  625     0688  5                  BLK : REF BBLOCK;
;  626     0689  5
;  627     0690  6              IF (STATUS = RMSGETSPC1 ( 0, .SIZE; BLK ) )
;  628     0691  5              THEN
;  629     0692  5                  IRAB[IRB$L_OLDBUF] = .BLK
;  630     0693  5
;  631     0694  5              ! Deallocate the record buffer, update buffer, and key buffer.
;  632     0695  5              !
;  633     0696  5              ELSE
;  634     0697  6                  BEGIN
;  635     0698  6                  RMSRETSPC1 ( .SIZE, 0, .IRAB[IRB$L_RECBUF] );
;  636     0699  6                  DEALLOC_UPDBUF;
;  637     0700  6                  DEALLOC_KEYBUF;
;  638     0701  5                  END;
;  639     0702  4              END;
;  640     0703  3          END;
;  641     0704  2      END;
;  642     0705  2
```

```
        ! The next record positioning information is kept in the IRAB. It was
        ! zeroed out when the IRAB was allocated; therefore, the first record
        ! retrieved sequentially will be the very first record in the designated
        ! key of reference.
        !
        ! Initialize the current key of reference to that which is specified by
        ! the user in the RAB.
        !
        IRAB[IRB$B_CUR_KREF] = .RAB[RAB$B_KRF];
        RETURN RMSSUC();

        END;                                    ! { end of routine }


                                                .TITLE  RM3CONN
                                                .IDENT  \V04-000\

                                                .EXTRN  RM$BDBALLOC, RM$CCLN1
                                                .EXTRN  RM$CLOSE3, RM$DISCOMMON
                                                .EXTRN  RM$KEY_DESC, RM$RETSPC1
                                                .EXTRN  RM$GETSPC1

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

                        00FC   8F  BB 00000 RM$CONNECT3B::
                                                PUSHR   #^M<R2,R3,R4,R5,R6,R7>          ; 0308
                   5E          04  C2 00004     SUBL2   #4, SP                          ;
                   18     05   A8  E9 00007     BLBC    5(RAB), 3$                      ; 0419
  0E         05    A8          03  E1 0000B     BBC     #3, 5(RAB), 2$                  ; 0424
                   00000000G   EF  16 00010     JSB     RM$CCLN1                        ; 0427
                   50   867C   8F  3C 00016     MOVZWL  #34428, R0                      ; 0428
                          0240 31 0001B 1$:     BRW     36$                             ;
             06    A9     80   8F  88 0001E 2$: BISB2   #128, 6(IRAB)                   ; 0431
                          40   A9  D6 00023 3$: INCL    64(IRAB)                        ; 0436
  0C         22    AA          06  E5 00026     BBCC    #6, 34(IFAB), 4$                ; 0442
  07         05    A8          03  E1 0002B     BBC     #3, 5(RAB), 4$                  ; 0448
                          0000G 30 00030        BSBW    RM$CLOSE3                       ; 0451
             22    AA          20  88 00033     BISB2   #32, 34(IFAB)                   ; 0452
                          20   A9  D4 00037 4$: CLRL    32(IRAB)                        ; 0455
                   50     5E   AA  9A 0003A     MOVZBL  94(IFAB), R0                    ; 0506
  6E               50          09  78 0003E     ASHL    #9, R0, BKS                     ;
                   55          6E  D0 00042     MOVL    BKS, BUFSIZ                     ;
  05       00A0    CA          03  E1 00045     BBC     #3, 160(IFAB), 5$               ; 0508
                   55   0094   C5  9E 0004B     MOVAB   148(R5), BUFSIZ                 ; 0510
  0B       00A0    CA          02  E1 00050 5$: BBC     #2, 160(IFAB), 6$               ; 0512
  50               6E 00000094 8F  C1 00056     ADDL3   #148, BKS, R0                   ; 0514
                   55          50  C0 0005E     ADDL2   R0, BUFSIZ                      ;
                   54          04  D0 00061 6$: MOVL    #4, R4                          ; 0516
                   00000000G   EF  16 00064     JSB     RM$BDBALLOC                     ;
                   AE          50  E9 0006A     BLBC    STATUS, 1$                      ;
                   51     44   AA  D0 0006D     MOVL    68(IFAB), DBDB                  ; 0522
                   52     54   A9  9A 00071     MOVZBL  84(IRAB), R2                    ; 0524
                               53  D4 00075     CLRL    I                               ; 0528
                               5D  11 00077     BRB     10$                             ;
             16    A1          6E  B0 00079 7$: MOVW    BKS, 22(DBDB)                   ; 0526
  27       00A0    CA          03  E1 0007D     BBC     #3, 160(IFAB), 8$               ; 0528
                   50     18   A1  D0 00083     MOVL    24(DBDB), JBDB                  ; 0531
```

```
                                34   A1              50   D0 00087          MOVL    JBDB, 52(DBDB)                          ; 0532
                                08   A0        140C  8F   B0 0008B          MOVW    #5132, 8(JBDB)                          ; 0534
                                60              50   D0 00091          MOVL    JBDB, (JBDB)                            ; 0536
                                04   A0         50   D0 00094          MOVL    JBDB, 4(JBDB)                           ; 0537
                          16    A0              6E   8F   A1 00098          ADDW3   #68, BKS, 22(JBDB)                      ; 0538
                                18   A0          50   A0   9E 0009F          MOVAB   80(R0), 24(JBDB)                        ; 0539
                                18   A1        0094   C0   9E 000A4          MOVAB   148(R0), 24(DBDB)                       ; 0540
                                22   00A0      CA     02   E1 000AA 8$:      BBC     #2, 160(IFAB), 9$                       ; 0543
                                50   18   A1         6E   C1 000B0          ADDL3   BKS, 24(DBDB), JBDB                     ; 0546
                                30   A1         50   D0 000B5          MOVL    JBDB, 48(DBDB)                          ; 0547
                                08   A0        140C  8F   B0 000B9          MOVW    #5132, 8(JBDB)                          ; 0549
                                60              50   D0 000BF          MOVL    JBDB, (JBDB)                            ; 0551
                                04   A0         50   D0 000C2          MOVL    JBDB, 4(JBDB)                           ; 0552
                          16    A0              6E   8F   A1 000C6          ADDW3   #68, BKS, 22(JBDB)                      ; 0553
                                18   A0          50   A0   9E 000CD          MOVAB   80(R0), 24(JBDB)                        ; 0554
                                51        04    A1   D0 000D2 9$:      MOVL    4(DBDB), DBDB                           ; 0556
                          9F                    53   52   F3 000D6 10$:     AOBLEQ  R2, I, 7$                       ; 0524
                          03    22   AA              05   E1 000DA          BBC     #5, 34(IFAB), 11$                       ; 0563
                                           0179   31   000DF          BRW     35$                                     ;
                                02              52   91 000E2 11$:     CMPB    R2, #2                                  ; 0567
                                                0E   1B 000E5          BLEQU   12$                                     ;
                                50        00B6  CA   9A 000E7          MOVZBL  182(IFAB), R0                           ; 0570
                                50              52   C0 000EC          ADDL2   R2, R0                                  ;
                          00B6  CA              50   02   83 000EF          SUBB3   #2, R0, 182(IFAB)                       ; 0569
                                                7E   D4 000F5 12$:     CLRL    -(SP)                                   ; 0574
                                           0000G  30   000F7          BSBW    RM$KEY_DESC                             ;
                                5E              04   C0 000FA          ADDL2   #4, SP                                  ;
                                24              50   E9 000FD          BLBC    STATUS, 14$                             ;
                                56        00B4  CA   3C 00100          MOVZWL  180(IFAB), SIZE                         ; 0597
                                56              06   C4 00105          MULL2   #6, SIZE                                ;
                                0C              56   D1 00108          CMPL    SIZE, #12                               ; 0599
                                                03   18 0010B          BGEQ    13$                                     ;
                                56              0C   D0 0010D          MOVL    #12, SIZE                               ; 0601
                                52              56   D0 00110 13$:     MOVL    SIZE, R2                                ; 0603
                                                51   D4 00113          CLRL    R1                                      ;
                             00000000G          EF   16 00115          JSB     RM$GETSPC1                              ;
                                06              50   E9 0011B          BLBC    R0, 14$                                 ;
                          60    A9              51   D0 0011E          MOVL    BLK, 96(IRAB)                           ; 0605
                                                09   11 00122          BRB     15$                                     ;
                             00000000G          EF   16 00124 14$:     JSB     RM$DISCOMMON                            ; 0607
                                           0131   31   0012A          BRW     36$                                     ;
37                        22    AA              03   E1 0012D 15$:     BBC     #3, 34(IFAB), 19$                       ; 0612
                                56        00B2  CA   9A 00132          MOVZBL  178(IFAB), SIZE                         ; 0619
                                0C              56   D1 00137          CMPL    SIZE, #12                               ; 0621
                                                03   18 0013A          BGEQ    16$                                     ;
                                56              0C   D0 0013C          MOVL    #12, SIZE                               ; 0623
                                52              56   D0 0013F 16$:     MOVL    SIZE, R2                                ; 0625
                                                51   D4 00142          CLRL    R1                                      ;
                             00000000G          EF   16 00144          JSB     RM$GETSPC1                              ;
                                6E              50   D0 0014A          MOVL    R0, STATUS                              ;
                                06              6E   E9 0014D          BLBC    STATUS, 17$                             ;
                          64    A9              51   D0 00150          MOVL    BLK, 100(IRAB)                          ; 0627
                                                13   11 00154          BRB     19$                                     ;
                                56        00B4  CA   3C 00156 17$:     MOVZWL  180(IFAB), SIZE                         ; 0628
                                56              06   C4 0015B          MULL2   #6, SIZE                                ;
                                0C              56   D1 0015E          CMPL    SIZE, #12                               ;
                                                03   19 00161          BLSS    18$                                     ;
```

```
                     00D5   31 00163       BRW     33$                       ;
                     00CF   31 00166 18$:  BRW     32$                       ;
           50      60   AA   3C 00169 19$:  MOVZWL  96(IFAB), R0              ; 0635
                     0A   12 0016D       BNEQ    20$                       ;
           56      17   A7   9A 0016F       MOVZBL  23(IDX_DFN), SIZE         ; 0637
  56       56           09   78 00173       ASHL    #9, SIZE, SIZE            ;
                     11   11 00177       BRB     22$                       ;
           03    00B7   CA   91 00179 20$:  CMPB    183(IFAB), #3             ; 0639
                     06   1F 0017E       BLSSU   21$                       ;
           56      07   A0   9E 00180       MOVAB   7(R0), SIZE               ; 0643
                     04   11 00184       BRB     22$                       ; 0641
           56      02   A0   9E 00186 21$:  MOVAB   2(R0), SIZE               ; 0645
           0C           56   D1 0018A 22$:  CMPL    SIZE, #12                 ; 0647
                     03   18 0018D       BGEQ    23$                       ;
           56           0C   D0 0018F       MOVL    #12, SIZE                 ; 0649
           03    00B7   CA   91 00192 23$:  CMPB    183(IFAB), #3             ; 0655
                     0B   13 00197       BEQL    24$                       ;
           01    00B2   CA   91 00199       CMPB    178(IFAB), #1             ; 0657
                     59   1B 0019E       BLEQU   28$                       ;
           55      06   AA   E9 001A0       BLBC    6(IFAB), 28$              ; 0659
           52           56   D0 001A4 24$:  MOVL    SIZE, R2                  ; 0666
                        51   D4 001A7       CLRL    R1                        ;
               00000000G EF   16 001A9       JSB     RM$GETSPC1                ;
           6E           50   D0 001AF       MOVL    R0, STATUS                ;
           06           6E   E9 001B2       BLBC    STATUS, 25$               ;
       68  A9           51   D0 001B5       MOVL    BLK, 104(IRAB)            ; 0668
                        24   11 001B9       BRB     27$                       ;
  96   22  AA           03   E1 001BB 25$:  BBC     #3, 34(IFAB), 17$         ; 0673
           56    00B2   CA   9A 001C0       MOVZBL  178(IFAB), SIZE           ;
           0C           56   D1 001C5       CMPL    SIZE, #12                 ;
                        03   18 001C8       BGEQ    26$                       ;
           56           0C   D0 001CA       MOVL    #12, SIZE                 ;
           54      64   A9   D0 001CD 26$:  MOVL    100(IRAB), R4             ;
                        53   D4 001D1       CLRL    R3                        ;
           52           56   D0 001D3       MOVL    SIZE, R2                  ;
               00000000G EF   16 001D6       JSB     RM$RETSPC1                ;
                     FF77   31 001DC       BRW     17$                       ; 0674
  71   22  AA           03   E1 001DF 27$:  BBC     #3, 34(IFAB), 34$         ; 0683
           52           56   D0 001E4       MOVL    SIZE, R2                  ; 0690
                        51   D4 001E7       CLRL    R1                        ;
               00000000G EF   16 001E9       JSB     RM$GETSPC1                ;
           6E           50   D0 001EF       MOVL    R0, STATUS                ;
           06           6E   E9 001F2       BLBC    STATUS, 29$               ;
       6C  A9           51   D0 001F5       MOVL    BLK, 108(IRAB)            ; 0692
                        5A   11 001F9 28$:  BRB     34$                       ;
           54      68   A9   D0 001FB 29$:  MOVL    104(IRAB), R4             ; 0698
                        53   D4 001FF       CLRL    R3                        ;
           52           56   D0 00201       MOVL    SIZE, R2                  ;
               00000000G EF   16 00204       JSB     RM$RETSPC1                ;
  1C   22  AA           03   E1 0020A       BBC     #3, 34(IFAB), 31$         ;
           56    00B2   CA   9A 0020F       MOVZBL  178(IFAB), SIZE           ;
           0C           56   D1 00214       CMPL    SIZE, #12                 ;
                        03   18 00217       BGEQ    30$                       ;
           56           0C   D0 00219       MOVL    #12, SIZE                 ;
           54      64   A9   D0 0021C 30$:  MOVL    100(IRAB), R4             ;
                        53   D4 00220       CLRL    R3                        ;
           52           56   D0 00222       MOVL    SIZE, R2                  ;
```

```
                    00000000G  EF  16 00225          JSB     RM$RETSPC1                                          ;
                56      00B4   CA  3C 00228 31$:     MOVZWL  180(IFAB), SIZE                                     ; 0699
                56             06  C4 00230          MULL2   #6, SIZE                                            ;
                0C             56  D1 00233          CMPL    SIZE, #12                                           ;
                               03  18 00236          BGEQ    33$                                                 ;
                56             0C  D0 00238 32$:     MOVL    #12, SIZE                                           ;
                54       60    A9  D0 0023B 33$:     MOVL    96(IRAB), R4                                        ;
                               53  D4 0023F          CLRL    R3                                                  ;
                52             56  D0 00241          MOVL    SIZE, R2                                            ;
                    00000000G  EF  16 00244          JSB     RM$RETSPC1                                          ;
                    00000000G  EF  16 0024A          JSB     RM$DISCOMMON                                        ;
                50             6E  D0 00250          MOVL    STATUS, R0                                          ;
                               09  11 00253          BRB     36$                                                 ;
             00C3  C9       35   A8  90 00255 34$:     MOVB    53(RAB), 195(IRAB)                                  ; 0714
                50             01  D0 0025B 35$:     MOVL    #1, R0                                              ; 0715
                5E             04  C0 0025E 36$:     ADDL2   #4, SP                                              ; 0717
                        00FC   8F  BA 00261          POPR    #^M<R2,R3,R4,R5,R6,R7>                              ;
                               05  00265          RSB                                                         ;

; Routine Size:  614 bytes,    Routine Base:  RM$RMS3 + 0000

;  655          0718  1
;  656          0719  1 END
;  657          0720  1
;  658          0721  0 ELUDOM
```

```
;                                  PSECT SUMMARY

;       Name                    Bytes                           Attributes
;  RM$RMS3                        614  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         67         2          154        00:00.4


;                                  COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RM3CONN/OBJ=OBJ$:RM3CONN MSRC$:RM3CONN/UPDATE=(ENH$:RM3CONN)
```

```
; Size:          614 code + 0 data bytes
; Run Time:        00:15.4
; Elapsed Time:    00:43.0
; Lines/CPU Min:    2809
; Lexemes/CPU-Min: 17000
; Memory Used:  222 pages
; Compilation Complete
```

0324 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RM3FACE
LIS

RM3DISCON
LIS

RM3CONN
LIS

RM3DELETE
LIS

RM3CREATE
LIS