```
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**RM1JOURNL
```

```
RRRRRRRR    MM      MM      11              JJ    000000    UU      UU  RRRRRRRR    NN      NN  LL
RRRRRRRR    MM      MM      11              JJ    000000    UU      UU  RRRRRRRR    NN      NN  LL
RR      RR  MMMM  MMMM    1111              JJ  00      00  UU      UU  RR      RR  NN      NN  LL
RR      RR  MMMM  MMMM    1111              JJ  00      00  UU      UU  RR      RR  NN      NN  LL
RR      RR  MM  MM  MM      11              JJ  00      00  UU      UU  RR      RR  NNNN    NN  LL
RR      RR  MM  MM  MM      11              JJ  00      00  UU      UU  RR      RR  NNNN    NN  LL
RRRRRRRR    MM      MM      11              JJ  00      00  UU      UU  RRRRRRRR    NN  NN  NN  LL
RRRRRRRR    MM      MM      11              JJ  00      00  UU      UU  RRRRRRRR    NN  NN  NN  LL
RR  RR      MM      MM      11      JJ      JJ  00      00  UU      UU  RR  RR      NN    NNNN  LL
RR  RR      MM      MM      11      JJ      JJ  00      00  UU      UU  RR  RR      NN    NNNN  LL
RR    RR    MM      MM      11      JJ      JJ  00      00  UU      UU  RR    RR    NN      NN  LL
RR    RR    MM      MM      11      JJ      JJ  00      00  UU      UU  RR    RR    NN      NN  LL               ....
RR      RR  MM      MM    111111      JJJJJJ      000000    UUUUUUUUUU  RR      RR  NN      NN  LLLLLLLLLL       ....
RR      RR  MM      MM    111111      JJJJJJ      000000    UUUUUUUUUU  RR      RR  NN      NN  LLLLLLLLLL       ....

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
```

Table of contents

```
0000            $BEGIN  RM1JOURNL,000,RM$RMS_JOURNAL,<Sequential specific journaling>
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                *
0000    ;*   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                 *
0000    ;*   ALL RIGHTS RESERVED.                                                   *
0000    ;*                                                                          *
0000    ;*   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
0000    ;*   ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
0000    ;*   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
0000    ;*   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
0000    ;*   OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
0000    ;*   TRANSFERRED.                                                           *
0000    ;*                                                                          *
0000    ;*   THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
0000    ;*   AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    ;*   CORPORATION.                                                           *
0000    ;*                                                                          *
0000    ;*   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
0000    ;*   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000    ;
0000    ;++
0000    ;
0000    ; FACILITY:     RMS-32
0000    ;
0000    ; ABSTRACT:     This module contains the routines which journal record
0000    ;               operations performed on sequential files.
0000    ;
0000    ;
0000    ; ENVIRONMENT:  VAX/VMS Operating System
0000    ;
0000    ;
0000    ;--
0000    ;
0000    ; AUTHOR:       Tamar Krichevsky, CREATION DATE: 28-May-1983
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-005 TSK0004         Tamar Krichevsky                9-Dec-1983
0000    ;               Add support for BI journaling.
0000    ;
^000    ;               ************************************************************
0000    ;               *
0000    ;               *  THE CODE FOR BI JOURNALING OF TRUNCATE OPERATIONS HAS NOT
0000    ;               *  BEEN TESTED.
0000    ;               *
0000    ;               ************************************************************
0000    ;
0000    ;       V03-004 JWT0141         Jim Teague                      11-Nov-1983
0000    ;               Change IFB$V_RUM to IFB$V_ONLY_RU
0000    ;
```

```
0000    ;       V03-003 TSK0003         Tamar Krichevsky                5-Oct-1983
0000    ;               Use RM$RETJNLBDB and RM$ALJNLBDB instead of RM$RETBDB and
0000    ;               RM$ALBDB when allocating a large journal buffer.
0000    ;
0000    ;       V03-002 TSK0002         Tamar Krichevsky                27-Jun-1983
0000    ;               Pass journal BDB to RM$WRTJNL instead of related BDB for
0000    ;               AI operations.
0000    ;
0000    ;       V03-001 TSK0001         Tamar Krichevsky                22-Jun-1983
0000    ;               Clean up comments for MAKE_AI_JNL.
0000    ;
0000    ;**
0000
0000
C000            .SBTTL  DECLARATIONS
0000    ;
0000    ; INCLUDE FILES:
0000    ;
0000
0000    ;
0000    ; MACROS:
0000    ;
0000            $IFBDEF
0000            $BDBDEF
0000            $IRBDEF
0000            $FABDEF
0000            $RABDEF
0000            $RMSDEF
0000            $RJRDEF
0000            $CJFDEF
0000    ;
0000    ; EQUATED SYMBOLS:
0000    ;
0000
0000    ;
0000    ; OWN STORAGE:
0000    ;
0000
```

```
                     0000          .SBTTL  RM$SEQJNL - Sequential journaling setup
                     0000
                     0000   ;++
                     0000   ;
                     0000   ; FUNCTIONAL DESCRIPTION:
                     0000   ;
                     0000   ;       RM$SEQJNL is called when a sequential file record operation needs
                     0000   ;       to be journaled.  It fills in the recover journal record (RJR) with
                     0000   ;       the appropriate information and the returns to the caller.
                     0000   ;
                     0000   ; CALLING SEQUENCE:
                     0000   ;
                     0000   ;       BSBW    RM$SEQJNL
                     0000   ;
                     0000   ; INPUT PARAMETERS:
                     0000   ;
                     0000   ;       4(SP)   type of record operation to be performed
                     0000   ;       R4      BDB address
                     0000   ;       R5      Data record address
                     0000   ;       R6      Data record size
                     0000   ;       R8      RAB
                     0000   ;       R9      IRAB
                     0000   ;       R10     IFAB
                     0000   ;       R11     Impure
                     0000   ;
                     0000   ; IMPLICIT INPUTS:
                     0000   ;
                     0000   ;       IRB$L_JNLBDB    Address of journal BDB
                     0000   ;
                     0000   ; OUTPUT PARAMETERS:
                     0000   ;
                     0000   ;       R0      Status
                     0000   ;       R1 - R3 Destroyed
                     0000   ;
                     0000   ; IMPLICIT OUTPUTS:
                     0000   ;
                     0000   ;       None
                     0000   ;
                     0000   ; COMPLETION CODES:
                     0000   ;
                     0000   ;       None
                     0000   ;
                     0000   ; SIDE EFFECTS:
                     0000   ;
                     0000   ;       The journal buffer and BDB may be released and new ones allocated,
                     0000   ;       if the the existing buffer is not large enough to hold the current
                     0000   ;       record.
                     0000   ;
                     0000   ;--
                     0000
                     0000
                     0000   RM$SEQJNL::
          30    BB   0000           PUSHR   #^M<R4, R5>                     ; Save BDB and record addresses
54     30 A9    D0   0002           MOVL    IRB$L_JNLBDB(R9), R4            ; Get the journal BDB
                     0006
                     0006   ;+
                     0006   ;
```

```
                            0006   ; The buffer for the journal entry must be large enough to hold the current
                            0006   ; record and any overhead necessary to describe the record.  If the buffer is
                            0006   ; not large enough, then deallocate it and its BDB.  Then, allocate a new buffer
                            0006   ; (and buffer descriptor block) which will be large enough.
                            0006   ;
                            0006   ; If the operation being journaled is a BI $TRUNCATE or BI $PUT, with the TPT
                            0006   ; option, then skip this check.  Journaling for these operations is done block
                            0006   ; mode.  Therefore, a different set of criteria is used to check the size of
                            0006   ; the journal buffer.
                            0006   ;
                            0006   ;-
                            0006
      06 00A0 CA    02   E1 0006               BBC     #IFB$V_BI, IFB$B_JNLFLG(R10), 10$ ; If not BI, check jnl buff size
           04 AE    1F   93 000C               BITB    #<RJR$_TPT!RJR$_TRUNCATE>, 4(SP)  ; BI TPT or TRUNCATE?
                    53   12 0010               BNEQ    30$                               ; Yes, skip size check
                            0012
 51   2C A4   0048 8F    A3 0012   10$:        SUBW3   #RJR$C_RECLEN,BDB$W_ALLOC_SIZE(R4),R1 ; Ignore jrnl entry overhead
   52   66 A9    64 A9   A3 0019               SUBW3   IRB$W_ROVHDSZ(R9), -              ; Ignore the overhead for the
                            001F                       IRB$W_RTOTLSZ(R9), R2             ;    current record, also
           52    5F AA   A0 001F               ADDW2   IFB$B_FSZ(R10), R2                ; Do count fixed header part
                 51 52   B1 0023               CMPW    R2, R1                            ; Will record fit in buffer?
                    3D   1B 0026               BLEQU   30$                               ; Yes, Make and write jnl entry
                            0028
                    04   BB 0028               PUSHR   #^M<R2>                           ; Save the record size
            00000000'EF  16 002A               JSB     RM$RETJNLBDB                      ; Release this buffer and BDB
                    20   BA 0030               POPR    #^M<R5>                           ; Get record size - It is put in
                            0032                                                         ; R5 because of RM$ALDBUF
    55      00000048 8F  C0 0032               ADDL2   #RJR$C_RECLEN, R5                 ; Add in journal entry overhead
    55      000001FF 8F  C0 0039               ADDL2   #511, R5                          ; Round up to a
    55      000001FF 8F  CA 0040               BICL2   #511, R5                          ;      page boundary
            00000000'EF  16 0047               JSB     RM$ALDJNLBUF                      ; Get new buffer and BDB
                 03 50   E8 004D               BLBS    R0, 20$                           ; Keep going if everything is ok
                  005A   31 0050               BRW     EXIT                              ; Get out on error
                            0053   20$:        SSB     #BDB$V_PRM, BDB$B_FLGS(R4)        ; Mark BDB as permanent
                 18 B4   7C 0058               CLRQ    @BDB$L_ADDR(R4)                   ; Clear the top of the buffer
              30 A9 54   D0 005B               MOVL    R4, IRB$L_JNLBDB(R9)              ; Save the BDB's address
           55    04 AE   D0 005F               MOVL    4(SP), R5                         ; Restore the record address
                    04   11 0063               BRB     40$
                            0065
                            0065   ;+
                            0065   ;
                            0065   ; The journal buffer is all set.  Now, the journal entry is filled in and
                            0065   ; written to the journal.  First, fill in the overhead which describes the
                            0065   ; record.
                            0065   ;
                            0065   ;-
                            0065
           53    18 A4   D0 0065   30$:        MOVL    BDB$L_ADDR(R4), R3                ; Get the jnl buffer address
           03 A3    02   90 0069   40$:        MOVB    #RJR$C_RECORD, RJR$B_ENTRY_TYPE(R3); A record is being jnl'ed
           04 A3    00   90 006D               MOVB    #RJR$C_SEQ, RJR$B_ORG(R3)         ; File org. is sequential
        05 A3    0C AE   90 0072               MOVB    12(SP), RJR$B_OPER(R3)            ; Type of record operation
        40 A3    48 A9   D0 0076               MOVL    IRB$L_RP_VBN(R9), RJR$L_RFA0(R3); RFA is needed, get VBN part
        44 A3    4C A9   B0 007B               MOVW    IRB$L_RP_OFF(R9), RJR$W_RFA4(R3);       and offset into block
           51    48 A3   9E 0080               MOVAB   RJR$T_RIMAGE(R3), R1              ; Point to start of record image
                            0084
                            0084   ;+
                            0084   ;
```

```
                          0084   ; Do the journal-type specific stuff.
                          0084   ;
                          0084   ; AI journaling
                          0084   ;
                          0084   ;-
                          0084
13 00A0 CA    03    E1    0084            BBC     #IFB$V_AI, IFB$B_JNLFLG(R10), 50$; If not AI jnl'ing, keep going
              24    10    008A            BSBB    MAKE_AI_JNL                     ; Make the AI record image.
           1E 50    E9    008C            BLBC    R0, EXIT                        ; Get out on error
              54    DD    008F            PUSHL   R4                              ; Use jnl BDB as related BDB
        7E    03    9A    0091            MOVZBL  #CJF$_AI, -(SP)                 ; Pass jnl type to RM$WRTJNL
           FF69'    30    0094            BSBW    RM$WRTJNL                       ; Write journal entry
        5E    08    C0    0097            ADDL2   #8, SP                          ; Remove arguments from stack
           10 50    E9    009A            BLBC    R0, EXIT                        ; Get out on error
                          009D
                          009D   ;+
                          009D   ;
                          009D   ; BI journaling
                          009D   ;
                          009D   ;-
                          009D
   00A0 CA    03    93    009D   50$:     BITB    #<IFB$V_BI!IFB$V_RU>,IFB$B_JNLFLG(R10)    ; BI or RU jnling?
              09    13    00A2            BEQL    EXIT                                      ; No, then continue
                          00A4
     05 A3    1F    93    00A4            BITB    #<RJR$_TPT!RJR$_TRUNCATE>, RJR$B_OPER(R3) ; BI TPT or TRUNCATE?
              03    12    00A8            BNEQ    EXIT                            ; Yes, jnl entry was already written
            020D    30    00AA            BSBW    WRTBIJNL                        ; Write jnl entry
                          00AD
                          00AD   EXIT:
              30    BA    00AD            POPR    #^M<R4, R5>
                    05    00AF            RSB
```

```
00B0
00B0          .SBTTL  MAKE_AI_JNL - Put operation specific info in AI jnl
00B0
00B0   ;++
00B0   ;
00B0   ; FUNCTIONAL DESCRIPTION:
00B0   ;
00B0   ;       MAKE_AI_JNL moves the operation specific information in the journal
00B0   ;       entry for an AI journal.
00B0   ;
00B0   ; CALLING SEQUENCE:
00B0   ;
00B0   ;       BSBW    MAKE_AI_JNL
00B0   ;
00B0   ; INPUT PARAMETERS:
00B0   ;
00B0   ;       R1      Address of record image portion of journal buffer
00B0   ;       R3      Journal buffer address
00B0   ;       R4      Journal BDB address
00B0   ;       R5      Record address
00B0   ;       R6      Record length
00B0   ;       R8      RAB
00B0   ;       R9      IRAB
00B0   ;       R10     IFAB
00B0   ;
00B0   ; IMPLICIT INPUTS:
00B0   ;
00B0   ;       None
00B0   ;
00B0   ; OUTPUT PARAMETERS:
00B0   ;
00B0   ;       R0      Status
00B0   ;       R1 - R3 Destroyed
00B0   ;
00B0   ; IMPLICIT OUTPUTS:
00B0   ;
00B0   ;       None
00B0   ;
00B0   ; COMPLETION CODES:
00B0   ;
00B0   ;       RHB or RBF
00B0   ;
00B0   ; SIDE EFFECTS:
00B0   ;
00B0   ;       None
00B0   ;
00B0   ;--
00B0
00B0
00B0   MAKE_AI_JNL:
00B0
00B0   ;+
00B0   ;
00B0   ; Fill in AI specific information in the journal entry.  Then if the record
00B0   ; being journaled is VFC format, copy the fixed header portion into the
00B0   ; the record image.
00B0   ;
```

```
RM
Sy
$$
$$
$$
$$
$$
BD
BD
BD
BD
BD
BD
BD
BI
BI
CH
CJ
CJ
CJ
CO
ER
ER
EX
EX
EX
FA
FA
FA
FA
FA
FA
FA
FA
IF
IF
IF
IF
IF
IF
IF
IF
IF
IF
IF
IF
IR
IR
IR
IR
IR
IR
IR
IR
IR
IR
IR
MA
MA
```

```
                                  00B0  ;-
                                  00B0
                         30  BB   00B0          PUSHR   #^M<R4, R5>                     ; Save jnl bdb and record adr
                   50    01  D0   00B2          MOVL    #1, R0                          ; Assume success
              06  A3     01  90   00B5          MOVB    #RJR$C_RMS_AI, RJR$B_JNL_TYPE(R3); This is an AI journal entry
        14  A4     0048  8F  B0   00B9          MOVW    #RJR$C_RECLEN, BDB$W_NUMB(R4)   ; Journal entry contAIns at
                                  00BF                                                  ;     least the overhead
              05  A3     1B  91   00BF          CMPB    #RJR$_TRUNCATE, RJR$B_OPER(R3)  ; Is the operation truncation?
                         57  13   00C3          BEQL    TRUNC_ENTRY                     ; No need to copy any data
                                  00C5
              46  A3     56  B0   00C5          MOVW    R6, RJR$W_RSIZE(R3)             ; Save the record's size
              14  A4     56  A0   00C9          ADDW2   R6, BDB$W_NUMB(R4)              ; Add rec len to jnl entry size
              50  AA     03  91   00CD          CMPB    #FAB$C_VFC, IFB$B_RFMORG(R10)   ; Is the record VFC format?
                         27  12   00D1          BNEQ    40$                             ; No, copy the record
                                  00D3
                                  00D3  ;+
                                  00D3  ;
                                  00D3  ; Copy the fixed header portion of the record.
                                  00D3  ;
                                  00D3  ;-
                                  00D3
              52   5F  AA    9A   00D3          MOVZBL  IFB$B_FSZ(R10), R2              ; Get length of fixed hdr part
              14  A4     52  A0   00D7          ADDW2   R2, BDB$W_NUMB(R4)              ; Count FSZ in jnl entry size
              50   2C  A8    D0   00DB          MOVL    RAB$L_RHB(R8), R0               ; Get adr of user's hdr buffer
                         08  12   00DF          BNEQ    20$                             ; If buffer adr given, copy hdr
      61    52    00    60   00  2C   00E1      MOVC5   #0, (R0), #0, R2, (R1)          ; Zero hdr part of record image
                         0B  11   00E7          BRB     30$                             ; Copy variable portion of rec
                                  00E9  20$:    IFNORD  R2, (R5), ERRRHB, IRB$B_MODE(R9); Quit if hdr can't be read
              61    65   52  28   00F0          MOVC3   R2, (R5), (R1)                  ; Copy hdr part to record image
                   51    53  D0   00F4  30$:    MOVL    R3, R1                          ; Point to next byte in rec image
                   54    6E  7D   00F7          MOVQ    (SP), R4                        ; Retrieve BDB and rec adr
                                  00FA
                                  00FA  ;+
                                  00FA  ;
                                  00FA  ; Copy the record to the journal entry.
                                  00FA  ;
                                  00FA  ;-
                                  00FA
           0200 8F       56  B1   00FA  40$:    CMPW    R6, #512                        ; Is record longer than a page?
                         0B  1B   00FF          BLEQU   50$                             ; Yes, do short read probe
            00000000'EF      16   0101          JSB     RMS$PROBEREAD                   ; No, do a long probe
                    1C   50  E9   0107          BLBC    R0, ERRBUF                      ; Get out on error
                         0B  11   010A          BRB     60$                             ; Continue processing
                                  010C  50$:    IFNORD  R6, (R5), ERRBUF, IRB$B_MODE(R9); Quit if record can't be read
              61    65   56  28   0113          MOVC3   R6, (R5), (R1)                  ; Copy record to jnl entry
                                  0117  60$:    RMSSUC                                  ; Journal entry is complete
                         00  11   011A          BRB     EXIT_AI_RTN
                                  011C
                                  011C  ;+
                                  011C  ;
                                  011C  ; All the information necessary for AI recovery of a $TRUNCATE operation is
                                  011C  ; already in the RMS journal record (RJR).  Therefore, no further modification
                                  011C  ; needs to be done to the journal entry.
                                  011C  ;
                                  011C  ;-
                                  011C
                                  011C  TRUNC_ENTRY:
```

```
            011C  EXIT_AI_RTN:
30    BA    011C          POPR    #^M<R4, R5>
      05    011E          RSB                                     ; Return to caller
            011F
            011F  ERRRHB:
            011F          RMSERR RHB
F6    11    0124          BRB     EXIT_AI_RTN
            0126
            0126  ERRBUF:
            0126          RMSERR  RBF
EF    11    012B          BRB     EXIT_AI_RTN
```

```
                    012D                 .SBTTL  MAKE_BI_JNL - Put operation specific info in BI jnl
                    012D
                    012D          ;++
                    012D          ;
                    012D          ; FUNCTIONAL DESCRIPTION:
                    012D          ;
                    012D          ;       MAKE_BI_JNL moves the operation specific information in the journal
                    012D          ;       entry for an BI journal.
                    012D          ;
                    012D          ; CALLING SEQUENCE:
                    012D          ;
                    012D          ;       BSBW    MAKE_BI_JNL
                    012D          ;
                    012D          ; INPUT PARAMETERS:
                    012D          ;
                    012D          ;       R1      Address of record image portion of journal buffer
                    012D          ;       R3      Journal buffer address
                    012D          ;       R4      Journal BDB address
                    012D          ;       R6      Record length
                    012D          ;       R8      RAB
                    012D          ;       R9      IRAB
                    012D          ;       R10     IFAB
                    012D          ;
                    012D          ; IMPLICIT INPUTS:
                    012D          ;
                    012D          ;       None
                    012D          ;
                    012D          ; OUTPUT PARAMETERS:
                    012D          ;
                    012D          ;       R0      Status
                    012D          ;       R1 - R3 Destroyed
                    012D          ;
                    012D          ; IMPLICIT OUTPUTS:
                    012D          ;
                    012D          ;       None
                    012D          ;
                    012D          ; COMPLETION CODES:
                    012D          ;
                    012D          ;       Any completion code returned by RM$NXTBLK1
                    012D          ;
                    012D          ; SIDE EFFECTS:
                    012D          ;
                    012D          ;       None
                    012D          ;
                    012D          ;--
                    012D
                    012D
                    012D          MAKE_BI_JNL:
                    012D
                    012D          ;+
                    012D          ;
                    012D          ; Fill in BI/RU specific information in the journal entry.
                    012D          ;
                    012D          ;-
                    012D
   00F0 8F     BB   012D                 PUSHR   #^M<R4, R5, R6, R7>            ; Save jnl bdb, record adr & len
```

```
        50   01    D0  0131             MOVL    #1, R0                          ; Assume success
  14 A4    0048 8F  B0  0134             MOVW    #RJR$C_RECLEN, BDB$W_NUMB(R4)   ; Journal entry contains at
                       013A                                                     ;       least the overhead
     05 A3   13    91  013A             CMPB    #RJR$_PUT, RJR$B_OPER(R3)       ; Is the operation $PUT?
             0C    12  013E             BNEQ    20$                             ; No, move data to jnl entry
           0169    31  0140             BRW     PUT_ENTRY                       ; Yes, no need to move data
     05 A3   1C    91  0143     10$:    CMPB    #RJR$_UPDATE, RJR$B_OPER(R3)    ; Is the operation $UPDATE?
             03    13  0147             BEQL    20$                             ; Yes
           0086    31  0149             BRW     BI_TRUNC_ENTRY                  ; No, it's truncate on put, or $TRUN
                       014C
                       014C
                       014C     ;+
                       014C     ;
                       014C     . \djust journal entry size to compensate for any overhead.  If it is VFC,
                       014C     ; include size of fixed header portion.  For UDF, VAR, FIX and VFC do not add in
                       014C     ; size of control (count) field.  Do include overhead for STM, STMLF and STMCR.
                       014C     ; The terminators are counted as overhead, but are also part of the record.
                       014C     ;
                       014C     ;-
                       014C
     46 A3   56    B0  014C     20$:    MOVW    R6, RJR$W_RSIZE(R3)             ; Save rec size in jnl entry
                       0150
                       0150             ASSUME  FAB$C_VFC   GT FAB$C_UDF
                       0150             ASSUME  FAB$C_VFC   GT FAB$C_VAR
                       0150             ASSUME  FAB$C_VFC   GT FAB$C_FIX
                       0150             ASSUME  FAB$C_STM   GT FAB$C_VFC
                       0150             ASSUME  FAB$C_STMLF GT FAB$C_VFC
                       0150             ASSUME  FAB$C_STMCR GT FAB$C_VFC
                       0150             ASSUME  FAB$C_STMCR EQ FAB$C_MAXRFM
                       0150
     50 AA   03    91  0150             CMPB    #FAB$C_VFC, IFB$B_RFMORG(R10)   ; Is the record VFC format?
             08    1F  0154             BLSSU   30$                             ; No, ignore overhead (count field)
             16    1A  0156             BGTRU   40$                             ; No, include overhead (terminators)
     56   5F AA    80  0158             ADDB2   IFB$B_FSZ(R10), R6              ; Yes, include header portion
             10    11  015C             BRB     40$
   00A1 CA   02    93  015E     30$:    BITB    #<IFB$V_BI_RECVR!IFB$V_RU_RECVR>, IFB$B_RECVRFLGS(R10) ; If in recov
             09    12  0163             BNEQ    40$                             ;   terminators are already counted
     56   64 A9    A0  0165             ADDW2   IRB$W_ROVHDSZ(R9), R6           ; Stream format, include overhead
  46 A3   64 A9    A0  0169             ADDW2   IRB$W_ROVHDSZ(R9), RJR$W_RSIZE(R3) ; Add overhead to jnl entry size
     14 A4   56    A0  016E     40$:    ADDW2   R6, BDB$W_NUMB(R4)              ; Increase size of jnl buffer
                       0172
                       0172     ;+
                       0172     ;
                       0172     ; Locate the first byte of the data to be copied to the journal entry.
                       0172     ; NOTE -- This assumes 512 byte blocks.
                       0172     ;
                       0172     ;-
                       0172
     50   20 A9    D0  0172             MOVL    IRB$L_CURBDB(R9), R0            ; Retrieve BDB for buffer
     55   48 A0    9A  0176             MOVZBL  BDB$B_REL_VBN(R0), R5           ; Get block containing record
     55   48 AA    C4  017A             MULL2   IFB$L_DEVBUFSIZ(R10), R5        ; Convert to byte offset
     55   18 A0    C0  017E             ADDL2   BDB$L_ADDR(R0), R5              ; Add offset to buffer address
     50   4C A9    3C  0182             MOVZWL  IRB$W_RP_OFF(R9), R0            ; Get offset with in block
        55   50    C0  0186             ADDL2   R0, R5                          ; Point to first byte of record
                       0189
                       0189     ;+
                       0189     ;
                       0189  ; If there is a count field preceeding the record, skip over it so that we are
                       0189  ; truely pointing to the first byte of the record.  Since the total record size
                       0189  ; includes the count field, if that value is different from the one calculated
                       0189  ; for the journal entry, then the record has a count field and it should be
                       0189  ; skipped.
                       0189  ;
                       0189  ;-
                       0189
  50   66 A9     3C    0189          MOVZWL  IRB$W_RTOTLSZ(R9), R0           ; Get total record size
     50   56     C2    018D          SUBL2   R6, R0                          ; Determine count field length
     55   50     C0    0190          ADDL2   R0, R5                          ; Move pointer over count field
                       0193
                       0193  ;+
                       0193  ;
                       0193  ; Save the current record pointer, in case the record crosses into the next
                       0193  ; buffer causing the rest of the record is read into the buffer.  After the
                       0193  ; whole record has been copied to the journal entry, the current record pointer
                       0193  ; will be needed to restore the current contents of the buffer.
                       0193  ;
                       0193  ;-
                       0193
                       0193          ASSUME  IRB$W_RP_OFF EQ <IRB$L_RP_VBN + 4>
  7E   48 A9     7D    0193          MOVQ    IRB$L_RP_VBN(R9), -(SP)
                       0197
                       0197  ;+
                       0197  ;
                       0197  ; Copy the record to the journal entry.  The current register contents are:
                       0197  ;
                       0197  ;       R1 - address of first byte of RJR record image (destination)
                       0197  ;       R5 - first byte of record in buffer (source)
                       0197  ;       R6 - number of bytes to transfer to journal entry
                       0197  ;       R7 - end of buffer address + 1
                       0197  ;
                       0197  ;-
                       0197
                       0197  COPY_DATA:
 50   57   55    C3    0197          SUBL3   R5, R7, R0                      ; Get # of bytes left in source buff
      56   50    D1    019B          CMPL    R0, R6                          ; Is whole record in buffer?
           03    1B    019E          BLEQU   10$                             ; No, transfer size = remaining buff
      50   56    D0    01A0          MOVL    R6, R0                          ; Yes, use rec len as transfer size
      56   50    C2    01A3  10$:    SUBL2   R0, R6                          ; Adjust size of record
 61   65   50    28    01A6          MOVC3   R0, (R5), (R1)                  ; Copy the (partial) record
           56    D5    01AA          TSTL    R6                              ; Any data left to copy?
           02    12    01AC          BNEQ    20$                             ; Yes, refill buffer, copy rest of r
           15    11    01AE          BRB     RESTORE_BUFF                    ; No, copy is complete
           08    BB    01B0  20$:    PUSHR   #^M<R3>                         ; Save source and destination
         00FC    30    01B2          BSBW    CHANGE_BUFF                     ; Get next buffer
      55   51    D0    01B5          MOVL    R1, R5                          ; Save source location
           51  8ED0    01B8          POPL    R1                              ; Restore the destination
        D9 50    E8    01BB          BLBS    R0, COPY_DATA                   ; Copy rest of record or fall thru t
                       01BE
                       01BE  BI_ERROR_EXIT:
                       01BE          ASSUME  IRB$W_RP_OFF EQ <IRB$L_RP_VBN + 4>
   48 A9   8E    7D    01BE          MOVQ    (SP)+, IRB$L_RP_VBN(R9)         ; Retrieve record pointer
         00E7    31    01C2          BRW     EXIT_BI_RTN                     ; Return with error status
                       01C5
                       01C5  RESTORE_BUFF:
```

```
                       01C5        ASSUME  IRB$W_NRP_OFF EQ <IRB$L_NRP_VBN + 4>
   40 A9    8E    7D   01C5        MOVQ    (SP)+, IRB$L_NRP_VBN(R9)         ; Retrieve record pointer
  00000000'EF     16   01C9        JSB     RM$GETBLKNRP                     ; Restore contents of the buffer
          00DA    31   01CF        BRW     EXIT_BI_RTN                      ; Return with error status
```

```
                         01D2   ;+
                         01D2   ;
                         01D2   ; The current operation involves truncation ($TRUNCATE or $PUT, with TPT set).
                         01D2   ; The rest of the file must be copied to the journal.  This is done one buffer
                         01D2   ; at a time, from the current VBN to the EOF.  The journal entries are formatted
                         01D2   ; as BLOCK I/O entries, not record entries.  Recovery should be done as a series
                         01D2   ; of $WRITEs.
                         01D2   ;
                         01D2   ;
                         01D2
                         01D2   ;
                         01D2   ;       ***************************************************************
                         01D2   ;       *
                         01D2   ;       *  THE CODE FOR BI JOURNALING OF TRUNCATE OPERATIONS HAS NOT
                         01D2   ;       *  BEEN TESTED.
                         01D2   ;       *
                         01D2   ;       ***************************************************************
                         01D2
                         01D2  BI_TRUNC_ENTRY:
                         01D2
                         01D2   ;
                         01D2   ; Write all dirty buffers out to the disk, to be sure that the file is in a
                         01D2   ; consistent state before ar  data is copied to the journal.
                         01D2   ;
                         01D2
     00000000'EF    16   01D2           JSB     RM$FLUSH                        ; Write buffers to disk
           3F 50    E8   01D8           BLBS    R0, 10$                         ; If that worked, keep going
            00CE    31   01DB           BRW     EXIT_BI_RTN                     ; Get out on error
                         01DE
                         01DE   ;
                         01DE   ; Determine the maximum size for the journal entry.  Check to see if it fits
                         01DE   ; in the current journal buffer.
                         01DE   ;
                         01DE
      55    55 A9   9A   01DE           MOVZBL  IRB$B_MBC(R9), R5               ; at most, MBC # of blks will be cop
              55    D6   01E2           INCL    R5                              ; MBC is zero based, not one based
      55    48 AA   C4   01E4           MULL2   IFB$L_DEVBUFSIZ(R10), R5        ; Convert to bytes
55   00000044 8F    C0   01E8           ADDL    #RJR$C_BLKLEN, R5               ; Include jnl entry overhead in size
      55    2C A4   B1   01EF           CMPW    BDB$W_ALLOC_SIZE(R4), R5        ; Will it fit in the curr buff?
              2D    1E   01F3           BGEQU   20$                             ; Yes, continue processing
                         01F5
                         01F5   ;
                         01F5   ; Get a new journal buffer is needed; the current one is too small.  Initialize
                         01F5   ; any journal entry fields which are assumed to already have values in them.
                         01F5   ;
                         01F5
              55    DD   01F5           PUSHL   R5                              ; Save jnl buff size
     00000000'EF    16   01F7           JSB     RM$RETJNLBDB                    ; Release this buffer
55   000001FF 8F    C0   01FD           ADDL2   #511, R5                        ; Round the number of bytes to
55   000001FF 8F    CA   0204           BICL    #511, R5                        ;    up to a page boundary
              55  8ED0   020B           POPL    R5                              ; Restore jnl buffer size
     00000000'EF    16   020E           JSB     RM$ALDJNLBUF                    ; Get a new BDB and buffer
           03 50    E8   0214           BLBS    R0, 10$                         ; Continue if new BDB is okay
            0092    31   0217           BRW     EXIT_BI_RTN
      30 A9   54    D0   021A  10$:     MOVL    R4, IRB$L_JNLBDB(R9)            ; Save the jnl BDB address
      04 A3   00    90   021E           MOVB    #RJR$C_SEQ, RJR$B_ORG(R3)       ; File is sequential organization
                         0222
```

```
                          0222  ;
                          0222  ; Initialize the journal BDB and the journal entry. The jnl entry should look
                          0222  ; like a BLOCK I/O operation is happening.
                          0222  ;
                          0222
   1C A4   48 AA     D0   0222  20$:    MOVL    IRB$L_RP_VBN(R10), BDB$L_VBN(R4); Start VBN is VBN of curr rec
55   00000044 8F     C2   0227          SUBL2   #RJR$C_BLKLEN, R5              ; Ovrhd not included in # bytes to j
     14 A4    55     B0   022E          MOVW    R5, BDB$W_NUMB(R4)             ; Size of tranfer into jnl entry
     03 A3    03     90   0232          MOVB    #RJR$C_BLOCK, RJR$B_ENTRY_TYPE(R3)    ; Block mode I/O
     05 A3    1E     90   0236          MOVB    #RJR$_WRITE, RJR$B_OPER(R3)           ; Operation is psuedo-$WRITE
   3C A3   1C A4     D0   023A          MOVL    BDB$L_VBN(R4), RJR$L_BLOCK_VBN(R3)    ; VBN of 1st blk being jnl'd
   40 A3   14 A4     3C   023F          MOVZWL  BDB$W_NUMB(R4), RJR$C_BLOCK_SIZE(R3) ; # of bytes being jnl'd
                          0244
                          0244  ;+
                          0244  ;
                          0244  ; Do until beyond EOF:
                          0244  ;     If EOF is in current buffer, set the number of bytes to journal so that
                          0244  ;     only data up to the first free byte is read into the journal buffer.
                          0244  ;     Read data into the journal entry and write the entry to the journal.
                          0244  ;     Determine the start VBN for the next buffer.
                          0244  ;
                          0244  ;-
                          0244
      51   55 A9     9A   0244          MOVZBL  IRB$B_MBC(R9), R1                ; EOF is in buffer if:
              51     D6   0248          INCL    R1                               ;  (MBC + 1) + start VBN
  52  51   48 A9     C1   024A          ADDL3   IRB$L_RP_VBN(R9), R1, R2         ;  is greater than EBK
                          024F  MAKE_TRUNC_ENTRY:
      74 AA   52     D1   024F          CMPL    R2, IFB$L_EBK(R10)               ; Is EOF in the current buffer?
              0F     1F   0253          BLSSU   10$                              ; No, journal whole buffer
 50   48 AA   5C AA  A3   0255          SUBW3   IFB$W_FFB(R10), IFB$L_DEVBUFSIZ(R10), R0 ; How many bytes are unused
      14 A4   50     A2   025B          SUBW2   R0, BDB$W_NUMB(R4)               ; Decrement # of bytes to jnl
   40 A3   14 A4     3C   025F          MOVZWL  BDB$W_NUMB(R4), RJR$L_BLOCK_SIZE(R3) ; Same for jnl entry size
                          0264
                          0264  ;
                          0264  ; Read VBNs into jnl buffer from the disk.
                          0264  ;
                          0264
                          0264          ASSUME  RJR$C_BLKLEN EQ RJR$T_BLOCK
18 A4   00000044 8F  C0   0264  10$:    ADDL2   #RJR$C_BLKLEN, BDB$L_ADDR(R4)    ; Use RJR$T_BLOCK as dest for read
              0E     BB   026C          PUSHR   #^M<R1, R2, R3>                  ; Save pointers and counters
     00000000'EF     16   026E          JSB     RM$RDBUFWT                       ; Read in data and wait for completi
18 A4   00000044 8F  C2   0274          SUBL2   #RJR$C_BLKLEN, BDB$L_ADDR(R4)    ; Return to real start of jnl buffer
           05 50     E8   027C          BLBS    R0, 20$                          ; If read worked, continue
              0E     BA   027F          POPR    #^M<R1, R2, R3>                  ; Otherwise, restore regs
            0028     31   0281          BRW     EXIT_BI_RTN                      ; Get our on error
                          0284
                          0284  ;
                          0284  ; Write journal entry out to journal.
                          0284  ;
                          0284
   14 A4    0044 8F  A0   0284  20$:    ADDW2   #RJR$C_BLKLEN, BDB$W_NUMB(R4)    ; Ovrhd included in jnl entry size
              2E     10   028A          BSBB    WRTBIJNL                         ; Write jnl entry
              0E     BA   028C          POPR    #^M<R1, R2, R3>                  ; Restore pointers and counters
           1B 50     E9   028E          BLBC    R0, EXIT_BI_RTN                  ; Get out on error
   14 A4    0044 8F  A2   0291          SUBW2   #RJR$C_BLKLEN, BDB$W_NUMB(R4)    ; Remove ovrhd from jnl entry size
                          0297
                          0297  ;
```

```
                         0297  ; Determine start and end VBN of next buffer.
                         0297  ;
                         0297
    1C A4    52    D0    0297          MOVL    R2, BDB$L_VBN(R4)                  ; Start VBN was already calculated
3C A3    1C A4    D0    029B          MOVL    BDB$L_VBN(R4), RJR$L_BLOCK_VBN(R3) ; Save start VBN in jnl entry
       52    51    C0    02A0          ADDL2   R1, R2                             ; Get 1st VBN past next buffer
    74 AA    52    B1    02A3          CMPW    R2, IFB$L_EBK(R10)                 ; Is EOF in next buffer?
             03    1A    02A7          BGTRU   EXIT_BI_RTN                        ; No, do not jnl past EOF (it has be
           FFA3    31    02A9          BRW     MAKE_TRUNC_ENTRY                   ; Journal next set of blocks
                         02AC
                         02AC
                         02AC  PUT_ENTRY:
                         02AC  EXIT_BI_RTN:
      00F0 8F      BA    02AC          POPR    #^M<R4, R5, R6, R7>
                   05    02B0          RSB                                        ; Return to caller
                         02B1
```

```
                          02B1          .SBTTL  CHANGE_BUFF - get next buffer
                          02B1
                          02B1   ;++
                          02B1   ;
                          02B1   ; FUNCTIONAL DESCRIPTION:
                          02B1   ;
                          02B1   ;      CHANGE_BUFF calls RM$NXTBLK1 for MAKE_BI_JNL.
                          02B1   ;
                          02B1   ; CALLING SEQUENCE:
                          02B1   ;
                          02B1   ;      BSBB    CHANGE_BUFF
                          02B1   ;
                          02B1   ; INPUT PARAMETERS:
                          02B1   ;
                          02B1   ;      R8      RAB
                          02B1   ;      R9      IRAB
                          02B1   ;      R10     IFAB
                          02B1   ;
                          02B1   ; IMPLICIT INPUTS:
                          02B1   ;
                          02B1   ;      None
                          02B1   ;
                          02B1   ; OUTPUT PARAMETERS:
                          02B1   ;
                          02B1   ;      R0      Status
                          02B1   ;      R1 - R3 Destroyed
                          02B1   ;
                          02B1   ; IMPLICIT OUTPUTS:
                          02B1   ;
                          02B1   ;      R1      address of current block in buffer
                          02B1   ;      R7      address of end of buffer + 1
                          02B1   ;
                          02B1   ; COMPLETION CODES:
                          02B1   ;
                          02B1   ;      Any completion code returned by RM$NXTBLK1
                          02B1   ;
                          02B1   ; SIDE EFFECTS:
                          02B1   ;
                          02B1   ;      None
                          02B1   ;
                          02B1   ;--
                          02B1
                          02B1   CHANGE_BUFF:
               53     D4  02B1          CLRL    R3                          ; Indicate read required
        00000000'EF   16  02B3          JSB     RM$NXTBLK1                  ; Get new buffer contents
                      05  02B9          RSB
                          02BA
                          02BA          .SBTTL  WRTBIJNL - writes BI/RU journal entry
                          02BA
                          02BA  ;++
                          02BA  ;
                          02BA  ; FUNCTIONAL DESCRIPTION:
                          02BA  ;
                          02BA  ;       WRTBIJNL writes a BI/RU jnl entry
                          02BA  ;
                          02BA  ; CALLING SEQUENCE:
                          02BA  ;
                          02BA  ;       BSBB    WRTBIJNL
                          02BA  ;
                          02BA  ; INPUT PARAMETERS:
                          02BA  ;
                          02BA  ;       R4      Journal BDB
                          02BA  ;       R8      RAB
                          02BA  ;       R9      IRAB
                          02BA  ;       R10     IFAB
                          02BA  ;
                          02BA  ; IMPLICIT INPUTS:
                          02BA  ;
                          02BA  ;       None
                          02BA  ;
                          02BA  ; OUTPUT PARAMETERS:
                          02BA  ;
                          02BA  ;       R0      Status
                          02BA  ;       R1      Destroyed
                          02BA  ;
                          02BA  ; IMPLICIT OUTPUTS:
                          02BA  ;
                          02BA  ;       None
                          02BA  ;
                          02BA  ; COMPLETION CODES:
                          02BA  ;
                          02BA  ;       Any completion code returned by RM$WRTJNL
                          02BA  ;
                          02BA  ; SIDE EFFECTS:
                          02BA  ;
                          02BA  ;       None
                          02BA  ;
                          02BA  ;--
                          02BA
                          02BA  WRTBIJNL:
12 00A0 CA    02    E1    02BA          BBC     #IFB$V_BI, IFB$B_JNLFLG(R10), 10$ ; If BI/RU jnl'ing, write a BI/RU
     06 A3    02    90    02C0          MOVB    #RJR$C_RMS_BI, RJR$B_JNL_TYPE(R3) ; This is a BI journal entry
              54    DD    02C4          PUSHL   R4                                ; Use jnl BDB as relate BDB
        7E    02    9A    02C6          MOVZBL  #CJF$_BI, -(SP)                   ; Pass jnl type to WRTBIJNL
          FD34'     30    02C9          BSBW    RM$WRTJNL                         ; Write jnl entry
        5E    08    C0    02CC          ADDL2   #8, SP                            ; Remove args from stack
          15 50     E9    02CF          BLBC    R0, 20$                           ; Get out on error
                          02D2
0F 00A0 CA    01    E1    02D2  10$:    BBC     #IFB$V_RU, IFB$B_JNLFLG(R10), 20$ ; If RU jnl'ing, write a RU entry
     06 A3    03    90    02D8          MOVB    #RJR$C_RMS_RU, RJR$B_JNL_TYPE(R3) ; This is an RU journal entry
              54    DD    02DC          PUSHL   R4                                ; Use jnl BDB as relate BDB
        7E    01    9A    02DE          MOVZBL  #CJF$_RU, -(SP)                   ; Pass jnl type to WRTBIJNL
          FD1C'     30    02E1          BSBW    RM$WRTJNL                         ; Write jnl entry
        5E    08    C0    02E4          ADDL2   #8, SP                            ; Remove args from stack
```

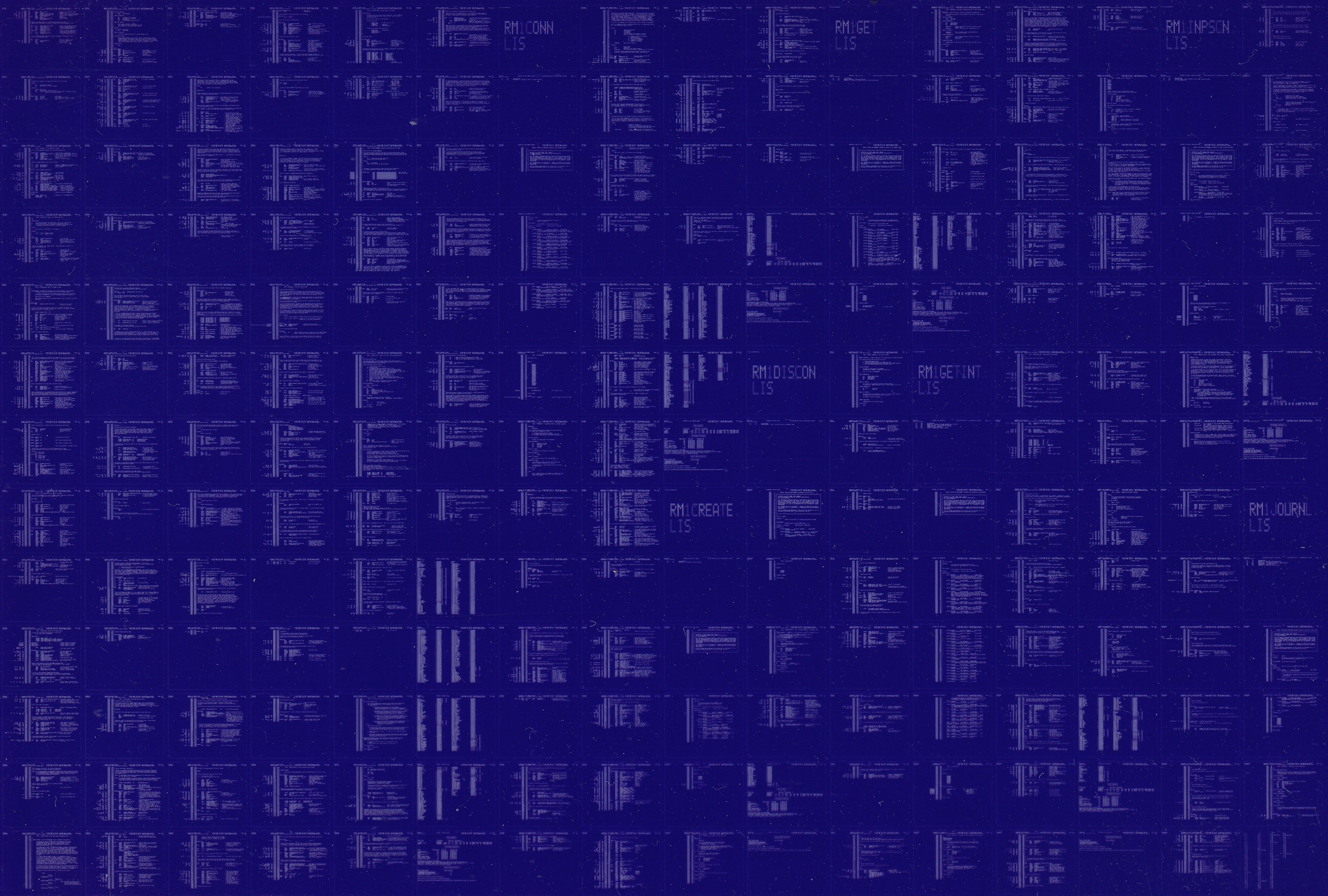
```
    02E7   781
05  02E7   782 20$:    RSB
    02E8   783
    02E8   784         .END
```

```
$$.PSECT_EP                = 00000000                  MAKE_TRUNC_ENTRY                  0000024F R      01
$$RMSTEST                  = 0000001A                  PUT_ENTRY                         000002AC R      01
$$RMS_PBUGCHK              = 00000010                  RAB$L_RHB                       = 0000002C
$$RMS_TBUGCHK              = 00000008                  RESTORE_BUFF                      000001C5 R      01
$$RMS_UMODE                = 00000004                  RJR$B_ENTRY_TYPE                = 00000003
BDB$B_FLGS                 = 0000000A                  RJR$B_JNL_TYPE                  = 00000006
BDB$B_REL_VBN              = 00000048                  RJR$B_OPER                      = 00000005
BDB$L_ADDR                 = 00000018                  RJR$B_ORG                       = 00000004
BDB$L_VBN                  = 0000001C                  RJR$C_BLKLEN                    = 00000044
BDB$V_PRM                  = 00000003                  RJR$C_BLOCK                     = 00000003
BDB$W_ALLOC_SIZE           = 0000002C                  RJR$C_RECLEN                    = 00000048
BDB$W_NUMB                 = 00000014                  RJR$C_RECORD                    = 00000002
BI_ERROR_EXIT                000001BE R      01        RJR$C_RMS_AI                    = 00000001
BI_TRUNC_ENTRY               000001D2 R      01        RJR$C_RMS_BI                    = 00000002
CHANGE_BUFF                  000002B1 R      01        RJR$C_RMS_RU                    = 00000003
CJF$_AI                    = 00000003                  RJR$C_SEQ                       = 00000000
CJF$_BI                    = 00000002                  RJR$L_BLOCK_SIZE                = 00000040
CJF$_RU                    = 00000001                  RJR$L_BLOCK_VBN                 = 0000003C
COPY_DATA                    00000197 R      01        RJR$L_RFA0                      = 00000040
ERRBUF                       00000126 R      01        RJR$T_BLOCK                     = 00000044
ERRRHB                       0000011F R      01        RJR$T_RIMAGE                    = 00000048
EXIT                         000000AD R      01        RJR$W_RFA4                      = 00000044
EXIT_AI_RTN                  0000011C R      01        RJR$W_RSIZE                     = 00000046
EXIT_BI_RTN                  000002AC R      01        RJR$_PUT                        = 00000013
FAB$C_FIX                  = 00000001                  RJR$_TPT                        = 0000001F
FAB$C_MAXRFM               = 00000006                  RJR$_TRUNCATE                   = 0000001B
FAB$C_STM                  = 00000004                  RJR$_UPDATE                     = 0000001C
FAB$C_STMCR                = 00000006                  RJR$_WRITE                      = 0000001E
FAB$C_STMLF                = 00000005                  RM$ACDJNLBUF                      ********    X   01
FAB$C_UDF                  = 00000000                  RM$FLUSH                          ********    X   01
FAB$C_VAR                  = 00000002                  RM$GETBLKNRP                      ********    X   01
FAB$C_VFC                  = 00000003                  PM$NXTBLK1                        ********    X   01
IFB$B_FSZ                  = 0000005F                  RM$PROBEREAD                      ********    X   01
IFB$B_JNLFLG               = 000000A0                  RM$RDBUFWT                        ********    X   01
IFB$B_RECVRFLGS            = 000000A1                  RM$RETJNLBDB                      ********    X   01
IFB$B_RFMORG               = 00000050                  RM$SEQJNL                         00000000 RG     01
IFB$L_DEVBUFSIZ            = 00000048                  RM$WRTJNL                         ********    X   01
IFB$L_EBK                  = 00000074                  RMS$_RBF                        = 00018654
IFB$V_AI                   = 00000003                  RMS$_RHB                        = 0001866C
IFB$V_BI                   = 00000002                  TRUNC_ENTRY                       0000011C R      01
IFB$V_BI_RECVR             = 00000002                  WRTBIJNL                          000002BA R      01
IFB$V_RU                   = 00000001
IFB$V_RU_RECVR             = 00000000
IFB$W_FFB                  = 0000005C
IRB$B_MBC                  = 00000055
IRB$B_MODE                 = 0000000A
IRB$L_CURBDB               = 00000020
IRB$L_JNLBDB               = 00000030
IRB$L_NRP_VBN              = 00000040
IRB$L_RP_OFF               = 0000004C
IRB$L_RP_VBN               = 00000048
IRB$W_NRP_OFF              = 00000044
IRB$W_ROVHDSZ              = 00000064
IRB$W_RP_OFF               = 0000004C
IRB$W_RTOTLSZ              = 00000066
MAKE_AI_JNL                  000000B0 R      01
MAKE_BI_JNL                  0000012D R      01
```

```
                                            +----------------+
                                            ! Psect synopsis !
                                            +----------------+

PSECT name                   Allocation           PSECT No.  Attributes
----------                   ----------           ---------  ----------
.  ABS  .                    00000000  (      0.)  00 (  0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
RMSRMS_JOURNAL               000002E8  (    744.)  01 (  1.)    PIC    USR    CON    REL    GBL NOSHR   EXE   RD   NOWRT NOVEC BYTE
$ABS$                        00000000  (      0.)  02 (  2.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE

                                          +------------------------+
                                          ! Performance indicators !
                                          +------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                   29    00:00:00.08    00:00:00.75
Command processing              116    00:00:00.64    00:00:04.98
Pass 1                          351    00:00:12.00    00:00:36.97
Symbol table sort                 0    00:00:01.78    00:00:03.05
Pass 2                          138    00:00:02.85    00:00:07.67
Symbol table output              13    00:00:00.11    00:00:00.20
Psect synopsis output             1    00:00:00.02    00:00:00.02
Cross-reference output            0    00:00:00.00    00:00:00.00
Assembler run totals            650    00:00:17.48    00:00:53.64

The working set limit was 1650 pages.
70414 bytes (138 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1275 non-local and 23 local symbols.
784 source lines were read in Pass 1, producing 14 object records in Pass 2.
23 pages of virtual memory were used to define 22 macros.

                                         +--------------------------+
                                         ! Macro library statistics !
                                         +--------------------------+

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                    12
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                     1
_$255$DUA28:[SYSLIB]STARLET.MLB;2                  5
TOTALS (all libraries)                            18

1385 GETS were required to define 18 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:RM1JOURNL/OBJ=OBJ$:RM1JOURNL MSRC$:RM1JOURNL/UPDATE=(ENH$:RM1JOURNL)+EXECML$/LIB+LIB$:RMS/LIB
```

0321 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0322 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY