```
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
```

FILEID**RM2UPDDEL

```
RRRRRRRR    MM       MM    222222    UU       UU  PPPPPPPP    DDDDDDDD    DDDDDDDD    EEEEEEEEEE  LL
RRRRRRRR    MM       MM    222222    UU       UU  PPPPPPPP    DDDDDDDD    DDDDDDDD    EEEEEEEEEE  LL
RR      RR  MMMM   MMMM  22      22  UU       UU  PP      PP  DD      DD  DD      DD  EE          LL
RR      RR  MMMM   MMMM  22      22  UU       UU  PP      PP  DD      DD  DD      DD  EE          LL
RR      RR  MM MM MM MM        22  UU       UU  PP      PP  DD      DD  DD      DD  EE          LL
RR      RR  MM MM MM MM        22  UU       UU  PP      PP  DD      DD  DD      DD  EE          LL
RRRRRRRR    MM  MM  MM      22    UU       UU  PPPPPPPP    DD      DD  DD      DD  EEEEEEEE    LL
RRRRRRRR    MM  MM  MM      22    UU       UU  PPPPPPPP    DD      DD  DD      DD  EEEEEEEE    LL
RR  RR      MM      MM    22      UU       UU  PP          DD      DD  DD      DD  EE          LL
RR  RR      MM      MM    22      UU       UU  PP          DD      DD  DD      DD  EE          LL
RR    RR    MM      MM  22        UU       UU  PP          DD      DD  DD      DD  EE          LL
RR    RR    MM      MM  22        UU       UU  PP          DD      DD  DD      DD  EE          LL
RR      RR  MM      MM  2222222222  UUUUUUUUUU  PP          DDDDDDDD    DDDDDDDD    EEEEEEEEEE  LLLLLLLLLL
RR      RR  MM      MM  2222222222  UUUUUUUUUU  PP          DDDDDDDD    DDDDDDDD    EEEEEEEEEE  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
```

....
....
....

```
RM2
Sym

$$.
$$.
$$R
$$R
$$R
$$R
BDB
BDB
BDB
BDB
BDB
CJF
CJF
CJF
CLE
CLE
CLE
CLE
CSH
CSH
DLC
DLC
ERR
ERR
ERR
IFB
IFB
IFB
IFB
IFB
IFB
IRB
IRB
IRB
IRB
IRB
IRB
IRB
JNL
PIO
RAB
RAB
RAB
RJR
RJR
RJR
RJR
RJR
RJR
RJR
RJR
RJR
RJR
RJR
RLS
RMS
RMS
```

Table of contents

RM2
Pse

PSE

RM$
$AB

Pha

Ini
Com
Pas
Sym
Pas
Sym
Pse
Cro
Ass

The
659
The
431
26

Mac

-$2
-$2
-$2
TOT

142

The

MAC

```
0000     1          $BEGIN  RM2UPDDEL,000,RM$RMS2,<RELATIVE SPECIFIC $UPDATE AND $DELETE>
0000     2
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
```

```
0000     28 ;++
0000     29 ; Facility: rms32
0000     30 ;
0000     31 ; Abstract:
0000     32 ;                This module provides relative file organization
0000     33 ;                specific processing for the $update and $delete functions.
0000     34 ;
0000     35 ;
0000     36 ; Environment:
0000     37 ;                star processor running starlet exec.
0000     38 ;
0000     39 ; Author: L F Laverdure,          creation date: 8-NOV-1977
0000     40 ;
0000     41 ; Modified By:
0000     42 ;
0000     43 ;       V03-013 DAS0001         David Solomon           20-Jul-1983
0000     44 ;               Fix BI journaling of $DELETEs (include deleted record).
0000     45 ;
0000     46 ;       V03-012 KPL0003         Peter Lieberwirth       20-Jun-1983
0000     47 ;               Change some references to jnlflg to jnlflg2.
0000     48 ;
0000     49 ;       V03-011 KPL0002         Peter Lieberwirth       26-May-1983
0000     50 ;               Changes for new RJR format.
0000     51 ;
0000     52 ;       V03-010 RAS0137         Ron Schaefer            18-Mar-1983
0000     53 ;               Maybe this time I'll spell RJR$_DELETE correctly.
0000     54 ;
0000     55 ;       V03-009 RAS0135         Ron Schaefer            17-Mar-1983
0000     56 ;               Correct problems with RAS0132, registers and record sizes.
0000     57 ;
0000     58 ;       V03-008 RAS0132         Ron Schaefer            16-Mar-1983
0000     59 ;               Merge $RMSRDEF into $RJRDEF and revise the interface
0000     60 ;               for RM$WRTJNL for easier use from ISAM.
0000     61 ;
0000     62 ;       V03-007 TMK0001         Todd M. Katz            27-Dec-1982
0000     63 ;               Clear the bit IRB$V_FIND_LAST as soon as RM$UPDATE2 is entered.
0000     64 ;
0000     65 ;       V03-006 JWH0150         Jeffrey W. Horn         2-Dec-1982
0000     66 ;               Fix incorrect branch destination introduced in
0000     67 ;               JWH0112.
0000     68 ;
0000     69 ;       V03-005 KBT0423         Keith B. Thompson       1-Dec-1982
0000     70 ;               Fix broken branch because of ifab getting bigger
0000     71 ;
0000     72 ;       V03-004 KPL0001         Peter Lieberwirth       26-Oct-1982
0000     73 ;               Correct size of RJR overhead addred to R# for call
0000     74 ;               to WRTJNL.  Change RMSR names.
0000     75 ;
0000     76 ;       V03-003 JWH0112         Jeffrey W. Horn         06-Oct-1982
0000     77 ;               Implement new RJR format.  ALso put in support for
C000     78 ;               RU journaling.
0000     79 ;
0000     80 ;       V03-002 KBT0134         Keith B. Thompson       20-Aug-1982
0000     81 ;               Reorganize psects and fix revision number in jwh0001
0000     82 ;
0000     83 ;       V03-002 JWH0001         Jeffrey W. Horn         20-May-1982
0000     84 ;               Put in relative $DELETE journaling support.
```

```
0000     85 ;
0000     86 ;       V02-013 KPL0001         Peter Lieberwirth       22-Oct-1981
0000     87 ;               Call alternate lock routines QUERY_HARD and UNLOCK_HARD.
0000     88 ;               This maps an owner-held REA lock into a RNL error to prevent
0000     89 ;               updates or deletes on REA locked records.  (Because REA locks
0000     90 ;               on one record can be held by several streams.)
0000     91 ;
0000     92 ;       V02-012 REFORMAT        Maria del C. Nasr       24-Jul-1980
0000     93 ;
0000     94 ;       v011    CDS0044         C D Saether     31-MAR-1980     16:35
0000     95 ;               fix additional case that v008 missed
0000     96 ;
0000     97 ;       v010    CDS0043         C D Saether     20-OCT-1979     17:00
0000     98 ;               delete on shared seq file (looks like rel) is illegal iop
0000     99 ;
0000    100 ;       V009    JAK0020         J A Krycka      11-SEP-1979     10:00
0000    101 ;               Remove network code.
0000    102 ;
0000    103 ;       v008    CDS0012          C D Saether    26-JUN-1979     17:50
0000    104 ;               Fix bug that crashes if $delete first operation after $connect.
0000    105 ;
0000    106 ;--
0000    107 ;
0000    108
```

```
                     0000    110          .SBTTL  DECLARATIONS
                     0000    111
                     0000    112 ;
                     0000    113 ; Include Files:
                     0000    114 ;
                     0000    115
                     0000    116 ;
                     0000    117 ; Macros:
                     0000    118 ;
                     0000    119
                     0000    120          $IFBDEF
                     0000    121          $CSHDEF
                     0000    122          $DLCDEF
                     0000    123          $FABDEF
                     0000    124          $RABDEF
                     0000    125          $IRBDEF
                     0000    126          $BDBDEF
                     0000    127          $RMSDEF
                     0000    128          $CJFDEF
                     0000    129          $RJRDEF
                       00    130
                     0000    131 ;
                     0000    132 ; Equated Symbols:
                     0000    133 ;
                     0000    134
00000020             0000    135          ROP=RAB$L_ROP*8                 ; bit offset to rop field
                     0000    136
                     0000    137 ;
                     0000    138 ; Own Storage:
                     0000    139 ;
                     0000    140
```

```
0000    142          .SBTTL  RM$UPDATE2 - HIGH LEVEL RELATIVE $UPDATE
0000    143
0000    144 ;++
0000    145 ; RM$UPDATE2
0000    146 ;
0000    147 ; This routine performs the following functions:
0000    148 ;       1. calls updd1t2 subroutine to access the bucket and do record locking
0000    149 ;          as required
0000    150 ;       2. calls rm$putupd2 to move the updated record to the buffer
0000    151 ;       3. releases access to the bucket (causing it to be written unless
0000    152 ;          deferred write was specified at open time) and exits rms
0000    153 ;
0000    154 ; Calling sequence:
0000    155 ;
0000    156 ; entered via case br nch from rm$update at rm$update1.
0000    157 ;
0000    158 ; Input Parameters:
0000    159 ;
0000    160 ;       r11     impure area address
0000    161 ;       r10     ifab addr
0000    162 ;       r9      irab addr
0000    163 ;       r8      rab addr
0000    164 ;
0000    165 ; Implicit Inputs:
0000    166 ;
0000    167 ;       the contents of the rab and related irab and ifab.
0000    168 ;
0000    169 ; Output Parameters:
0000    170 ;
0000    171 ;       r7 thru r1      destroyed
0000    172 ;       r0              status
0000    173 ;
0000    174 ; Implicit Outputs:
0000    175 ;
0000    176 ;       various fields of the rab are filled in to reflect
0000    177 ;       the status of the operation (see functional spec
0000    178 ;       for details).
0000    179 ;
0000    180 ;       the irab is similarly updated.
0000    181 ;
0000    182 ; Completion Codes:
0000    183 ;
0000    184 ;       standard rms (see functional spec).
0000    185 ;
0000    186 ; Side Effects:
0000    187 ;
0000    188 ;       none
0000    189 ;
0000    190 ;--
0000    191
```

```
                                                                J 9
                         0000   193 RM$UPDATE2::
                         0000   194         $TSTPT  UPDATE2
                         0006   195         CSB     #IRB$V_FIND_LAST,(R9)   ; last operation no longer is $FIND
      00E0      30       000A   196         BSBW    UPDDLT2                 ; access bkt and do locking
     15 50      E9       000D   197         BLBC    R0,CLEAN                ; get out on error
10 A8  51       D0       0010   198         MOVL    R1,RAB$W_RFA(R8)        ; set rfa
     14 A8      B4       0014   199         CLRW    RAB$W_RFA+4(R8)         ; be neat
        5C      D4       0017   200         CLRL    AP                      ; flag to rm$putupd2 for $update
      FFE4'     30       0019   201         BSBW    RM$PUTUPD2              ; go move the record
     09 57      E9       001C   202         BLBC    R7,CLEAN1               ; get out on error
                         001F   203
     48 A9      D4       001F   204 RLSXIT: CLRL    IRB$L_RP(R9)            ; say no current record
      FFDB'     31       0022   205         BRW     RM$RLS2                 ; go release bucket
                         0025   206
                         0025   207 ;
                         0025   208 ;  handle error
                         0025   209 ;
                         0025   210
   57   50      D0       0025   211 CLEAN:  MOVL    R0,R7                   ; get status code to r7
      FFD5'     31       0028   212 CLEAN1: BRW     RM$CLN2_UPD             ; go clean up
```

```
002B    214          .SBTTL  RM$DELETE2 - HIGH LEVEL RELATIVE $DELETE
002B    215
002B    216 ;++
002B    217 ; RM$DELETE2
002B    218 ;
002B    219 ; This routine performs the following functions:
002B    220 ;;       1. calls upddlt2 subroutine to access the bucket and do record locking
002B    221 ;           as required
002B    222 ;        2. sets the record deleted flag in the control byte and declares the
002B    223 ;           buffer dirty.
002B    224 ;        3. releases access to the bucket (causing it to be written unless
002B    225 ;           deferred write was specified at open time) and exits rms
002B    226 ;
002B    227 ; Calling sequence:
002B    228 ;
002B    229 ; entered via case branch from rm$delete at rm$delete1.
002B    230 ;
002B    231 ; Input Parameters:
002B    232 ;
002B    233 ;        r11      impure area address
002B    234 ;        r10      ifab addr
002B    235 ;        r9       irab addr
002B    236 ;        r8       rab addr
002B    237 ;
002B    238 ; Implicit Inputs:
002B    239 ;
002B    240 ;        the contents of the rab and related irab and ifab.
002B    241 ;
002B    242 ; Output Parameters:
002B    243 ;
002B    244 ;        r7 thru r1       destroyed
002B    245 ;        r0               status
002B    246 ;
002B    247 ; Implicit Outputs:
002B    248 ;
002B    249 ;        various fields of the rab are filled in to reflect
002B    250 ;        the status of the operation (see functional spec
002B    251 ;        for details).
002B    252 ;
002B    253 ;        the irab is similarly deleted.
002B    254 ;
002B    255 ; Completion Codes:
002B    256 ;
002B    257 ;        standard rms (see functional spec).
002B    258 ;
002B    259 ; Side Effects:
002B    260 ;
002B    261 ;        none
002B    262 ;
002B    263 ;--
002B    264
```

```
                     002B   266 RM$DELETE2::
                     002B   267         $TSTPT  DELETE2
          03 6A 38   E1 0031   268         BBC     #IFB$V_SEQFIL,(R10),10$ ; continue if not sequential file
             00AF    31 0035   269         BRW     ERRIOP                  ; illegal operation for seq fil
             00B2    30 0038   270 10$:    BSBW    UPDDLT2                 ; access bucket and do locking
          57 50      D0 003B   271         MOVL    R0,R7                   ; copy status code
          03 50      E8 003E   272         BLBS    R0,JNL                  ; ok? continue
             FFBC'   31 0041   273 CLEAN2: BRW     RM$CLN2_DEL             ; go clean up
                     0044   274
                     0044   275 ;
                     0044   276 ;       Write out journal records, if needed
                     0044   277 ;
                     0044   278
          00A0 CA    95 0044   279 JNL:    TSTB    IFB$B_JNLFLG(R10)       ; journaling?
                03   12 0048   280         BNEQ    5$                      ; branch if yes.
                0090 31 004A   281         BRW     40$                     ; branch if not.
                30   BB 004D   282 5$:     PUSHR   #^M<R4,R5>              ; save R4,R5
       56 30 A9      D0 004F   283         MOVL    IRB$L_JNLBDB(R9),R6     ; get journaling BDB
       52 18 A6      D0 0053   284         MOVL    BDB$L_ADDR(R6),R2       ; get journaling buffer
       03 A2 02      90 0057   285         MOVB    #RJR$C_RECORD,RJR$B_ENTRY_TYPE(R2) ; RJR type
                     005B   286
                     005B   287         ASSUME  RJR$B_OPER      EQ      RJR$B_ORG+1
                     005B   288
          0501 8F    B0 005B   289         MOVW    #<RJR$_DELETE@8 + RJR$C_REL>,-
            04 A2           005F   290                 RJR$B_ORG(R2)           ; set file type and operation
    40 A2   48 A9    D0 0061   291         MOVL    IRB$L_RP(R9),RJR$L_RRN(R2) ; set relative record number
 14 A6   0048 8F    B0 0066   292         MOVW    #RJR$C_RECLEN,BDB$W_NUMB(R6)    ; set journal record length
          7E   55   7D 006C   293         MOVQ    R5,-(SP)                ; make type and jBDB arglist
 0C 00A0 CA 03      E1 006F   294         BBC     #IFB$V_AI,IFB$B_JNLFLG(R10),10$ ; branch if not AI journaling
          6E   03   9A 0075   295         MOVZBL  #CJF$_AI,(SP)           ; indicate AI journal
    00000000'EF     16 0078   296         JSB     RM$WRTJNL               ; go write record
             51 50  E9 007E   297         BLBC    R0,30$                  ; get out on error
                     0081   298
 24 00A2 CA 02      E1 0081   299 10$:    BBC     #IFB$V_RUP,IFB$B_JNLFLG2(R10),20$ ; branch if not RUP
       52   18 A6   D0 0087   300         MOVL    BDB$L_ADDR(R6),R2       ; get journaling buffer
    46 A2   62 A9    B0 008B   301         MOVW    IRB$W_CSIZ(R9),RJR$W_RSIZE(R2)  ; set cell size
 14 A6      62 A9    A0 0090   302         ADDW2   IRB$W_CSIZ(R9),BDB$W_NUMB(R6)   ; set journal record size
48 A2   65  62 A9    28 0095   303         MOVC3   IRB$W_CSIZ(R9),(R5),RJR$T_RIMAGE(R2) ; copy cell image
       54   08 AE    D0 009B   304         MOVL    8(SP),R4                ; restore BDB addr
          6E   01   9A 009F   305         MOVZBL  #CJF$_RU,(SP)           ; indicate RU journal
    0C000000'EF     16 00A2   306         JSB     RM$WRTJNL               ; go write record
             27 50  E9 00A8   307         BLBC    R0,30$                  ; get out on error
                     00AB   308
 21 00A0 CA 02      E1 00AB   309 20$:    BBC     #IFB$V_BI,IFB$B_JNLFLG(R10),30$ ; branch if not BI journaling
       52   18 A6   D0 00B1   310         MOVL    BDB$L_ADDR(R6),R2       ; get journaling buffer
    46 A2   62 A9    B0 00B5   311         MOVW    IRB$W_CSIZ(R9),RJR$W_RSIZE(R2)  ; set cell size
 14 A6      62 A9    A0 00BA   312         ADDW2   IRB$W_CSIZ(R9),BDB$W_NUMB(R6)   ; set journal record size
48 A2   65  62 A9    28 00BF   313         MOVC3   IRB$W_CSIZ(R9),(R5),RJR$T_RIMAGE(R2) ; copy cell image
       54   08 AE    D0 00C5   314         MOVL    8(SP),R4                ; restore BDB addr
          6E   02   9A 00C9   315         MOVZBL  #CJF$_BI,(SP)           ; indicate BI journal
    00000000'EF     16 00CC   316         JSB     RM$WRTJNL               ; go write record
                     00D2   317
          5E   08   C0 00D2   318 30$:    ADDL2   #8,SP                   ; discard arglist
               30   BA 00D5   319         POPR    #^M<R4,R5>              ; restore R4,R5
          57   50   D0 00D7   320         MOVL    R0,R7                   ; copy status code
            0D 50   E9 00DA   321         BLBC    R0,CLEAN3
                     00DD   322
```

```
                0C   90  00DD   323 40$:    MOVB    #DLC$M_REC!DLC$M_DELETED,-
                65          00DF   324              (R5)                    ; set delete flag
                03   88     00E0   325              BISB2   #BDB$M_VAL!BDB$M_DRT,-
            0A  A4          00E2   326              BDB$B_FLGS(R4)          ; say buffer dirty
              FF38   31     00E4   327              BRW     RLSXIT          ; go release bkt and exit
                             00E7   328
                             00E7   329 ;
                             00E7   330 ;  handle error
                             00E7   331 ;
                             00E7   332
              FF16'  31     00E7   333 ERRIOP: BRW     RM$ERRIOP               ; illegal for seq file
              FF13'  31     00EA   334 CLEAN3: BRW     RM$CLN2_DEL             ; go clean up
```

```
00ED    336          .SBTTL  UPDDLT2 - COMMON $UPDATE/$DELETE RELATIVE ROUTINE
00ED    337
00ED    338 ;++
00ED    339 ; UPDDLT2
00ED    340 ;
00ED    341 ;  This routine performs the following functions:
00ED    342 ;       1. checks that there is a current record.
00ED    343 ;       2. accesses the bucket with lock.
00ED    344 ;       3. verifies that the record is locked.
00ED    345 ;       4. unlocks the record unless manual locking is specified (rab$v_ulk = 1)
00ED    346 ;
00ED    347 ;  calling sequence:
00ED    348 ;
00ED    349 ;       bsbb    upddlt2
00ED    350 ;
00ED    351 ;  input parameters:
00ED    352 ;
00ED    353 ;       as for rm$update2
00ED    354 ;
00ED    355 ;  implicit inputs:
00ED    356 ;
00ED    357 ;  the fields of the various control blocks, including:
00ED    358 ;
00ED    359 ;       irb$l_rp        record #
00ED    360 ;       irb$l_curvbn    starting vbn for bucket
00ED    361 ;       irb$l_rp_off    offset to record in bucket
00ED    362 ;       rab$v_ulk       manual locking flag
00ED    363 ;
00ED    364 ;  output parameters:
00ED    365 ;
00ED    366 ;       r5              record address in buffer
00ED    367 ;       r4              bdb address
00ED    368 ;       r1              record #
00ED    369 ;       r0              status code
00ED    370 ;       r2_r3,ap        destroyed
00ED    371 ;
00ED    372 ;  implicit outputs:
00ED    373 ;
00ED    374 ;       none.
00ED    375 ;
00ED    376 ;  completion codes:
00ED    377 ;
00ED    378 ;       standard rms.
00ED    379 ;
00ED    380 ;  side effects:
00ED    381 ;
00ED    382 ;       may have switched to running at ast level.
00ED    383 ;--
00ED    384
```

```
                     00ED   386 UPDDLT2:
          54   D4    00ED   387          CLRL    R4                              ; initialize r4
51   44 A9   D0    00EF   388          MOVL    IRB$L_CURVBN(R9),R1             ; get current vbn
          3A   13    00F3   389          BEQL    ERRCUR                          ; error if none
                     00F5   390          $CSHFLAGS       LOCK                    ; get lock on bucket
52   5E AA   9A    00F8   391          MOVZBL  IFB$B_BKS(R10),R2               ; set up for readbkt2
        FF01'  30    00FC   392          BSBW    RM$READBKT2_UPD                 ; access the bucket
        24 50  E9    00FF   393          BLBC    R0,50$                          ; get out on error
51   48 A9   D0    0102   394          MOVL    IRB$L_RP(R9),R1                 ; get record #
          27   13    0106   395          BEQL    ERRCUR                          ; error if none
1A 6A     33   E0    0108   396          BBS     #IFB$V_NORECLK,(R10),50$; branch if no locking
09 69     2D   E5    010C   397          BBCC    #IRB$V_UNLOCK_RP,(R9),30$; clear auto unlock flag
                     0110   398                                                  ; if manual lock, don't unlock
                     0110   399
                     0110   400 ;
                     0110   401 ;  record locking required.  if record locked via automatic locking
                     0110   402 ;  (irb$v_unlock_rp = 1), unlock the record, giving an error if it was not
                     0110   403 ;  locked.  since the bucket is still locked, no other user can lock the
                     0110   404 ;  record before the bucket is released.
                     0110   405 ;
                     0110   406 ;  if manual locking (irb$v_unlock_rp = 0) need merely check that the
                     0110   407 ;  record is locked.
                     0110   408 ;
                     0110   409
          52   D4    0110   410          CLRL    R2                              ; clear high order rp
        FEEB'  30    0112   411          BSBW    RM$UNLOCK_HARD                  ; unlock record
        0F 50  E9    0115   412          BLBC    R0,ERRRNL                       ; branch if not locked
               05    0118   413          RSB                                     ; all set
        FEE4'  30    0119   414 30$:     BSBW    RM$QUERY_HARD                   ; is record locked?
8039 8F    50   B1    011C   415          CMPW    R0,#RMS$_OK_ALK&^XFFFF          ; well is it?
          04   12    0121   416          BNEQ    ERRRNL                          ; branch if not
                     0123   417          RMSSUC                                  ; vanilla success
               05    0126   418 50$:     RSB                                     ; all set
                     0127   419
                     0127   420 ;
                     0127   421 ;  handle errors
                     0127   422 ;
                     0127   423
                     0127   424 ERRRNL:
0C A8   000181A0 8F   D0    0127   425          MOVL    #RMS$_RNL,RAB$L_STV(R8) ; sub error code of record not
                     012F   426                                                  ;  locked, and fall thru to errcur
                     012F   427 ERRCUR:
                     012F   428          RMSERR  CUR                             ; no current record
               05    0134   429          RSB
                     0135   430
                     0135   431          .END
```

```
$$.PSECT_EP                     = 00000000
$$.TMP                          = 00000001
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
BDB$B_FLGS                      = 0000000A
BDB$L_ADDR                      = 00000018
BDB$M_DRT                       = 00000002
BDB$M_VAL                       = 00000001
BDB$W_NUMB                      = 00000014
CJF$_AI                         = 00000003
CJF$_BI                         = 00000002
CJF$_RU                         = 00000001
CLEAN                             00000025 R      01
CLEAN1                            00000028 R      01
CLEAN2                            00000041 R      01
CLEAN3                            000000EA R      01
CSH$M_LOCK                      = 00000001
CSH$M_NOBUFFER                  = 00000008
DLC$M_DELETED                   = 00000004
DLC$M_REC                       = 00000008
ERRCUR                            0000012F R      01
ERRIOP                            000000E7 R      01
ERRRNL                            00000127 R      01
IFB$B_BKS                       = 0000005E
IFB$B_JNLFLG                    = 000000A0
IFB$B_JNLFLG2                   = 000000A2
IFB$V_AI                        = 00000003
IFB$V_BI                        = 00000002
IFB$V_NORECLK                   = 00000033
IFB$V_RUP                       = 00000002
IFB$V_SEQFIL                    = 00000038
IRB$L_CURVBN                    = 00000044
IRB$L_JNLBDB                    = 00000030
IRB$L_RP                        = 00000048
IRB$V_FIND_LAST                 = 00000025
IRB$V_UNLOCK_RP                 = 0000002D
IRB$W_CSIZ                      = 00000062
JNL                               00000044 R      01
PIO$A_TRACE                       ********   X    01
RAB$L_ROP                       = 00000004
RAB$L_STV                       = 0000000C
RAB$W_RFA                       = 00000010
RJR$B_ENTRY_TYPE                = 00000003
RJR$B_OPER                      = 00000005
RJR$B_ORG                       = 00000004
RJR$C_RECLEN                    = 00000048
RJR$C_RECORD                    = 00000002
RJR$C_REL                       = 00000001
RJR$L_RRN                       = 00000040
RJR$T_RIMAGE                    = 00000048
RJR$W_RSIZE                     = 00000046
RJR$_DELETE                     = 00000005
RLSXIT                            0000001F R      01
RM$CLN2_DEL                       ********   X    01
RM$CLN2_UPD                       ********   X    01
RM$DELETE2                        0000002B RG     01
RM$ERRIOP                         ********   X    01
RM$PUTUPD2                        ********   X    01
RM$QUERY_HARD                     ********   X    01
RM$READBRT2_UPD                   ********   X    01
RM$RLS2                           ********   X    01
RM$UNLOCK_HARD                    ********   X    01
RM$UPDATE2                        00000000 RG     01
RM$WRTJNL                         ********   X    01
RMS$_CUR                        = 000184B4
RMS$_OK_ALK                     = 00018039
RMS$_RNL                        = 000181A0
ROP                             = 00000020
TPT$L_DELETE2                     ********   X    01
TPT$L_UPDATE2                     ********   X    01
UPDDLT2                           000000ED 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
RM$RMS2	00000135 (309.)	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	31	00:00:00.10	00:00:01.17
Command processing	145	00:00:00.74	00:00:07.20
Pass 1	341	00:00:11.63	00:00:24.80
Symbol table sort	0	00:00:01.73	00:00:02.19
Pass 2	86	00:00:02.27	00:00:04.95
Symbol table output	10	00:00:00.10	00:00:00.23
Psect synopsis output	1	00:00:00.05	00:00:00.24
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	616	00:00:16.62	00:00:40.78

The working set limit was 1350 pages.
65977 bytes (129 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1290 non-local and 9 local symbols.
431 source lines were read in Pass 1, producing 14 object records in Pass 2.
26 pages of virtual memory were used to define 25 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	16
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	21

1424 GETS were required to define 21 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:RM2UPDDEL/OBJ=OBJ$:RM2UPDDEL MSRC$:RM2UPDDEL/UPDATE=(ENH$:RM2UPDDEL)+EXECML$/LIB+LIB$:RMS/LIB
```

RM2UPDDEL
LIS

RM3CLOSE
LIS

RM3ALLBKT
LIS

RM2CREATE
LIS

RM2GET
LIS

RM2PUT
LIS

RM3BKTIO
LIS

RM3BKTSPL
LIS

RM3CMPKEY
LIS

RM3CMPRSS
LIS

RM2EXTEND
LIS

RM2FMTBKT
LIS

RM3BUG
LIS

RM2OPEN
LIS