```
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    222222    EEEEEEEEEE  XX      XX  TTTTTTTTTT  EEEEEEEEEE  NN      NN  DDDDDDDD
RRRRRRRR    MM      MM    222222    EEEEEEEEEE  XX      XX  TTTTTTTTTT  EEEEEEEEEE  NN      NN  DDDDDDDD
RR      RR  MMMM  MMMM  22      22  EE          XX      XX      TT      EE          NN      NN  DD      DD
RR      RR  MMMM  MMMM  22      22  EE          XX      XX      TT      EE          NN      NN  DD      DD
RR      RR  MM  MM  MM          22  EE            XX  XX        TT      EE          NNNN    NN  DD      DD
RR      RR  MM  MM  MM          22  EE            XX  XX        TT      EE          NNNN    NN  DD      DD
RRRRRRRR    MM      MM        22    EEEEEEEE        XX          TT      EEEEEEEE    NN  NN  NN  DD      DD
RRRRRRRR    MM      MM        22    EEEEEEEE        XX          TT      EEEEEEEE    NN  NN  NN  DD      DD
RR  RR      MM      MM      22      EE            XX  XX        TT      EE          NN    NNNN  DD      DD
RR  RR      MM      MM      22      EE            XX  XX        TT      EE          NN    NNNN  DD      DD
RR    RR    MM      MM    22        EE          XX      XX      TT      EE          NN      NN  DD      DD      ....
RR    RR    MM      MM    22        EE          XX      XX      TT      EE          NN      NN  DD      DD      ....
RR      RR  MM      MM  2222222222  EEEEEEEEEE  XX      XX      TT      EEEEEEEEEE  NN      NN  DDDDDDDD        ....
RR      RR  MM      MM  2222222222  EEEEEEEEEE  XX      XX      TT      EEEEEEEEEE  NN      NN  DDDDDDDD        ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                 '      SS
LL                        SS
LL                          SSSSSS
LL                          SSSSSS
LL                II                SS
LL                II                SS
LL                II                SS
LL                II                SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          $BEGIN  RM2EXTEND,000,RM$RMS2,<PROLOG LOCK AND UPDATE ROUTINES>
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    27 ;++
0000    28 ; Facility: RMS32
0000    29 ;
0000    30 ; Abstract:
0000    31 ;
0000    32 ;       This module provides two routines to handle the locking
0000    33 ;       and updating of file prologs on extends.
0000    34 ;
0000    35 ; Environment:
0000    36 ;               STAR PROCESSOR RUNNING STARLET EXEC.
0000    37 ;
0000    38 ;
0000    39 ; Author: L F Laverdure,          Creation date: 1-DEC-1977
0000    40 ;
0000    41 ; Modified By:
0000    42 ;
0000    43 ;       V03-006 JEJ0050         J E Johnson             24-Jul-1984
0000    44 ;               Fix broken error paths in RM$LOCK_PROLOG to keep from
0000    45 ;               releasing the same BDB/BLB twice on an error exit.
0000    46 ;
0000    47 ;       V03-005 RAS0284         Ron Schaefer            30-Mar-1984
0000    48 ;               Fix STV value on error paths for RMS$_RPL and RMS$_WPL errors.
0000    49 ;
0000    50 ;       V03-004 KBT0333         Keith B. Thompson       10-Sep-1982
0000    51 ;               Remove $FRBDEF
0000    52 ;
0000    53 ;       V03-003 KBT0220         Keith B. Thompson       23-Aug-1982
0000    54 ;               Reorganize psects
0000    55 ;
0000    56 ;       V03-002 KBT0118         Keith B. Thompson       6-Aug-1982
0000    57 ;               Remove ref. to set_sifb_adr
```

```
0000     58 ;
0000     59 ;       V03-001 KBT0092         Keith B. Thompson       13-Jul-1982
0000     60 ;               Clean up psects
0000     61 ;
0000     62 ;       V02-010 CDS0003         C Saether               10-Dec-1981
0000     63 ;               Fix broken branch.
0000     64 ;
0000     65 ;       V02-009 CDS0002         C. Saether              28-Aug-1981
0000     66 ;               Modifications for new cache/release routines.
0000     67 ;               Use RM$RLNER1 entry point now.
0000     68 ;
0000     69 ;       V02-008 CDS0001         C. Saether              17-Apr-1981
0000     70 ;               Fix error path on prologue write errors to not release
0000     71 ;               bucket twice.
0000     72 ;
0000     73 ;       V02-007 REFORMAT        Maria del C. Nasr       2-Aug-1980
0000     74 ;
0000     75 ;       V006    CDS0049         C Saether               31-OCT-1979     19:45
0000     76 ;               Lock BDB allocated at connect time now.  Remove code to
0000     77 ;               allocate and deallocate on the fly.
0000     78 ;
0000     79 ;       V005    CDS0025         C Saether               27-jul-1979     5:30
0000     80 ;               interlock on vbn -1 if sharing sequential file
0000     81 ;
0000     82 ;       V004    CDS0001         C D Saether             16-Mar-1979     16:10
0000     83 ;               get bcb on extend when shared, don't release prologue
0000     84 ;               twice on error
0000     85 ;
0000     86 ;       V003    RAN0003         R A Newell              9-Nov-1978      10:36
0000     87 ;               file sharing code enhancements
0000     88 ;--
0000     89 ;
0000     90
```

```
0000     92          .SBTTL  DECLARATIONS
0000     93
0000     94 ;
0000     95 ; Include Files:
0000     96 ;
0000     97
0000     98 ;
0000     99 ; Macros:
0000    100 ;
0000    101
0000    102          $BDBDEF
0000    103          $CSHDEF
0000    104          $FABDEF
0000    105          $IFBDEF
0000    106          $IRBDEF
0000    107          $PLGDEF
0000    108          $RLSDEF
0000    109          $RMSDEF
0000    110
0000    111 ;
0000    112 ; Equated Symbols:
0000    113 ;
0000    114
0000    115 ;
0000    116 ; Own Storage:
0000    117 ;
0000    118
```

```
0000    120          .SBTTL  RM$LOCK_PROLOG - ROUTINE TO LOCK PROLOG AND CHECK FOR EXTEND O.K.
0000    121
0000    122 ;++
0000    123 ; RM$LOCK_PROLOG
0000    124 ;
0000    125 ;  This routine causes the prolog (vbn 1) for a relative file
0000    126 ;  to be locked for an extend operation.
0000    127 ;  The prolog is read in, its checksum is verified, the noextend flag is checked
0000    128 ;  for being clear if this is a relative file, a lock bdb is allocated, and
0000    129 ;  the buffer for the prolog is released keeping vbn 1 locked.
0000    130 ;
0000    131 ; Calling sequence:
0000    132 ;
0000    133 ;       bsbw    rm$lock_prolog
0000    134 ;
0000    135 ; Input Parameters:
0000    136 ;
0000    137 ;       r11     impure area address
0000    138 ;       r10     ifab address
0000    139 ;       r9      ifab/irab address
0000    140 ;       r8      fab/rab address
0000    141 ;
0000    142 ; Implicit Inputs:
0000    143 ;
0000    144 ;       none
0000    145 ;
0000    146 ; Output Parameters:
0000    147 ;
0000    148 ;       r0              status code
0000    149 ;       r4              lock bdb address
0000    150 ;       r1-r3,r5,ap     destroyed
0000    151 ;
0000    152 ; Implicit Outputs:
0000    153 ;
0000    154 ;       none
0000    155 ;
0000    156 ; Completion Codes:
0000    157 ;
0000    158 ;       standard rms.
0000    159 ;
0000    160 ; Side Effects:
0000    161 ;
0000    162 ;       none
0000    163 ;
0000    164 ;--
0000    165
```

```
                     0000   167
                     0000   168 ;
                     0000   169 ;  This is not an entry point to this module.
                     0000   170 ;
                     0000   171
                     0000   172 SEQLCK:
 51    01    CE      0000   173          MNEGL   #1,R1                           ; -1 to r1
       52    D4      0003   174          CLRL    R2                              ; no size
                     0005   175          $CACHE  VBN=R1,SIZE=R2,-
                     0005   176                   FLAGS=<LOCK,NOREAD,NOBUFFER>
                     000B   177                                                  ; lock on vbn -1
             05      000B   178          RSB                                     ; and return
                     000C   179
                     000C   180 RM$LOCK_PROLOG::
 6A    38    E0      000C   181          BBS     #IFB$V_SEQFIL,(R10),-
       F0            000F   182                   SEQLCK                        ; branch if sequential file
                     0010   183          $CACHE  VBN=#1,SIZE=#512,-
                     0010   184                   FLAGS=LOCK,ERR=ERRRPE         ; read & lock prolog
    FFDC'    30      0021   185          BSBW    RM$CHKSUM                       ; verify checksum
   28 50     E9      0024   186          BLBC    R0,ERROR
                     0027   187          ASSUME  PLG$V_NOEXTEND EQ 0
07 10 A5     E9      0027   188          BLBC    PLG$B_FLAGS(R5),10$             ; extends allowed
                     002B   189          RMSERR  EXT                             ; extends not allowed
       1D    11      0030   190          BRB     ERROR                           ; and out
                     0032   191 10$:
 53    04    D0      0032   192          MOVL    #RLS$M_KEEP_LOCK,R3             ; specify flag for release
    FFC8'    30      0035   193          BSBW    RM$RELEASE                      ; release buffer for vbn 1
             05      0038   194          RSB
                     0039   195
                     0039   196 ;
                     0039   197 ;  error reading prolog  -  change error code
                     0039   198 ;
                     0039   199
                     0039   200 ERRRPE:
                0C A8    D5  0039   201          TSTL    FAB$L_STV(R8)           ; do we have an stv?
                  09     12  003C   202          BNEQ    10$                     ; okay use it
0C A8    50   00001000 8F    C9  003E   203          BISL3   #^X1000,R0,FAB$L_STV(R8); else set the RMS error there
                     0047   204 10$:    RMSERR  RPL
                  54     D4  004C   205          CLRL    R4                      ; don't release bdb/blb again in put.
                     05      004E   206          RSB
                     004F   207
                     004F   208 ;
                     004F   209 ;  handle checksum error by releasing lock on prolog
                     004F   210 ;
                     004F   211
                     004F   212 ERROR:
                  50     DD  004F   213          PUSHL   R0                      ; save status code
                FFAC'    30  0051   214          BSBW    RM$RLNER1               ; release lock
                  50  8ED0  0054   215          POPL    R0                      ; restore status
                  54     D4  0057   216          CLRL    R4                      ; don't release again in put
                     05      0059   217          RSB                             ; return to caller
                     005A   218
```

```
005A   220          .SBTTL  RM$UPD_PROLOG2 - ROUTINE TO UPDATE AND REWRITE PROLOG
005A   221
005A   222 ;++
005A   223 ;
005A   224 ; RM$UPD_PROLOG2
005A   225 ; RM$SETAEBK
005A   226 ; RM$RLSPLG
005A   227 ;
005A   228 ;  This routine causes the prolog (vbn 1) for a relative file to be updated
005A   229 ;  and rewritten to the file, releasing the lock on the prolog.
005A   230 ;  If the status code on entry to this routine indicates an error, the
005A   231 ;  noextend bit is set in the prolog indicating that the file cannot be extended.
005A   232 ;
005A   233 ; Calling sequence:
005A   234 ;
005A   235 ;       bsbw    rm$upd_prolog2
005A   236 ;
005A   237 ;  alternate entry at rm$rlsplg to merely release the prolog lock.
005A   238 ;
005A   239 ; Input Parameters:
005A   240 ;
005A   241 ;       r11     impure area address
005A   242 ;       r10     ifab address
005A   243 ;       r9      ifab/irab address
005A   244 ;       r8      fab/rab address
005A   245 ;       r6      new eof vbn (= high block + 1)
005A   246 ;       r0      status code
005A   247 ;
005A   248 ; Implicit Inputs:
005A   249 ;
005A   250 ;       It is assumed that the prolog is already locked.
005A   251 ;
005A   252 ; Output Parameters:
005A   253 ;
005A   254 ;       r0              status code
005A   255 ;       r1-r5,ap        destroyed
005A   256 ;
005A   257 ; Implicit Outputs:
005A   258 ;
005A   259 ;       IFB$L_HBK and IFB$L_EBK are set from (R6 - 1) and R6 respectively.
005A   260 ;       For sequential files being shared, IFB$L_EBK is updated from
005A   261 ;       IRB$L_CURVBN.
005A   262 ;       The lock BDB is deallocated.
005A   263 ;
005A   264 ; Completion Codes:
005A   265 ;
005A   266 ;       standard rms.
005A   267 ;
005A   268 ; Side Effects:
005A   269 ;
005A   270 ;       Any stream waiting on the prolog will be restarted.
005A   271 ;
005A   272 ;--
005A   273
```

```
                         005A   275 RM$UPD_PROLOG2::
                 50   DD  005A   276         PUSHL   R0                      ; save status
                         005C   277         $CACHE  VBN=#1,SIZE=#512,-
                         005C   278                  FLAGS=LOCK,ERR=ERRRPL   ; reread prolog blk 1
               FF90'  30  006D   279         BSBW    RM$CHKSUM               ; verify checksum
              2E 50   E9  0070   280         BLBC    R0,ERRRPL               ; branch if bad
              4C 6E   E9  0073   281         BLBC    (SP),ERREXT             ; branch if error on entry
          70 A5   56  D0  0076   282         MOVL    R6,PLG$L_EOF(R5)        ; set eof vbn
               FF83'  30  007A   283 MAKSUM: BSBW    RM$MAKSUM               ; recompute checksum
                 03   88  007D   284         BISB2   #BDB$M_DRT!BDB$M_VAL,-
              0A A4           007F   285                  BDB$B_FLGS(R4)     ; say buffer valid & dirty
             53   06  D0  0081   286         MOVL    #RLS$M_WRT_THRU!-
                         0084   287                  RLS$M_KEEP_LOCK,R3     ; cause immediate write
               FF79'  30  0084   288         BSBW    RM$RELEASE              ; write vbn 1
              47 50   E9  0087   289         BLBC    R0,ERRWPL               ; branch if error
                         008A   290
                         008A   291 ;
                         008A   292 ;  update eof vbn and highest allocated block
                         008A   293 ;
                         008A   294
                         008A   295 RM$SETHEBK::
      70 AA   56  01  C3 008A   296         SUBL3   #1,R6,IFB$L_HBK(R10)            ; update hi block
           0A 6A   38  E0 008F   297         BBS     #IFB$V_SEQFIL,(R10),SEQFL       ; branch if really seq
           74 AA   56  D0 0093   298         MOVL    R6,IFB$L_EBK(R10)               ; update eof block
                         0097   299
                         0097   300 ;
                         0097   301 ;  release lock on prolog
                         0097   302 ;
                         0097   303
               FF66'  30  0097   304 RLSPLG: BSBW    RM$RLNER1               ; free lock on prolog
                 01   BA  009A   305         POPR    #^M<R0>                 ; restore status code
                      05  009C   306         RSB                             ; return to caller
                         009D   307 SEQFL:
                 5A   DD  009D   308         PUSHL   R10                     ; save ifab addr for what i dont know
                 F6   11  009F   309         BRB     RLSPLG                  ; go release vbn -1
```

```
                                00A1   311
                                00A1   312 ;
                                00A1   313 ;  Handle prolog read error
                                00A1   314 ;  Prolog may still be locked.  Must release it.
                                00A1   315 ;
                                00A1   316
                                00A1   317 ERRRPL:
                                00A1   318          $CACHE  VBN=#1,SIZE=#0,-
                                00A1   319                   FLAGS=NOREAD              ; find prolog bdb
                      E8 6E   E9 00AC   320          BLBC    (SP),RLSPLG               ; branch if already had error
                      0C A8   D5 00AF   321          TSTL    FAB$L_STV(R8)             ; do we have an stv?
                         09   12 00B2   322          BNEQ    10$                       ; okay use it
    0C A8   6E  00001000 8F   C9 00B4   323          BISL3   #^X1000,(SP),FAB$L_STV(R8); else set the RMS error there
                                00BD   324 10$:      RMSERR  RPL,(SP)                  ; replace status code
                                00C2   325
                                00C2   326 ;
                                00C2   327 ;  Caller entered rm$upd_prolog2 with an error status.
                                00C2   328 ;  This must have been due to a bucket format write failure.
                                00C2   329 ;  Set the noextend bit in the prolog to prevent further attempts
                                00C2   330 ;  to extend the file.
                                00C2   331 ;
                                00C2   332
                                00C2   333 ERREXT:
                         01   88 00C2   334          BISB2   #1@PLG$V_NOEXTEND,-
                      10 A5      00C4   335                   PLG$B_FLAGS(R5)           ; set flag
                                00C6   336          RMSERR  EXT,(SP)                  ; update error code
                   56 74 AA   D0 00CB   337          MOVL    IFB$L_EBK(R10),R6         ; cause ebk not to be updated
                         A9   11 00CF   338          BRB     MAKSUM                    ; go rewrite prolog
                                00D1   339
                                00D1   340 ;
                                00D1   341 ;  Error occurred trying to rewrite the prolog.
                                00D1   342 ;  Update status code and return.  The bdb was already
                                00D1   343 ;  released in release due to the error.
                                00D1   344 ;
                                00D1   345
                                00D1   346 ERRWPL:
                      13 6E   E9 00D1   347          BLBC    (SP),20$                  ; branch if already had error
                      0C A8   D5 00D4   348          TSTL    FAB$L_STV(R8)             ; do we have an stv?
                         09   12 00D7   349          BNEQ    10$                       ; okay use it
    0C A8   6E  00001000 8F   C9 00D9   350          BISL3   #^X1000,(SP),FAB$L_STV(R8); else set the RMS error there
                                00E2   351 10$:      RMSERR  WPL,(SP)                  ; change status code
                         01   BA 00E7   352 20$:      POPR    #^M<R0>                   ; pop status off stack
                         05      00E9   353          RSB                               ; return
                                00EA   354
                                00EA   355 ;++
                                00EA   356 ;  Entry point to release the prolog.
                                00EA   357 ;
                                00EA   358 ;  inputs:
                                00EA   359 ;       r4      bdb address
                                00EA   360 ;       r0      status code
                                00EA   361 ;
                                00EA   362 ;  outputs:
                                00EA   363 ;       R1-R5, AP destroyed
                                00EA   364 ;
                                00EA   365 ;--
                                00EA   366
                                00EA   367 RM$RLSPLG::
```

```
                50   DD  00EA    368          PUSHL   R0                      ; push status
                A9   11  00EC    369          BRB     RLSPLG                  ; go do release
                         00EE    370
                         00EE    371
                         00EE    372          .END
```

```
$$.PSECT_EP                     =  00000000
$$.TMP                          =  00000004
$$RMSTEST                       =  0000001A
$$RMS_PBUGCHK                   =  00000010
$$RMS_TBUGCHK                   =  00000008
$$RMS_UMODE                     =  00000004
BDB$B_FLGS                      =  0000000A
BDB$M_DRT                       =  00000002
BDB$M_VAL                       =  00000001
CSH$M_LOCK                      =  00000001
CSH$M_NOBUFFER                  =  00000008
CSH$M_NOREAD                    =  00000004
ERREXT                             000000C2  R       01
ERROR                              0000004F  R       01
ERRRPE                             00000039  R       01
ERRRPL                             000000A1  R       01
ERRWPL                             000000D1  R       01
FAB$L_STV                       =  0000000C
IFB$L_EBK                       =  00000074
IFB$L_HBK                       =  00000070
IFB$V_SEQFIL                    =  00000038
MAKSUM                             0000007A  R       01
PLG$B_FLAGS                     =  00000010
PLG$L_EOF                       =  00000070
PLG$V_NOEXTEND                  =  00000000
RLS$M_KEEP_LOCK                 =  00000004
RLS$M_WRT_THRU                     00000002
RLSPLG                             00000097  R       01
RMS$CACHE                          ********    X     01
RMS$CHKSUM                         ********    X     01
RMS$LOCK_PROLOG                    0000000C  RG      01
RMS$MAKSUM                         ********    X     01
RMS$RELEASE                        ********    X     01
RMS$RLNER1                         ********    X     01
RMS$RLSPLG                         000000EA  RG      01
RMS$SETHEBK                        0000008A  RG      01
RMS$UPD_PROLOG2                    0000005A  RG      01
RMS$_EXT                        =  0001C022
RMS$_RPL                        =  0001C104
RMS$_WPL                        =  0001C11C
SEQFL                              0000009D  R       01
SEQLCK                             00000000  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
RMSRMS2	000000EE (238.)	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	00:00:00.08	00:00:01.04
Command processing	129	00:00:00.76	00:00:03.75
Pass 1	278	00:00:08.68	00:00:24.97
Symbol table sort	0	00:00:01.04	00:00:02.45
Pass 2	74	00:00:01.72	00:00:04.01
Symbol table output	6	00:00:00.08	00:00:00.41
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	521	00:00:12.38	00:00:36.68

The working set limit was 1350 pages.
46784 bytes (92 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 865 non-local and 5 local symbols.
372 source lines were read in Pass 1, producing 13 object records in Pass 2.
22 pages of virtual memory were used to define 21 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	13
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	17

997 GETS were required to define 17 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:RM2EXTEND/OBJ=OBJ$:RM2EXTEND MSRC$:RM2EXTEND/UPDATE=(ENH$:RM2EXTEND)+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