```
RRRRRRRRRRR      MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSS
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
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  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
```

```
RRRRRRRR    MM      MM    000000    RRRRRRRR     CCCCCCCC  LL          CCCCCCCC  KK      KK    222222
RRRRRRRR    MM      MM    000000    RRRRRRRR     CCCCCCCC  LL          CCCCCCCC  KK      KK    222222
RR      RR  MMMM  MMMM  OO      OO  RR      RR  CC        LL        CC          KK      KK  22      22
RR      RR  MMMM  MMMM  OO      OO  RR      RR  CC        LL        CC          KK      KK  22      22
RR      RR  MM  MM  MM  OO      OO  RR      RR  CC        LL        CC          KK    KK            22
RR      RR  MM  MM  MM  OO      OO  RR      RR  CC        LL        CC          KK    KK            22
RRRRRRRR    MM      MM  OO  OO  OO  RRRRRRRR    CC        LL        CC          KKKKK             22
RRRRRRRR    MM      MM  OO  OO  OO  RRRRRRRR    CC        LL        CC          KKKKK             22
RR  RR      MM      MM  OOOO    OO  RR  RR      CC        LL        CC          KK    KK        22
RR  RR      MM      MM  OOOO    OO  RR  RR      CC        LL        CC          KK    KK        22
RR    RR    MM      MM  OO      OO  RR    RR    CC        LL        CC          KK      KK    22          ....
RR    RR    MM      MM  OO      OO  RR    RR    CC        LL        CC          KK      KK    22          ....
RR      RR  MM      MM    000000    RR      RR    CCCCCCCC  LLLLLLLLLL  CCCCCCCC  KK      KK  2222222222  ....
RR      RR  MM      MM    000000    RR      RR    CCCCCCCC  LLLLLLLLLL  CCCCCCCC  KK      KK  2222222222  ....

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

```
;     1          0001  0 MODULE RMORCLCK2(
;     2          0002  0                 LANGUAGE (BLISS32),
;     3          0003  0                 IDENT = 'V04-000'
;     4          0004  0                 ) =
;     5          0005  1 BEGIN
;     6          0006  1
;     7          0007  1 !
;     8          0008  1 !**********************************************************************************
;     9          0009  1 !*                                                                                *
;    10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                       *
;    11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                         *
;    12          0012  1 !*  ALL RIGHTS RESERVED.                                                           *
;    13          0013  1 !*                                                                                *
;    14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED         *
;    15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE        *
;    16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER        *
;    17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY        *
;    18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY        *
;    19          0019  1 !*  TRANSFERRED.                                                                   *
;    20          0020  1 !*                                                                                *
;    21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE        *
;    22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT        *
;    23          0023  1 !*  CORPORATION.                                                                   *
;    24          0024  1 !*                                                                                *
;    25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS        *
;    26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                        *
;    27          0027  1 !*                                                                                *
;    28          0028  1 !*                                                                                *
;    29          0029  1 !**********************************************************************************
;    30          0030  1
;    31          0031  1 !++
;    32          0032  1 !
;    33          0033  1 ! FACILITY:
;    34          0034  1 !
;    35          0035  1 !       VAX-11 RMS
;    36          0036  1 !
;    37          0037  1 ! ABSTRACT:
;    38          0038  1 !
;    39          0039  1 !       This module contains record locking related routines, specifically
;    40          0040  1 !       timeout on record lock support.
;    41          0041  1 !
;    42          0042  1 ! ENVIRONMENT:
;    43          0043  1 !
;    44          0044  1 !       VAX/VMS Operating System, executive mode
;    45          0045  1 !--
;    46          0046  1 !
;    47          0047  1 ! AUTHOR: David Solomon, CREATION DATE: 16-Feb-1983
;    48          0048  1 !
;    49          0049  1 ! MODIFIED BY:
;    50          0050  1 !
;    51          0051  1 !       V03-004 DAS0001         David Solomon                   28-Feb-1984
;    52          0052  1 !               Use ISAM BUG_CHECK macro for bugcheck. Also, remove unnecessary
;    53          0053  1 !               code to set and clear PIO$V_INHAST.
;    54          0054  1 !
;    55          0055  1 !       V03-003 SHZ0002         Stephen H. Zalewski             26-Apr-1983
;    56          0056  1 !               Change PSECT attributes for this module.
;    57          0057  1 !
```

```
;    58          0058  1 !       V03-002 MCN0001         Maria del C. Nasr               22-Mar-1983
;    59          C059  1 !               Reorganize linkages
;    60          0060  1 !
;    61          0061  1 !       V03-001 SHZ0001         Stephen H. Zalewski             24-Feb-1983
;    62          0062  1 !               Changed name of module from rm0reclck2 to rm0rclck2 so that
;    63          0063  1 !               when linking we pick up the correct module name.
;    64          0064  1 !
;    65          0065  1 !--
```

```
;    67          0066  1 !
;    68          0067  1 ! INCLUDE FILES:
;    69          0068  1 !
;    70          0069  1
;    71          0070  1 LIBRARY
;    72          0071  1     'RMSLIB:RMSINTDEF'
;    73          0072  1     ;
;    74          0073  1
;    75          0074  1 LIBRARY
;    76          0075  1     'SYS$LIBRARY:STARLET'
;    77          0076  1     ;
;    78          0077  1
;    79          0078  1 SWITCHES
;    80          0079  1     ADDRESSING_MODE( EXTERNAL = GENERAL )
;    81          0080  1     ;
;    82          0081  1
;    83          0082  1 PSECT
;    84          0083  1     CODE = RM$RMS0(PSECT_ATTR),
;    85          0084  1     PLIT = RM$RMS0(PSECT_ATTR)
;    86          0085  1     ;
;    87          0086  1
;    88          0087  1 !
;    89          0088  1 ! MACROS:
;    90          0089  1 !
;    91          0090  1
;    92          0091  1 MACRO
;    93        M 0092  1     R_RLB =                                ! RLB address register declaration.
;    94          0093  1         RLB = 3 %
;    95        M 0094  1     ,R_RLB_STR =                           ! RLB structure declaration.
;    96          0095  1         R_RLB : REF BBLOCK %
;    97        M 0096  1     ,L_SET_LOCK_TMO =                      ! Linkage to RM$SET_LOCK_TMO.
;    98        M 0097  1         RL$SET_LOCK_TMO =
;    99        M 0098  1         JSB:
;   100        M 0099  1         GLOBAL( COMMON_RABREG, RLB = 3 )
;   101        M 0100  1         PRESERVE( 1, 2 )                   ! RM$LOCK needs R1 and R2.
;   102          0101  1         %
;   103          0102  1     ;
;   104          0103  1
;   105          0104  1 LINKAGE
;   106          0105  1     L_SET_LOCK_TMO,
;   107          0106  1     L_JSB;
;   108          0107  1
;   109          0108  1 !
;   110          0109  1 ! TABLE OF CONTENTS:
;   111          0110  1 !
;   112          0111  1
;   113          0112  1 FORWARD ROUTINE
;   114          0113  1     RM$SET_LOCK_TMO: RL$SET_LOCK_TMO       ! Set up timer AST for wait on lock.
;   115          0114  1     ,RM$LOCK_TMO_AST: NOVALUE              ! AST routine for lock timeout.
;   116          0115  1     ;
;   117          0116  1
;   118          0117  1 !
;   119          0118  1 ! EXTERNAL REFERENCES:
;   120          0119  1 !
;   121          0120  1
;   122          0121  1 EXTERNAL ROUTINE
;   123          0122  1     RM$CHKAST_ANY: RL$JSB ADDRESSING_MODE( WORD_RELATIVE )
```

```
;  124          0123  1
;  125          0124  1     ;

! Check for ASTs inhibited.
```

```
;  127     0125  1 %SBTTL 'RM$SET_LOCK_TMO'
;  128     0126  1 GLOBAL ROUTINE RM$SET_LOCK_TMO: RL$SET_LOCK_TMO =
;  129     0127  1
;  130     0128  1 !++
;  131     0129  1 !
;  132     0130  1 ! FUNCTIONAL DESCRIPTION:
;  133     0131  1 !
;  134     0132  1 !       This routine sets up a timer as a result of a record lock wait. It is
;  135     0133  1 !       called by RMORECLCK\DO_ENQ if the user specified ROP bits WAT and TMO.
;  136     0134  1 !
;  137     0135  1 ! CALLING SEQUENCE:
;  138     0136  1 !
;  139     0137  1 !       ret-status.wlc.v = RM$SET_LOCK_TMO()
;  140     0138  1 !
;  141     0139  1 ! FORMAL PARAMETERS:
;  142     0140  1 !
;  143     0141  1 !       NONE
;  144     0142  1 !
;  145     0143  1 ! IMPLICIT INPUTS:
;  146     0144  1 !
;  147     0145  1 !       R3              Address of RLB.
;  148     0146  1 !
;  149     0147  1 ! IMPLICIT OUTPUTS:
;  150     0148  1 !
;  151     0149  1 !       NONE
;  152     0150  1 !
;  153     0151  1 ! ROUTINE VALUE:
;  154     0152  1 !
;  155     0153  1 !       Return status from $SETIMR.
;  156     0154  1 !
;  157     0155  1 ! SIDE EFFECTS:
;  158     0156  1 !
;  159     0157  1 !       A timer AST is declared.
;  160     0158  1 !--
;  161     0159  1
;  162     0160  2 BEGIN
;  163     0161  2
;  164     0162  2 LOCAL
;  165     0163  2     TIMEOUT: VECTOR[2, LONG]            ! 64-bit delta time
;  166     0164  2     ,STATUS                             ! Holds $SETIMR status.
;  167     0165  2     ;
;  168     0166  2
;  169     0167  2 EXTERNAL REGISTER
;  170     0168  2     R_RLB_STR                           ! RLB.
;  171     0169  2     ;
;  172     0170  2
;  173     0171  2 !+
;  174     0172  2 ! Convert timeout value from seconds to a 64-bit delta time.
;  175     0173  2 !
;  176     0174  2 ! Details: delta times are negative, hence the upper longword is %XFFFFFFFF.
;  177     0175  2 ! The short cut used to convert the seconds to 100-nanosecond units is
;  178     0176  2 ! acceptable here because the number of 100-nanosecond units one can express
;  179     0177  2 ! in a longword (429 seconds) is greater than the number of 100-nanosecond units
;  180     0178  2 ! one can express in the number of seconds that will fit in a byte (255).
;  181     0179  2 !-
;  182     0180  2
;  183     0181  2 IF .RLB[RLB$B_TMO] NEQU 0               ! If the timeout value is non-zero,
```

```
;   184          0182  2 THEN                                    ! then convert to delta time.
;   185          0183  3     BEGIN
;   186          0184  3
;   187          0185  3     TIMEOUT[0] = .RLB[RLB$B_TMO] * -10000000;
;   188          0186  3                                         ! Convert seconds to 100-nanoseconds.
;   189          0187  3
;   190          0188  3     TIMEOUT[1] = -1;                    ! Upper longword is negative.
;   191          0189  3
;   192          0190  3     END
;   193          0191  2 ELSE                                    ! Else special case zero timeout.
;   194          0192  3     BEGIN
;   195          0193  3
;   196          0194  3     TIMEOUT[0] = 0;
;   197          0195  3
;   198          0196  3     TIMEOUT[1] = 0;
;   199          0197  3
;   200          0198  2     END;
;   201          0199  2
;   202          0200  2 !+
;   203          0201  2 ! Setup timer request.
;   204          0202  2 !-
;   205          0203  2
;   206        P 0204  2 STATUS = $SETIMR(                       ! Value of routine is $SETIMR status.
;   207        P 0205  2                  DAYTIM = TIMEOUT       ! Address of 64-bit delta time.
;   208        P 0206  2                  ,ASTADR = RM$LOCK_TMO_AST
;   209        P 0207  2                                         ! Address of AST routine for timeout.
;   210        P 0208  2                  ,REQIDT = .RLB         ! ASTPRM (address of RLB).
;   211          0209  2                  );
;   212          0210  2
;   213          0211  2 IF .STATUS                                ! If successful,
;   214          0212  2 THEN                                      ! then
;   215          0213  2     RLB[RLB$V_TIMER_INPROG] = 1;          ! set timeout in progress flag.
;   216          0214  2
;   217          0215  2 RETURN .STATUS;                           ! Return $SETIMR status.
;   218          0216  2
;   219          0217  1 END;                                      ! End of routine RM$SET_LOCK_TMO


                                                    .TITLE  RMORCLCK2
                                                    .IDENT  \V04-000\

                                                    .EXTRN  RM$CHKAST_ANY, SYS$SETIMR

                                                    .PSECT  RM$RMS0,NOWRT,  GBL,  PIC,2

                                 51  DD 00000 RM$SET_LOCK_TMO::
                                                    PUSHL   R1                                          ; 0126
                              5E                 08  C2 00002                SUBL2   #8, SP                      ;
                                              0A    A3  95 00005             TSTB    10(RLB)                     ; 0181
                                                    12  13 00008             BEQL    1$                          ;
                              50                0A  A3  9A 0000A             MOVZBL  10(RLB), R0                 ; 0185
                        6E    50 FF676980         8F  C5 0000E             MULL3   #-10000000, R0, TIMEOUT     ;
                                          04    AE  01  CE 00016             MNEGL   #1, TIMEOUT+4               ; 0188
                                                    02  11 0001A             BRB     2$                          ; 0181
                                                    6E  7C 0001C 1$:         CLRQ    TIMEOUT                     ; 0194
                                                    53  DD 0001E 2$:         PUSHL   RLB                         ; 0209
                                          0000V     CF  9F 00020             PUSHAB  RM$LOCK_TMO_AST             ;
```

```
                                 08    AE  9F 00024          PUSHAB  TIMEOUT                                ;
                                       7E  D4 00027          CLRL    -(SP)                               ;
                          00000000G    00  FB 00029          CALLS   #4, SYS$SETIMR                      ;
                                       04  50 E9 00030          BLBC    STATUS, 3$                      ; 0211
                                 04    A3  01 88 00033          BISB2   #1, 4(RLB)                      ; 0213
                                       5E  08 C0 00037 3$:      ADDL2   #8, SP                          ; 0217
                                           02 BA 0003A          POPR    #^M<R1>                         ;
                                           05 0003C             RSB                                     ;

; Routine Size:  61 bytes,    Routine Base:  RM$RMS0 + 0000
```

```
;  221     0218  1 %SBTTL 'RM$LOCK_TMO_AST'
;  222     0219  1 GLOBAL ROUTINE RM$LOCK_TMO_AST( RLB: REF BBLOCK ): NOVALUE =
;  223     0220  1
;  224     0221  1 !++
;  225     0222  1 !
;  226     0223  1 ! FUNCTIONAL DESCRIPTION:
;  227     0224  1 !
;  228     0225  1 !       This is the AST routine that fires when the timer for a lock
;  229     0226  1 !       request expires. It is declared by a $SETIMR in RM$SET_LOCK_TMO.
;  230     0227  1 !
;  231     0228  1 ! CALLING SEQUENCE:
;  232     0229  1 !
;  233     0230  1 !       ret-status.wlc.v = RM$LOCK_TMO_AST( astprm.rlu.v )
;  234     0231  1 !
;  235     0232  1 ! FORMAL PARAMETERS:
;  236     0233  1 !
;  237     0234  1 !       ASTPRM          RLB address.
;  238     0235  1 !
;  239     0236  1 ! IMPLICIT INPUTS:
;  240     0237  1 !
;  241     0238  1 !       NONE
;  242     0239  1 !
;  243     0240  1 ! IMPLICIT OUTPUTS:
;  244     0241  1 !
;  245     0242  1 !       NONE
;  246     0243  1 !
;  247     0244  1 ! ROUTINE VALUE:
;  248     0245  1 !
;  249     0246  1 !       NONE
;  250     0247  1 !
;  251     0248  1 ! SIDE EFFECTS:
;  252     0249  1 !
;  253     0250  1 !       Lock request is dequeued.
;  254     0251  1 !--
;  255     0252  1
;  256     0253  2 BEGIN
;  257     0254  2
;  258     0255  2 BUILTIN
;  259     0256  2     TESTBITSC
;  260     0257  2     ;
;  261     0258  2
;  262     0259  2 !+
;  263     0260  2 ! Validate ASTPRM (RLB structure).
;  264     0261  2 !-
;  265     0262  2
;  266     0263  2 !** should we probe RLB?
;  267     0264  2 !** should we check RLB BID/BLN?
;  268     0265  3 IF ( .RLB[RLB$B_BID] NEQU RLB$C_BID ) OR ( .RLB[RLB$B_BLN] NEQU RLB$K_BLN/4 )
;  269     0266  2 THEN
;  270     0267  2     BUG_CHECK;
;  271     0268  2
;  272     0269  2 !+
;  273     0270  2 ! Check if ASTs should be inhibited; if so, requeue this AST and do a RET
;  274     0271  2 ! (e.g. the routine won't return). If ASTs are not inhibited, simply RSB
;  275     0272  2 ! back here.
;  276     0273  2 !-
;  277     0274  2
```

```
;  278          0275  2 RM$CHKAST_ANY();                        ! Check PIO$V_INHAST.
;  279          0276  2
;  280          0277  2 !+
;  281          0278  2 ! If the $ENQ hasn't completed, then clear the timer in progress flag and
;  282          0279  2 ! dequeue (cancel) the lock request.
;  283          0280  2 !
;  284          0281  2 ! If the timer fired immediately after the $ENQ completion, but before the
;  285          0282  2 ! timer could be cancelled, then simply return; we have nothing to do.
;  286          0283  2 !-
;  287          0284  2
;  288          0285  2 IF TESTBITSC( RLB[RLB$V_TIMER_INPROG] ) ! If the $ENQ hasn't completed,
;  289          0286  2 THEN                                    ! then dequeue the lock request.
;  290          0287  3     BEGIN
;  291          0288  3
;  292          0289  3     LOCAL
;  293          0290  3         STATUS
;  294          0291  3         ;
;  295          0292  3
;  296          0293  3     STATUS = $DEQ( LKID = .RLB[RLB$L_LOCK_ID] );
;  297          0294  3                                         ! Dequeue (cancel) the request.
;  298          0295  3
;  299          0296  3     !+
;  300          0297  3     ! The only acceptable failure status from $DEQ should be SS$_IVLOCKID;
;  301          0298  3     ! this could only happen if the timer fired immediately after the $ENQ
;  302          0299  3     ! completed, but before the RLB$V_TIMER_INPROG bit could be cleared by
;  303          0300  3     ! routine RMORECLCK\DO_ENQ.
;  304          0301  3     !-
;  305          0302  3
;  306          0303  4     IF ( NOT .STATUS ) AND ( .STATUS NEQ SS$_IVLOCKID )
;  307          0304  4                                         ! If the $DEQ failed for other than
;  308          0305  4     THEN                                ! an invalid lock id, then
;  309          0306  4         BUG_CHECK;                      ! bugcheck.
;  310          0307  3
;  311          0308  2     END;
;  312          0309  2
;  313          0310  1 END;                                    ! End of routine RM$LOCK_TMO_AST


                                                        .EXTRN  RM$BUG3, SYS$DEQ

                                        0FFC 00000      .ENTRY  RM$LOCK_TMO_AST, Save R2,R3,R4,R5,R6,R7,R8,-; 0219
                                                                R9,R10,R11                                    ;
                                53 00000000G   00  9E 00002     MOVAB   RM$BUG3, R3                          ;
                                52       04    AC  D0 00009     MOVL    RLB, R2                              ; 0265
                                0E       08    A2  91 0000D     CMPB    8(R2), #14                           ;
                                               06  12 00011     BNEQ    1$                                   ;
                                07       09    A2  91 00013     CMPB    9(R2), #7                            ;
                                               02  13 00017     BEQL    2$                                   ;
                                               63  16 00019 1$:  JSB     RM$BUG3                              ; 0266
                                           0000G   30 0001B 2$:  BSBW    RM$CHKAST_ANY                        ; 0275
                        1C       04    A2          00  E5 0001E     BBCC    #0, 4(R2), 3$                        ; 0285
                                               7E  7C 00023     CLRQ    -(SP)                                ; 0293
                                               7E  D4 00025     CLRL    -(SP)                                ;
                                           18  A2  DD 00027     PUSHL   24(R2)                               ;
                    00000000G  00              04  FB 0002A     CALLS   #4, SYS$DEQ                          ;
                               0B              50  E8 00031     BLBS    STATUS, 3$                           ; 0303
```

```
                 00002124   8F            50  D1 00034          CMPL    STATUS, #8484                          ;
                                            02  13 0003B          BEQL    3$                                     ;
                                            63  16 0003D          JSB     RM$BUG3                                ; 0305
                                                04 0003F 3$:      RET                                            ; 0310

; Routine Size:  64 bytes,    Routine Base:  RM$RMS0 + 003D

;  314          0311  1
;  315          0312  1 END
;  316          0313  0 ELUDOM                                  ! End of module RMORCLCK2




;                                  PSECT SUMMARY

;       Name                    Bytes                           Attributes

;  RM$RMS0                       125  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  GBL,  REL,  CON,  PIC,ALIGN(2)



;                                Library Statistics

;                                          -------- Symbols --------      Pages       Processing
;       File                               Total    Loaded   Percent      Mapped      Time

;  _$255$DUA28:[RMS.OBJ]RMSINTDEF.L32;1      1484        16         1          83       00:00.3
;  _$255$DUA28:[SYSLIB]STARLET.L32;1         9776         5         0         581       00:02.4




;                                    COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RMORCLCK2/OBJ=OBJ$:RMORCLCK2 MSRC$:RMORCLCK2/UPDATE=(ENH$:RMORCLCK2)

; Size:          125 code + 0 data bytes
; Run Time:          00:07.4
; Elapsed Time:      00:22.6
; Lines/CPU Min:      2530
; Lexemes/CPU-Min:    9113
; Memory Used:  51 pages
; Compilation Complete
```

RM0RECLCK
LIS

RM0SET
LIS

RM0SCAN
LIS

RM0PRFLNM
LIS

RM0RCLCK2
LIS

RM0RABCHK
LIS

RM0RELEAS
LIS

RM0NAMSTR
LIS