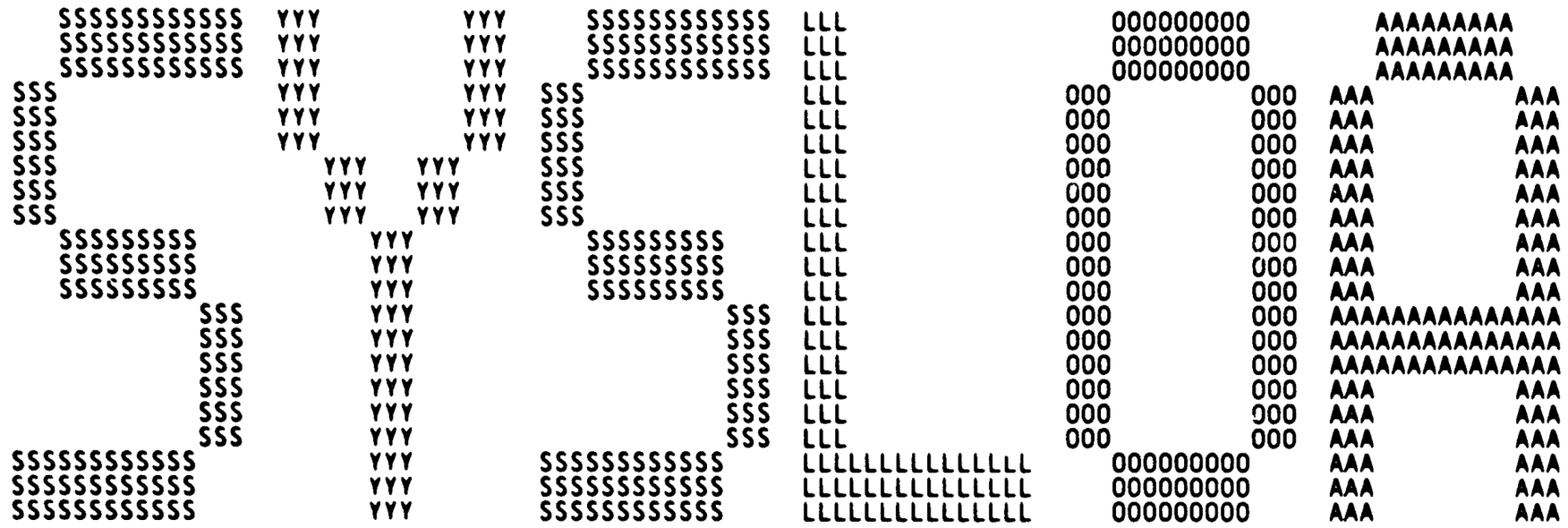
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA

**FILE**ID**REBLDLOCK

```
RRRRRRRR    EEEEEEEEEE  BBBBBBBB    LL          DDDDDDDD    LL            OOOOOO      CCCCCCCC  KK      KK
RRRRRRRR    EEEEEEEEEE  BBBBBBBB    LL          DDDDDDDD    LL            OOOOOO      CCCCCCCC  KK      KK
RR      RR  EE          BB      BB  LL          DD      DD  LL          OO      OO  CC          KK      KK
RR      RR  EE          BB      BB  LL          DD      DD  LL          OO      OO  CC          KK      KK
RR      RR  EE          BB      BB  LL          DD      DD  LL          OO      OO  CC          KK    KK
RR      RR  EE          BB      BB  LL          DD      DD  LL          OO      OO  CC          KK    KK
RRRRRRRR    EEEEEEEE    BBBBBBBB    LL          DD      DD  LL          OO      OO  CC          KKKKKK
RRRRRRRR    EEEEEEEE    BBBBBBBB    LL          DD      DD  LL          OO      OO  CC          KKKKKK
RR  RR      EE          BB      3B  LL          DD      DD  LL          OO      OO  CC          KK    KK
RR  RR      EE          BB      BB  LL          DD      DD  LL          OO      OO  CC          KK    KK
RR    RR    EE          BB      BB  LL          DD      DD  LL          OO      OO  CC          KK      KK      ....
RR    RR    EE          BB      BB  LL          DD      DD  LL          OO      OO  CC          KK      KK      ....
RR      RR  EEEEEEEEEE  BBBBBBBB    LLLLLLLLLL  DDDDDDDD    LLLLLLLLLL    OOOOOO      CCCCCCCC  KK      KK      ....
RR      RR  EEEEEEEEEE  BBBBBBBB    LLLLLLLLLL  DDDDDDDD    LLLLLLLLLL    OOOOOO      CCCCCCCC  KK      KK      ....

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     1          .TITLE  REBLDLOCK - Rebuild Lock Database on Failover
0000     2          .IDENT  'V04-000'
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ;
0000    31 ; FACILITY:     Executive, system services and fork level code
0000    32 ;
0000    33 ; ABSTRACT:
0000    34 ;       This module contains routines used to rebuild the lock database
0000    35 ;       when a system is removed from the cluster.
0000    36 ;
0000    37 ; ENVIRONMENT:  Kernel mode, fork level, loadable code
0000    38 ;
0000    39 ;--
0000    40 ;
0000    41 ; AUTHOR: Steve Beckhardt,       CREATION DATE: 25-May-1983
0000    42 ;
0000    43 ; MODIFIED BY:
0000    44 ;
0000    45 ;       V03-015 SRB0135         Steve Beckhardt         6-Jul-1984
0000    46 ;               Zero deadlock bitmap expiration timestamps on every
0000    47 ;               state change.
0000    48 ;
0000    49 ;       V03-014 SRB0134         Steve Beckhardt         22-Jun-1984
0000    50 ;               Fixed bugs in lock rebuilding.  1)  Put all locks
0000    51 ;               in response states (RSP...) into RETRY state and
0000    52 ;               2) store newly computed group grant mode in all resources,
0000    53 ;               regardless of where the resource is mastered.
0000    54 ;
0000    55 ;       V03-013 SRB0132         Steve Beckhardt         25-May-1984
0000    56 ;               Added support for LCK$M_NODLCKWT flag.
0000    57 ;
```

```
0000     58 ;       V03-012 SRB0121         Steve Beckhardt         29-Apr-1984
0000     59 ;               Added more integrity checks.
0000     60 ;
0000     61 ;       V03-011 SRB0117         Steve Beckhardt         10-Mar-1984
0000     62 ;               Added code to remove RSBs from time out queue
0000     63 ;               during failover.
0000     64 ;
0000     65 ;       V03-010 SRB0115         Steve Beckhardt         24-Feb-1984
0000     66 ;               Added support for distributed deadlock detection.
0000     67 ;
0000     68 ;       V03-008 SRB0110         Steve Beckhardt         27-Jan-1984
0000     69 ;               Added MEMSEQ checking for REBUILD messages.  Added
0000     70 ;               PMS counters.  Fixed RSPDOLOCL bug in DISPATCH.
0000     71 ;               Added routines to set rebuild state.  Fixed CSID_VALID bug.
0000     72 ;
0000     73 ;       V03-007 SRB0108         Steve Beckhardt         11-Jan-1984
0000     74 ;               Redesigned rebuilding lock database to remaster all trees
0000     75 ;               and to support distributed root directory.  Added support
0000     76 ;               for request sequence numbering on failover and for
0000     77 ;               maintaining the EPID in locks.
0000     78 ;
0000     79 ;       V03-006 SRB0106         Steve Beckhardt         6-Dec-1983
0000     80 ;               Changed LKB$L_REFCNT, RSB$L_REFCNT, and RSB$L_BLKASTCNT
0000     81 ;               to be word fields.
0000     82 ;
0000     83 ;       V03-005 SRB0104         Steve Beckhardt         17-Oct-1983
0000     84 ;               Fixed bug where second quadword of value block was lost
0000     85 ;               when rebuilding locks.
0000     86 ;
0000     87 ;       V03-004 SRB0100         Steve Beckhardt         29-Jul-1983
0000     88 ;               Changed interface to failover routines
0000     89 ;
0000     90 ;       V03-003 SRB0094         Steve Beckhardt         23-Jun-1983
0000     91 ;               Continued adding support for n-node failover.
0000     92 ;
0000     93 ;       V03-002 SRB0093         Steve Beckhardt         3-Jun-1983
0000     94 ;               Removed spurious test and resultant BUG_CHECK.
0000     95 ;
```

```
                        0000     97          .SBTTL  DECLARATIONS
                        0000     98 ;
                        0000     99 ; INCLUDE FILES:
                        0000    100 ;
                        0000    101
                        0000    102 ;
                        0000    103 ; MACROS:
                        0000    104 ;
                        0000    105
                        0000    106          $ACBDEF                         ; ACB offsets
                        0000    107          $CADEF                          ; Conditional assembly switches
                        0000    108          $CDRPDEF                        ; CDRP offsets
                        0000    109          $CLSMSGDEF                      ; Cluster message offsets
                        0000    110          $CLUBDEF                        ; CLUB offsets
                        0000    111          $CSBDEF                         ; CSB offsets
                        0000    112          $DYNDEF                         ; Data structure names
                        0000    113          $FKBDEF                         ; Fork block offsets
                        0000    114          $IPLDEF                         ; IPL definitions
                        0000    115          $LCKDEF                         ; LCK definitions
                        0000    116          $LKBDEF                         ; LKB offsets
                        0000    117          $PCBDEF                         ; PCB offsets
                        0000    118          $PRIDEF                         ; Priority definitions
                        0000    119          $PSLDEF                         ; PSL definitions
                        0000    120          $RSBDEF                         ; RSB offsets
                        0000    121          $RSNDEF                         ; Resource numbers
                        0000    122
                        0000    123 ;
                        0000    124 ; EQUATED SYMBOLS:
                        0000    125 ;
                        0000    126
                        0000    127
                        0000    128 ;
                        0000    129 ; OWN STORAGE:
                        0000    130 ;
                        0000    131
                    00000000    132          .PSECT  $$$040,LONG
                        0000    133
                        0000    134          .ALIGN  LONG
                        0000    135
                        0000    136 CURR_LOCKID:                            ; Current lockid
                    00000000    137          .LONG   0
                        0004    138 RETURN_ADDR:                            ; Return address from failover routines
                    00000000    139          .LONG   0
                        0008    140 LCK$GL_TS_CSID::                        ; CSID of system issuing timestamps
                    00000000    141          .LONG   0                      ; (0= this system)
                        000C    142
                    00000000    143          .PSECT  $$$020
                        0000    144
                        0000    145 ;****************************************************************************
                        0000    146 ;
                        0000    147 ; NOTE: The following assumption is in effect for this entire module.
                        0000    148 ;
                        0000    149 ;****************************************************************************
                        0000    150
                        0000    151          ASSUME  IPL$_SYNCH  EQ  IPL$_SCS
```

```
                                        .SBTTL  LCK$INIT_REBUILD - Initialize lock database for rebuild

                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       This routine initializes the lock database for the rebuild
                                ;       operation.  It does the following:
                                ;
                                ;               o  Removes all LKBs and RSBs from time out queue and
                                ;                  deallocates RSBs
                                ;               o  Clears all LKB$M_RESEND bits
                                ;               o  Removes all master copy locks
                                ;               o  Changes all locally issued locks in transient SCS states
                                ;                  to RETRY state
                                ;               o  Clears all RSB$M_DIRENTRY flags
                                ;               o  Sets all RSB$L_CSID fields to an illegal CSID
                                ;               o  Removes all directory entries
                                ;               o  Selects a node to issue timestamps (for deadlock detection)
                                ;                  and resets expiration timestamps
                                ;
                                ;       The result is a lock database with strictly local information;
                                ;       no master copies and no directory entries.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       JSB     LCK$INIT_REBUILD (called from failover table dispatcher)
                                ;       IPL must be at IPL$_SYNCH
                                ;
                                ; INPUT ARGUMENTS:
                                ;
                                ;       None
                                ;
                                ; OUTPUT ARGUMENTS:
                                ;
                                ;       None
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       R0 - R5 not preserved
                                ;--

                                LCK$INIT_REBUILD::
            0FE0 8F     BB      PUSHR   #^M<R5,R6,R7,R8,R9,R10,R11>

                                ; Remove all locks (and RSBs) from the timeout queue.  RSBs represent
                                ; calls to LCK$SND_RMVDIR that failed due to insufficient pool.
                                ; These should be deallocated.

    55  00000000'GF    DE      MOVAL   G^LCK$GL_TIMOUTQ,R5     ; Get address of queue header
      56    00 B5     0F      REMQUE  @(R5),R6                ; Remove LKB (or RSB)
              17      1D      BVS     8$                      ; Queue is empty
            0040 8F   AA      BICW    #LKB$M_TIMOUTQ,-        ; Clear status bit
              2A A6                   LKB$W_STATUS(R6)
              0A A6   91      CMPB    LKB$B_TYPE(R6),-        ; Is it a RSB?
                      36              #DYN$C_RSB
                      EE      BNEQ    5$                      ; No
            50    56  D0      MOVL    R6,R0                   ; Yes deallocate it
```

```
                00000000'GF   16  0020   210          JSB     G^EXE$DEANONPAGED
                          E3  11  0026   211          BRB     5$
                              0028   212
                              0028   213 ; Loop through the lock id. table to remove all master copy locks.
                              0028   214 ; If a lock we wish to delete has a non-zero reference count then
                              0028   215 ; it is deferred until later.  Conversly, whenever we delete a lock
                              0028   216 ; we also try to delete its parent lock.
                              0028   217
                          5B  D4  0028   218 8$:      CLRL    R11                     ; Initialize lock id.
              5A  00000000'GF   D0  002A   219          MOVL    G^LCK$GL_IDTBL,R10      ; Get address of lock id. table
                              0031   220
                              0031   221 10$:     ; Get next lock
                              0031   222
                          5B  D6  0031   223          INCL    R11                     ; Advance to next lock id.
          00000000'GF     5B  D1  0033   224          CMPL    R11,G^LCK$GL_MAXID      ; Reached the end?
                          3F  1A  003A   225          BGTRU   40$                     ; Yes
                    56  6A4B  D0  003C   226          MOVL    (R10)[R11],R6           ; Get pointer to LKB
                          EF  18  0040   227          BGEQ    10$                     ; Unused entry
                              0042   228
                              0042   229          ; Delete lock if it is a master copy  and has a zero
                              0042   230          ; reference count.  Also delete parents, if possible.
                              0042   231          ; If the lock belongs to this system, then change all transient
                              0042   232          ; SCS states to RETRY state.
                              0042   233
                    0400 8F   AA  0042   234          BICW    #LKB$M_RESEND,-         ; Clear RESEND bit
                       2A A6      0046   235                  LKB$W_STATUS(R6)
                          04  E0  0048   236          BBS     #LKB$V_MSTCPY,-         ; Branch if master copy
                    17 2A A6      004A   237                  LKB$W_STATUS(R6),20$
                              004D   238          DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
                              004D   239                  <-
                              004D   240                  <RSPNOTQED,15$>,-       ; Change temporary SCS wait states
                              004D   241                  <RSPQUEUED,15$>,-       ; into RETRY state
                              004D   242                  <RSPGRANTD,15$>,-
                              004D   243                  <RSPDOLOCL,15$>,-
                              004D   244                  >
                          D4  11  005B   245          BRB     10$                     ; All other states are okay as is
                       FE 8F  90  005D   246 15$:     MOVB    #LKB$K_RETRY,-
                       36 A6      0060   247                  LKB$B_STATE(R6)
                          CD  11  0062   248          BRB     10$
                              0064   249
                       4C A6  B5  0064   250 20$:     TSTW    LKB$W_REFCNT(R6)        ; Are there any sublocks?
                          C8  12  0067   251          BNEQ    10$                     ; Yes, defer to later
                 50    38 A6  0F  0069   252          REMQUE  LKB$L_SQFL(R6),R0       ; Remove LKB from RSB state queue
                 55    48 A6  D0  006D   253          MOVL    LKB$L_PARENT(R6),R5     ; Save address of parent
                    FF8C'     30  0071   254          BSBW    LCK$DEALLOC_LKB         ; Deallocate LKB
                    56    55  D0  0074   255          MOVL    R5,R6                   ; Now try parent
                          EB  12  0077   256          BNEQ    20$                     ; There is one
                          B6  11  0079   257          BRB     10$                     ; Advance to next lock id.
                              007B   258
                              007B   259 ; Loop through all RSBs in the resource hash table.  For each RSB,
                              007B   260 ; clear its DIRENTRY bit, set its CSID to an invalid CSID, delete it
                              007B   261 ; if all the queues are empty and its reference count is zero.
                              007B   262 ; Also delete parents, if possible.
                              007B   263
5A  01  00000000'GF   78  007B   264 40$:     ASHL    G^LCK$GL_HTBLCNT,#1,R10 ; Get size of hash table
    5B  00000000'GF   D0  0083   265          MOVL    G^LCK$GL_HASHTBL,R11    ; Get address of hash table
                              008A   266
```

```
                        008A   267 50$:    ; Start on next hash chain
                        008A   268
          59    8B  DE  008A   269         MOVAL   (R11)+,R9               ; Get address of next list head
                        008D   270
                        008D   271 60$:    ; Get next RSB in this hash chain.  R9 serves as the linkage
                        008D   272         ; pointer while R8 points to RSB to be processed.  These start
                        008D   273         ; out the same but if the RSB pointed to by R8 gets deleted, then
                        008D   274         ; R9 is backed up to point to the previous RSB.
                        008D   275
          59    69  D0  008D   276         MOVL    (R9),R9                 ; Get address of next RSB
                43  13  0090   277         BEQL    90$                     ; Reached end of chain
          58    59  D0  0092   278         MOVL    R9,R8                   ; Move RSB address to R8
                01  AA  0095   279 70$:     BICW    #RSB$M_DIRENTRY,-       ; Clear directory entry bit
              0E  A8        0097   280                 RSB$W_STATUS(R8)
       38 A8    01  CE  0099   281         MNEGL   #1,RSB$L_CSID(R8)       ; Store invalid CSID
       50   10  A8  DE  009D   282         MOVAL   RSB$L_GRQFL(R8),R0      ; Get address of granted queue
          50    60  D1  00A1   283         CMPL    (R0),R0                 ; Is granted queue empty?
                E7  12  00A4   284         BNEQ    60$                     ; No
          50    08  C0  00A6   285         ADDL    #8,R0                   ; Yes, get address of conversion queue
          50    60  D1  00A9   286         CMPL    (R0),R0                 ; Is conversion queue empty?
                DF  12  00AC   287         BNEQ    60$                     ; No
          50    08  C0  00AE   288         ADDL    #8,R0                   ; Yes, get address of wait queue
          50    60  D1  00B1   289         CMPL    (R0),R0                 ; Is wait queue empty?
                D7  12  00B4   290         BNEQ    60$                     ; No
                        00B6   291
                        00B6   292         ; All queues are empty.  Now check to see if it's reference count
                        00B6   293         ; is zero.  If it's reference count is non-zero then we will
                        00B6   294         ; handle it later when we climb up the tree of one of it's
                        00B6   295         ; descendants.
                        00B6   296
            40  A8  B5  00B6   297         TSTW    RSB$W_REFCNT(R8)        ; Are there any sub-resources?
                D2  12  00B9   298         BNEQ    60$                     ; Yes, move onto next RSB in chain
          59    58  D1  00BB   299         CMPL    R8,R9                   ; Are we deallocating our linkage?
                04  12  00BE   300         BNEQ    80$                     ; No
          59    04  A9  D0  00C0   301     MOVL    RSB$L_HSHCHNBK(R9),R9   ; Yes, back up linkage pointer
          57    48  A8  D0  00C4   302 80$: MOVL    RSB$L_PARENT(R8),R7     ; Save parent RSB address
       00000000'GF  16  00C8   303         JSB     G^LCK$DEALLOC_RSB       ; Deallocate RSB
          58    57  D0  00CE   304         MOVL    R7,R8                   ; Get parent RSB address
                C2  12  00D1   305         BNEQ    70$                     ; There is a parent, work on it
                B8  11  00D3   306         BRB     60$                     ; Repeat
                        00D5   307
                        00D5   308 90$:    ; Finished one complete hash chain.
                        00D5   309
                5A  D7  00D5   310         DECL    R10                     ; Decr. count of hash chains
                B1  14  00D7   311         BGTR    50$                     ; Repeat
                        00D9   312
                        00D9   313         ; Select a node to issue timestamps for deadlock detection.
                        00D9   314         ; Every node must select the same system.  An easy way to select
                        00D9   315         ; the same system everywhere is to use the first entry in the
                        00D9   316         ; directory vector.  These also have the (required) property
                        00D9   317         ; that the local CSID is referenced with a zero.
                        00D9   318         ; Also zero the bitmap expiration timestamps to prevent possible
                        00D9   319         ; false deadlocks due to the new timestamp issuing system
                        00D9   320         ; having a system time slightly behind timestamps already issued.
                        00D9   321
    50  00000000'GF  D0  00D9   322         MOVL    G^LCK$GL_DIRVEC,R0      ; Get address of directory vector
      0008'CF   60  D0  00E0   323         MOVL    (R0),W^LCK$GL_TS_CSID   ; Copy CSID
```

```
                50  00000000'GF   7E  00E5   324          MOVAQ   G^LCK$GQ_BITMAP_EXP,R0  ; Get address of timestamps
                              80   7C  00EC   325          CLRQ    (R0)+                   ; Reset exact expiration time
                              60   7C  00EE   326          CLRQ    (R0)                    ; Reset local (approx.) expir. time
                                       00F0   327
                          0FE0 8F   BA  00F0   328          POPR    #^M<R5,R6,R7,R8,R9,R10,R11>
                                    05  00F4   329          RSB
```

```
                     00F5   331          .SBTTL  LCK$REBUILD_LKBS - Rebuild LKBs
                     00F5   332
                     00F5   333 ;++
                     00F5   334 ; FUNCTIONAL DESCRIPTION:
                     00F5   335 ;
                     00F5   336 ;       This routine makes a pass through the lock id. table to
                     00F5   337 ;       process each lock (and its parents).  Root locks are
                     00F5   338 ;       sent to the appropriate directory system.  Sublocks (if mastered
                     00F5   339 ;       remotely) are sent to the system mastering the tree.
                     00F5   340 ;
                     00F5   341 ; CALLING SEQUENCE:
                     00F5   342 ;
                     00F5   343 ;       JSB     LCK$REBUILD_LKBS (called from failover table dispatcher)
                     00F5   344 ;       IPL must be at IPL$_SYNCH
                     00F5   345 ;
                     00F5   346 ; INPUT ARGUMENTS:
                     00F5   347 ;
                     00F5   348 ;       R5      Address of failover control block
                     00F5   349 ;
                     00F5   350 ; OUTPUT ARGUMENTS:
                     00F5   351 ;
                     00F5   352 ;       None
                     00F5   353 ;
                     00F5   354 ; SIDE EFFECTS:
                     00F5   355 ;
                     00F5   356 ;       R0 - R5 not preserved
                     00F5   357 ;--
                     00F5   358
                     00F5   359 LOCKS_DONE:
                     00F5   360          ; Finished processing entire lock id. table.  Continue with next
                     00F5   361          ; phase of failover.
                     00F5   362
01C0 8F    BA        00F5   363          POPR    #^M<R6,R7,R8>           ; Restore registers
0004'DF    17        00F9   364          JMP     @W^RETURN_ADDR          ; Return to caller via saved ret. addr.
                     00FD   365
                     00FD   366 LCK$REBUILD_LKBS::
0004'CF 8ED0         00FD   367          POPL    W^RETURN_ADDR           ; Save return address
                     0102   368
                     0102   369 ; Process all locks in the lock id. table.  For each lock, do the
                     0102   370 ; following:
                     0102   371 ;       If RSB$L_CSID is 0, then it is being mastered here and
                     0102   372 ;       there is nothing to do for this lock.
                     0102   373 ;       If RSB$L_CSID is -1 then we have to find out where it
                     0102   374 ;       is being mastered.  We find this out by climbing its tree
                     0102   375 ;       until we find a RSB$L_CSID not equal to -1.  If we reach the
                     0102   376 ;       top of the tree, then we send it to the appropriate directory
                     0102   377 ;       system.
                     0102   378 ;       If RSB$L_CSID is a non-zero CSID then we send the lock to that
                     0102   379 ;       system is LKB$M_RESEND is set.
                     0102   380 ;
0000'CF    D4        0102   381          CLRL    W^CURR_LOCKID           ; Initialize current lockid
                     0106   382
                     0106   383 NEXT_LOCKID_SAVE:
01C0 8F    BB        0106   384          PUSHR   #^M<R6,R7,R8>           ; Save registers
                     010A   385 NEXT_LOCKID:
0000'CF    D6        010A   386          INCL    W^CURR_LOCKID           ; Advance to next lock id.
0000'CF    D1        010E   387          CMPL    W^CURR_LOCKID,-         ; Reached the end of the id. table?
```

```
                00000000'GF        0112   388                      G^LCK$GL_MAXID
                          DC   1A  0117   389              BGTRU   LOCKS_DONE              ; Yes
                               0119   390 SAME_LOCKID:
              51    0000'CF    D0  0119   391              MOVL    W^CURR_LOCKID,R1        ; Get lock id.
           50   00000000'GF    D0  011E   392              MOVL    G^LCK$GL_IDTBL,R0       ; Get address of lock id. table
                   56   6041   D0  0125   393              MOVL    (R0)[R1],R6             ; Get pointer to LKB
                          DF   18  0129   394              BGEQ    NEXT_LOCKID             ; Unused entry
                               012B   395
                    58    50 A6   D0  012B   396              MOVL    LKB$L_RSB(R6),R8        ; Get address of RSB
                    53    38 A8   D0  012F   397              MOVL    RSB$L_CSID(R8),R3       ; Get system managing this resource
                          D5   13  0133   398              BEQL    NEXT_LOCKID             ; It's us
                               0135   399
                               0135   400              ; R3 contains CSID of system managing this resource tree or -1.
                               0135   401              ; If it is a CSID and the lock does not have to be resent, then
                               0135   402              ; just continue onto the next lock id.
                               0135   403              ; Otherwise, climb up the tree to the first lock that is not
                               0135   404              ; marked to be resent and whose RSB has a valid CSID (or zero).
                               0135   405              ; Then resend locks below that lock to that system.  If we
                               0135   406              ; reach the root of the tree, then send the root lock to the
                               0135   407              ; appropriate directory system.  If we are the directory
                               0135   408              ; system for this resource, then we manage the
                               0135   409              ; tree.  Likewise, if we reach a lock whose RSB has a
                               0135   410              ; zero CSID then we are managing this tree.  In this case,
                               0135   411              ; reclimb the tree clearing the RSB$L_CSID fields along the way.
                               0135   412
                          53   B5  0135   413              TSTW    R3                      ; Is CSID valid?
                          05   19  0137   414              BLSS    20$                     ; No, lock must be remastered
                          0A   E1  0139   415              BBC     #LKB$V_RESEND,-         ; Branch if this lock
                      CC 2A A6        013B   416                      LKB$W_STATUS(R6),NEXT_LOCKID  ; need not be resent
                               013E   417
                               013E   418 20$:     ; The current lock must be resent.  Climb tree to first lock
                               013E   419              ; that doesn't have to be resent (RESEND bit = 0 and CSID is valid).
                               013E   420
                     57    56   D0  013E   421              MOVL    R6,R7                   ; Save starting LKB in R7
                     55    56   D0  0141   422              MOVL    R6,R5                   ; Put starting LKB in R5
                               0144   423
                     56    55   D0  0144   424 30$:     MOVL    R5,R6                   ; R6 points to last LKB
                 55    48 A5   D0  0147   425              MOVL    LKB$L_PARENT(R5),R5     ; R5 points to parent LKB
                          15   13  014B   426              BEQL    40$                     ; Reached the top; R6 is root LKB
                 58    50 A5   D0  014D   427              MOVL    LKB$L_RSB(R5),R8        ; Get RSB
                 53    38 A8   D0  0151   428              MOVL    RSB$L_CSID(R8),R3       ; Get CSID
                          2C   13  0155   429              BEQL    50$                     ; This system is managing resource tree
                          0A   E0  0157   430              BBS     #LKB$V_RESEND,-         ; Branch if this LKB must be resent
                    E8 2A A5        0159   431                      LKB$W_STATUS(R5),30$
                          53   B5  015C   432              TSTW    R3                      ; Is CSID valid?
                          E4   19  015E   433              BLSS    30$                     ; No
                               0160   434
                               0160   435              ; R8 points to a RSB with a valid CSID and the corresponding lock
                               0160   436              ; (in R5) does not need to be resent.  Resend the lock pointed to
                               0160   437              ; by R6 to the same system.
                               0160   438
                          61   11  0160   439              BRB     REBUILD
                               0162   440
                               0162   441 40$:     ; R6 points to root LKB.  If its RSB$L_CSID (R3) is valid then resend
                               0162   442              ; to that system.  Otherwise resend to the appropriate directory
                               0162   443              ; system (unless we are the directory system for that resource).
                               0162   444
```

```
                                                                I 15
                     53   B5  0162    445          TSTW    R3                      ; Is CSID valid?
                     5D   18  0164    446          BGEQ    REBUILD                 ; Yes
          53  00000000'GF   D0  0166    447          MOVL    G^LCK$GL_DIRVEC,R3      ; No, get address of directory vector
              51    44 A8   3C  016D    448          MOVZWL  RSB$W_HASHVAL(R8),R1    ; Get hash value
                     52   D4  0171    449          CLRL    R2                      ; Clear high order hash value
      51  50  51    F4 A3   7B  0173    450          EDIV    -12(R3),R1,R0,R1        ; Compute hash index (in R1)
               53  6341   D0  0179    451          MOVL    (R3)[R1],R3             ; Get directory system
                     44   12  017D    452          BNEQ    REBUILD                 ; It's not us
                     01   A8  017F    453          BISW    #RSB$M_DIRENTRY,-       ; Set directory entry bit
                  0E A8           0181    454                  RSB$W_STATUS(R8)
                                  0183    455
                                  0183    456 50$:    ; This system is now managing this resource tree.  Starting at
                                  0183    457         ; the current LKB (R7) climb tree to LKB whose RSB contains a
                                  0183    458         ; zero CSID (R5) clearing RSB$L_CSID along the way.
                                  0183    459
              58    50 A7   D0  0183    460          MOVL    LKB$L_RSB(R7),R8        ; Get address of RSB
                    38 A8   B5  0187    461          TSTW    RSB$L_CSID(R8)          ; Make sure current CSID is invalid
                     33   18  018A    462          BGEQ    60$                     ; Error - it's valid
                    38 A8   D4  018C    463          CLRL    RSB$L_CSID(R8)          ; Clear CSID
              57    48 A7   D0  018F    464          MOVL    LKB$L_PARENT(R7),R7     ; Get next parent
                 55   57   D1  0193    465          CMPL    R7,R5                   ; Reached the top?
                     EB   12  0196    466          BNEQ    50$                     ; No
                                  0198    467
                                  0198    468         ; If R5 is 0 then the current lock is a root lock that we are
                                  0198    469         ; now mastering (on the directory system).  Fork in this case
                                  0198    470         ; to allow other systems a chance to get directory entries.
                                  0198    471
                     55   D5  0198    472          TSTL    R5                      ; Is this a root lock?
                     20   12  019A    473          BNEQ    55$                     ; No, advance to next lock id.
                01C0 8F   BA  019C    474          POPR    #^M<R6,R7,R8>
            0000010C 8F   C1  01A0    475          ADDL3   #LUB$B_CLUFCB,-         ; Get address of failover control block
      55    00000000'GF       01A6    476                  G^CLU$GL_CLUB,R5
                     08   90  01AC    477          MOVB    #IPL$_SYNCH,-           ; Store fork IPL
                  0B A5           01AE    478                  FKB$B_FIPL(R5)
                                  01B0    479          FORK
                   FE47'  30  01B6    480          BSBW    CNX$CHECK_FAILOVER      ; Check for another failover
                   FF4A   31  01B9    481          BRW     NEXT_LOCKID_SAVE        ; Yes
                                  01BC    482
                   FF4B   31  01BC    483 55$:     BRW     NEXT_LOCKID
                                  01BF    484 60$:     BUG_CHECK       LOCKMGRERR,FATAL
                                  01C3    485
                                  01C3    486 REBUILD:
                                  01C3    487         ; Have a LKB (in R6) that needs to be rebuilt on destination system
                                  01C3    488         ; (CSID in R3).  However, locks in certain states aren't resent.
                                  01C3    489         ; Dispatch on lock state.
                                  01C3    490
          58    50 A6   D0  01C3    491          MOVL    LKB$L_RSB(R6),R8        ; Get RSB address
                                  01C7    492          DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
                                  01C7    493                  <-
                                  01C7    494                  <GRANTED,20$>,-
                                  01C7    495                  <CONVERT,20$>,-
                                  01C7    496                  <WAITING,20$>,-
                                  01C7    497                  <RETRY,10$>,-
                                  01C7    498                  <SCSWAIT,10$>,-
                                  01C7    499                  >
                                  01D7    500          BUG_CHECK       LOCKMGRERR,FATAL; Other states are not allowed
                                  01DB    501
```

```
                          01DB    502 10$:    ; Handle locks in RETRY and SCSWAIT states as special cases.
                          01DB    503         ; If they are a conversion then they must be rebuilt.  If they
                          01DB    504         ; are a new lock then they are not sent as they will be
                          01DB    505         ; retried or deallocated.
                          01DB    506
                01   E0   01DB    507         BBS     #LCK$V_CONVERT,-        ; Branch if conversion
                03 28 A6   01DD    508                 LKB$W_FLAGS(R6),20$
                    00C7   31   01E0    509         BRW     STORE_CSID              ; New lock - just store CSID
                          01E3    510
                          01E3    511 20$:    ; Rebuild LKB (in R6) on system whose CSID is in R3
                          01E3    512
                     FE1A' 30   01E3    513         BSBW    CNX$ALLOC_WARMCDRP      ; Allocate a CDRP
                     FE17' 30   01E6    514         BSBW    CNX$RESOURCE_CHECK      ; Only retry a limited number of times
                   67 50   E9   01E9    515         BLBC    R0,30$                  ; Couldn't allocate one or CSID invalid
              51   0C A6   3C   01EC    516         MOVZWL  LKB$L_PID(R6),R1        ; Get PID index
                     0B   13   01F0    517         BEQL    25$                     ; Branch if system owned
         50 00000000'GF   D0   01F2    518         MOVL    G^SCH$GL_PCBVEC,R0      ; Get address of PCB vector
              51   6041   D0   01F9    519         MOVL    (R0)[R1],R1             ; Get address of PCB
             48 A5   51   D0   01FD    520 25$:    MOVL    R1,CDRP$L_VAL8(R5)      ; Store PCB address in CDRP
             2C A5   56   D0   0201    521         MOVL    R6,CDRP$L_VAL1(R5)      ; Store LKB address
             30 A5   58   D0   0205    522         MOVL    R8,CDRP$L_VAL2(R5)      ; Store RSB address
             50   64 A3   D0   0209    523         MOVL    CSB$L_CLUB(R3),R0       ; Get CLUB address
                00AC C0   B0   020D    524         MOVW    CLUB$W_MEMSEQ(R0),-     ; Store MEMSEQ in CDRP
                   34 A5        0211    525                 CDRP$L_VAL3(R5)
                0000'CF   9E   0213    526         MOVAB   W^LCK$BLD_REBLDLOCK,-   ; Store address of message build routine
                   4C A5        0217    527                 CDRP$L_MSGBLD(R5)
                01C0 8F   BA   0219    528         POPR    #^M<R6,R7,R8>           ; Restore registers
                     FDE0' 30   021D    529         BSBW    CNX$SEND_MSG_CSB        ; Send the message
                   57 50   E9   0220    530         BLBC    R0,50$                  ; Exit this routine (restart failover)
                01C0 8F   BB   0223    531         PUSHR   #^M<R6,R7,R8>           ; Save registers
         56   2C A5   D0   0227    532         MOVL    CDRP$L_VAL1(R5),R6      ; Get address of LKB
         58   50 A6   D0   022B    533         MOVL    LKB$L_RSB(R6),R8        ; Get address of RSB
                          022F    534
                          022F    535         ; Have response message.  Registers contain:
                          022F    536         ;       R2      Address of response message
                          022F    537         ;       R3      Address of CSB
                          022F    538         ;       R5      Address of CDRP
                          022F    539         ;       R6      Address of LKB
                          022F    540         ;       R8      Address of RSB
                          022F    541
         50   64 A3   D0   022F    542         MOVL    CSB$L_CLUB(R3),R0       ; Get address of CLUB
            00AC C0   B1   0233    543         CMPW    CLUB$W_MEMSEQ(R0),-     ; Check MEMSEQ in message
              0C A2        0237    544                 LKMSG$W_MEMSEQ(R2)
                 58   12   0239    545         BNEQ    REBLD_RETRY             ; No match!
                          023B    546         DISPATCH        LKMSG$B_STATE(R2),TYPE=B,PREFIX=LKB$K_,-
                          023B    547                 <-
                          023B    548                 <RSPNOTQED,REBLD_NOTQED>,-
                          023B    549                 <RSPDOLOCL,REBLD_DOLOCL>,-
                          023B    550                 <RSPRESEND,REBLD_RESEND>,-
                          023B    551                 <RSPGRANTD,REBLD_GRANTD>,-
                          023B    552                 <RETRY,REBLD_RETRY>,-
                          023B    553                 >
                          024F    554         BUG_CHECK       LOCKMGRERR,FATAL; Other states are not allowed
                          0253    555
                          0253    556 30$:    ; Failed to allocate a CDRP.  Wait and retry if SS$_INSFMEM.
                          0253    557         ; Bugcheck otherwise.
                          0253    558
```

```
                0000'8F    50   B1  0253   559          CMPW    R0,#SS$_INSFMEM
                      1C   12   12  0258   560          BNEQ    40$
                    01C0   8F   BA  025A   561          POPR    #^M<R6,R7,R8>
                0000010C   8F   C1  025E   562          ADDL3   #CLUB$B_CLUFCB,-        ; Get address of failover control block
              55 00000000'GF                0264   563                  G^CLU$GL_CLUB,R5
                      08   90   9A  026A   564          MOVB    #IPL$_SYNCH,-           ; Store fork IPL
                   0B A5        9A  026C   565                  FKB$B_FIPL(R5)
                               9A  026E   566          FORK_WAIT                       ; Fork and wait
                      4B   11   11  0274   567          BRB     CHECK_FAILOVER2         ; Check failover and redo same lock
                                  0276   568 40$:      BUG_CHECK       LOCKMGRERR,FATAL
                                  027A   569
                                  027A   570 50$:      ; Exit this routine as we will start another failover.  Have to
                                  027A   571           ; deallocate CDRP in R5.
                                  027A   572
                      50   55   D0  027A   573          MOVL    R5,R0                   ; Address of CDRP
                00000000'GF    16   027D   574          JSB     G^EXE$DEANONPAGED       ; Deallocate it
                        FD7A'  31   0283   575          BRW     CNX$END_FAILOVER        ; End this failover
                                  0286   576
                                  0286   577 REBLD_NOTQED:
                                  0286   578           ; Lock wasn't rebuilt.  This may be due to insufficient lockids
                                  0286   579           ; on the remote system or it may be a bug.  Either way, bugcheck.
                                  0286   580           ; As currently implemented in DSTRLOCK, we should never see this
                                  0286   581           ; bugcheck as the remote system bugchecks with an appropriate
                                  0286   582           ; resource exhausted bugcheck.
                                  0286   583
                                  0286   584           BUG_CHECK       RESEXH,FATAL
                                  028A   585
                                  028A   586 REBLD_DOLOCL:
                                  028A   587           ; Manage this resource on this system
                                  028A   588
                        00000002  028A   589           .IF NE  CA$_MEASURE
                00000000'GF    D6   028A   590           INCL    G^PMS$GL_DIR_OUT
                                  0290   591           .ENDC
                                  0290   592
                      38 A8    D4   0290   593           CLRL    RSB$L_CSID(R8)          ; Indicate it's managed locally
                                  0293   594
                                  0293   595 REBLD_RETRY:
                       FD6A'   30   0293   596           BSBW    CNX$DEALL_WARMCDRP_CSB  ; Deallocate CDRP, message bfr., etc.
                         25    11   0296   597           BRB     CHECK_FAILOVER
                                  0298   598
                                  0298   599 REBLD_GRANTD:
                                  0298   600           ; Lock was rebuilt on specified destination system
                                  0298   601
                        00000002  0298   602           .IF NE  CA$_MEASURE
                00000000'GF    D6   0298   603           INCL    G^PMS$GL_ENQNEW_OUT
                                  029E   604           .ENDC
                                  029E   605
                      10 A2    D0   029E   606           MOVL    LKMSG$L_MSTLKID(R2),-   ; Store master lock id. in LKB
                      54 A6        02A1   607                   LKB$L_REMLKID(R6)
                       FD5A'   30   02A3   608           BSBW    CNX$DEALL_WARMCDRP_CSB  ; Deallocate CDRP, message bfr., etc.
               53     4C A3    D0   02A6   609           MOVL    CSB$L_CSID(R3),R3       ; Get destination CSID
                                  02AA   610
                                  02AA   611 STORE_CSID:
                                  02AA   612           ; If the CSID in R3 is different than the stored CSID, then store
                                  02AA   613           ; this one and mark all locks to be resent.  However, we verify
                                  02AA   614           ; that the one we are overwriting is not valid.  If it is valid,
                                  02AA   615           ; then we verify that it got stored as a result of NOT rebuilding
```

```
                          02AA   616          ; a lock (see REBUILD and 10$ code, above and LCK$MARK_FOR_RESEND).
                          02AA   617
 38 A8    53    D1        02AA   618          CMPL    R3,RSB$L_CSID(R8)       ; Does it match what's already stored?
             07    13     02AE   619          BEQL    40$                     ; Yes
          0109    30      02B0   620          BSBW    LCK$MARK_FOR_RESEND     ; Mark all other LKBs to be resent
 38 A8    53    D0        02B3   621          MOVL    R3,RSB$L_CSID(R8)       ; Store new CSID
    0400 8F    AA        02B7   622 40$:      BICW    #LKB$M_RESEND,-         ; Clear resend bit on this one to
      2A A6               02BB   623                  LKB$W_STATUS(R6)        ; indicate it has been resent
                          02BD   624
                          02BD   625 CHECK_FAILOVER:
    01C0 8F    BA        02BD   626          POPR    #^M<R6,R7,R8>
                          02C1   627 CHECK_FAILOVER2:
       FD3C'   30        02C1   628          BSBW    CNX$CHECK_FAILOVER      ; Check for new failover
    01C0 8F    BB        02C4   629          PUSHR   #^M<R6,R7,R8>
       FE4E    31        02C8   630          BRW     SAME_LOCKID
                          02CB   631
                          02CB   632 REBLD_RESEND:
                          02CB   633          ; Resend request to specified system.
                          02CB   634
          00000002        02CB   635          .IF NE  CA$_MEASURE
00000000'GF    D6        02CB   636          INCL    G^PMS$GL_DIR_OUT
                          02D1   637          .ENDC
                          02D1   638
      18 A2    DD        02D1   639          PUSHL   LKMSG$L_CSID(R2)        ; Save CSID of specified system
       FD29'   30        02D4   640          BSBW    CNX$DEALL_WARMCDRP_CSB  ; Deallocate CDRP, message bfr., etc.
         53 8ED0         02D7   641          POPL    R3                      ; Restore CSID
 52      56    D0        02DA   642          MOVL    R6,R2                   ; Move LKB address
    01C0 8F    BA        02DD   643          POPR    #^M<R6,R7,R8>
       FD1C'   30        02E1   644          BSBW    CNX$CHECK_FAILOVER      ; Check for new failover
    01C0 8F    BB        02E4   645          PUSHR   #^M<R6,R7,R8>
 56      52    D0        02E8   646          MOVL    R2,R6                   ; Move LKB address
       FED5    31        02EB   647          BRW     REBUILD                 ; Send to specified system
```

```
                     02EE    649          .SBTTL  LCK$REBLD_LOCK - Rebuild a lock during failover
                     02EE    650 ;++
                     02EE    651 ; FUNCTIONAL DESCRIPTION:
                     02EE    652 ;
                     02EE    653 ;       This routine is called from the received lock message routines
                     02EE    654 ;       to build a LKB during failover.
                     02EE    655 ;
                     02EE    656 ; CALLING SEQUENCE:
                     02EE    657 ;
                     02EE    658 ;       BSBW    LCK$REBLD_LOCK
                     02EE    659 ;       IPL must be at IPL$_SYNCH
                     02EE    660 ;
                     02EE    661 ; INPUT PARAMETERS:
                     02EE    662 ;
                     02EE    663 ;       R6      Address of LKB
                     02EE    664 ;       R8      Address of RSB
                     02EE    665 ;       R9      Address of input message
                     02EE    666 ;
                     02EE    667 ; OUTPUT PARAMETERS:
                     02EE    668 ;
                     02EE    669 ;       None
                     02EE    670 ;
                     02EE    671 ; SIDE EFFECTS:
                     02EE    672 ;
                     02EE    673 ;       R0 - R4 not preserved
                     02EE    674 ;--
                     02EE    675
                     02EE    676 LCK$REBLD_LOCK::
     6A A9    90     02EE    677          MOVB    LKMSG$B_LCKSTATE(R9),-  ; Store lock state
     36 A6           02F1    678                  LKB$B_STATE(R6)
                     02F3    679
                     02F3    680          ; Dispatch according to lock state
                     02F3    681
                     02F3    682          DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
                     02F3    683                  <-
                     02F3    684                  <GRANTED,40$>,-
                     02F3    685                  <CONVERT,30$>,-
                     02F3    686                  <WAITING,60$>,-
                     02F3    687                  >
                     02FF    688 20$:     BUG_CHECK       LOCKMGRERR,FATAL; Illegal lock mode
                     0303    689
                     0303    690 30$:     ; Lock state is CONVERT.  Lock needs to be placed in sequence
                     0303    691          ; number order in the conversion queue and then the value
                     0303    692          ; block should be stored if it's newer than the existing value block.
                     0303    693
 53  18 A8    9E     0303    694          MOVAB   RSB$L_CVTQFL(R8),R3     ; Get address of conversion queue
    35        10     0307    695          BSBB    80$                     ; Process like WAITING locks
    0A        11     0309    696          BRB     50$                     ; Process value block like GRANTED locks
                     030B    697
                     030B    698 40$:     ; Lock state is GRANTED.  Insert on granted queue, set LKB status
                     030B    699          ; bits and store value block if it's newer than the existing one.
                     030B    700
     38 A6    0E     030B    701          INSQUE  LKB$L_SQFL(R6),-        ; Insert lock on granted queue
     10 A8           030E    702                  RSB$L_GRQFL(R8)
     68 A9    88     0310    703          BISB    LKMSG$B_LSTATUS(R9),-   ; Set appropriate status bits
     2A A6           0313    704                  LKB$W_STATUS(R6)
     3C A8    C3     0315    705 50$:     SUBL3   RSB$L_VALSEQNUM(R8),-   ; Is value block in message newer
```

```
                                0318   706                      LKMSG$L_VALSEQALT(R9),R0; than current one in RSB?
                          1C    031B   707              BLEQ    55$                      ; No
                          7D    031D   708              MOVQ    LKMSG$Q_VALBLKALT(R9),- ; Yes store new value block
                                0320   709                      RSB$Q_VALBLK(R8)
                          7D    0322   710              MOVQ    LKMSG$Q_VALBLKALT+8(R9),-
                                0325   711                      RSB$Q_VALBLK+8(R8)
                          D0    0327   712              MOVL    LKMSG$L_VALSEQALT(R9),- ; Store new sequence number
                                032A   713                      RSB$L_VALSEQNUM(R8)
                          AA    032C   714              BICW    #RSB$M_VALINVLD,-       ; Clear value invalid flag
                                032E   715                      RSB$W_STATUS(R8)
                          E1    0330   716              BBC     #RSB$V_VALINVLD,-       ; Optionally set flag
                                0332   717                      LKMSG$B_RSTATUS(R9),55$
                          A8    0335   718              BISW    #RSB$M_VALINVLD,-
                                0337   719                      RSB$W_STATUS(R8)
                          05    0339   720 55$:         RSB
                                033A   721
                                033A   722 60$:         ; Lock state is WAITING.  Set appropriate LKB status bits and
                                033A   723              ; insert lock onto waiting queue in sequence number order.
                                033A   724
                          9E    033A   725              MOVAB   RSB$L_WTQFL(R8),R3      ; Get address of WAIT queue
                                033E   726
                                033E   727 80$:         ; Common code for waiting and converting locks
                                033E   728
                          A8    033E   729              BISW    #LKB$M_ASYNC,-          ; Set ASYNC bit
                                0340   730                      LKB$W_STATUS(R6)
                          3C    0342   731              MOVZWL  LKMSG$W_RQSEQALT(R9),R4 ; Get request sequence number
                          B0    0346   732              MOVW    R4,LKB$W_RQSEQNM(R6)    ; Store it
                                034A   733
                                034A   734              ; Traverse queue (address in R3) for correct place to insert lock.
                                034A   735              ; R4 contains this lock's sequence number.
                                034A   736
                          D0    034A   737              MOVL    R3,R2                   ; Save address of header in R2
                          D0    034D   738 90$:         MOVL    (R2),R2                 ; Get next LKB
                          D1    0350   739              CMPL    R2,R3                   ; Reached the end?
                          13    0353   740              BEQL    97$                     ; Yes
                    54    A3    0355   741              SUBW3   R4,LKB$W_RQSEQNM-LKB$L_SQFL(R2),R0 ; Compare sequence numbers
                          19    035A   742              BLSS    90$                     ; Move to next LKB
                    38    0E    035C   743 95$:         INSQUE  LKB$L_SQFL(R6),@4(R2)   ; Insert this lock before lock in R2
                          05    0361   744              RSB
                                0362   745
                                0362   746 97$:         ; Have a new highest sequence number.  Store it +1 in RSB.
                                0362   747
                    54    01    A1     0362   748              ADDW3   #1,R4,RSB$W_RQSEQNM(R8)
                          F3    11     0367   749              BRB     95$
```

```
0369   751          .SBTTL  LCK$CHECK_DIRENTRY - Check if this is a directory entry
0369   752 ;++
0369   753 ; FUNCTIONAL DESCRIPTION:
0369   754 ;
0369   755 ;       This routine is called during failover when it appears that
0369   756 ;       another system has performed a directory lookup.
0369   757 ;       If this is a root resource then we verify the CSID in the
0369   758 ;       directory entry is valid.  If it's not, then the requesting
0369   759 ;       system gets to manage this resource.
0369   760 ;
0369   761 ;       If this is not a root resource, then we must be managing this
0369   762 ;       tree but have not yet reached this RSB.  In this case, clear
0369   763 ;       the CSID in this RSB and in all parent RSBs until we reach
0369   764 ;       an RSB that already has a zero CSID.
0369   765 ;
0369   766 ; CALLING SEQUENCE:
0369   767 ;
0369   768 ;       BSBW    LCK$CHECK_DIRENTRY
0369   769 ;
0369   770 ; INPUT PARAMETERS:
0369   771 ;
0369   772 ;       R4      CSID of system managing resource (or -1)
0369   773 ;       R5      CSID of system doing directory lookup (or lock rebuild)
0369   774 ;       R8      Address of RSB
0369   775 ;
0369   776 ; OUTPUT PARAMETERS:
0369   777 ;
0369   778 ;       R0      Completion code
0369   779 ;       R4      CSID of system managing resource or CSID
0369   780 ;               of system to resend request to
0369   781 ;
0369   782 ; IMPLICIT OUTPUTS:
0369   783 ;
0369   784 ;       The CSIDs in this RSB tree may be modified
0369   785 ;
0369   786 ; COMPLETION CODES:
0369   787 ;
0369   788 ;       0       This is not a directory entry - rebuild the lock
0369   789 ;       1       This is a directory entry - perform a directory lookup
0369   790 ;
0369   791 ; SIDE EFFECTS:
0369   792 ;
0369   793 ;       R0 - R3 not preserved
0369   794 ;--
0369   795
0369   796 LCK$CHECK_DIRENTRY::
            53   54    D0  0369   797          MOVL    R4,R3                   ; Move CSID for this RSB
              48 A8    D5  036C   798          TSTL    RSB$L_PARENT(R8)        ; Is this a root resource?
                 2E    12  036F   799          BNEQ    50$                     ; No
                 53    B5  0371   800          TSTW    R3                      ; Is CSID valid?
                 26    14  0373   801          BGTR    40$                     ; Yes
                           0375   802
                           0375   803          ; CSID is invalid.  Verify this is the directory system for this
                           0375   804          ; resource and if so, make this the correct directory entry.
                           0375   805
      53   00000000'GF    D0  0375   806          MOVL    G^LCK$GL_DIRVEC,R3      ; Get address of directory vector
            51   44 A8    3C  037C   807          MOVZWL  RSB$W_HASHVAL(R8),R1    ; Get hash value
```

```
                                52   D4  0380    808          CLRL    R2                      ; Clear high order hash value
          51    50    51    F4 A3   7B  0382    809          EDIV    -12(R3),R1,R0,R1        ; Compute hash index (in R1)
                       54     6341   D0  0388    810          MOVL    (R3)[R1],R4             ; Get directory system
                                 0D   12  038C    811          BNEQ    40$                     ; It's not us
                                 01   A8  038E    812          BISW    #RSB$M_DIRENTRY,-       ; It is us; set directory entry bit
                              0E A8           0390    813                  RSB$W_STATUS(R8)
                                 28   10  0392    814          BSBB    LCK$MARK_FOR_RESEND     ; Mark all LKBs on this RSB to be resent
                          38 A8  55   D0  0394    815          MOVL    R5,RSB$L_CSID(R8)       ; Requesting system manages resource
                             54  55   D0  0398    816          MOVL    R5,R4                   ; Return this CSID in R4
                             50  01   D0  039B    817 40$:     MOVL    #1,R0                   ; Process like a directory lookup
                                 05   039E    818          RSB
                                      039F    819
                                      039F    820 50$:     ; This RSB is not a root resource.  This tree must be managed
                                      039F    821          ; on this system but we haven't reached this RSB yet.  Set
                                      039F    822          ; all RSBs on this leg of the tree to be managed locally.
                                      039F    823          ;       R3 = CSID
                                      039F    824          ;       R8 = RSB
                                      039F    825
                             52   58   D0  039F    826          MOVL    R8,R2                   ; Move address of RSB
                                 53   B5  03A2    827 60$:     TSTW    R3                      ; Verify CSID is not valid
                                 12   14  03A4    828          BGTR    80$                     ; Error
                              38 A2   D4  03A6    829          CLRL    RSB$L_CSID(R2)          ; Manage this resource locally
                          52  48 A2   D0  03A9    830          MOVL    RSB$L_PARENT(R2),R2     ; Get parent
                                 09   13  03AD    831          BEQL    80$                     ; Error if we reach top of tree
                          53  38 A2   D0  03AF    832          MOVL    RSB$L_CSID(R2),R3       ; Get CSID
                                 ED   12  03B3    833          BNEQ    60$                     ; Repeat
                                      03B5    834
                                 50   D4  03B5    835          CLRL    R0                      ; Set status to indicate rebuild lock
                                 05   03B7    836          RSB
                                      03B8    837
                                      03B8    838 80$:     BUG_CHECK       LOCKMGRERR,FATAL
```

```
                     03BC    840          .SBTTL  LCK$MAPK_FOR_RESEND - Mark LKBs on RSB for resending
                     03BC    841
                     03BC    842 ;++
                     03BC    843 ; FUNCTIONAL DESCRIPTION:
                     03BC    844 ;
                     03BC    845 ;       LCK$MARK_FOR_RESEND is called during failover whenever the RSB$L_CSID
                     03BC    846 ;       field changes from invalid to valid (but not zero).  This routine
                     03BC    847 ;       scans all the state queues on the RSB and marks all the LKBs
                     03BC    848 ;       to be resent.
                     03BC    849 ;
                     03BC    850 ;       This routine is also called during failover whenever the RSB$L_CSID
                     03BC    851 ;       field changes from one valid system to another (but not zero).
                     03BC    852 ;       In this case, it is necessary to verify that no locks have
                     03BC    853 ;       actually been resent to the old system.
                     03BC    854 ;
                     03BC    855 ;       NOTE:  The two cases are distinguished by whether or not the CSID in
                     03BC    856 ;              RSB$L_CSID is valid.  Consequently, storing a new CSID MUST be
                     03BC    857 ;              done AFTER calling this routine.
                     03BC    858 ;
                     03BC    859 ; CALLING SEQUENCE:
                     03BC    860 ;
                     03BC    861 ;       BSBW    LCK$MARK_FOR_RESEND
                     03BC    862 ;       IPL must be at IPL$_SYNCH
                     03BC    863 ;
                     03BC    864 ; INPUT PARAMETERS:
                     03BC    865 ;
                     03BC    866 ;       R8      Address of RSB
                     03BC    867 ;
                     03BC    868 ; IMPLICIT INPUTS:
                     03BC    869 ;
                     03BC    870 ;       RSB$L_CSID is used as described above.
                     03BC    871 ;
                     03BC    872 ; OUTPUT PARAMETERS:
                     03BC    873 ;
                     03BC    874 ;       None
                     03BC    875 ;
                     03BC    876 ; SIDE EFFECTS:
                     03BC    877 ;
                     03BC    878 ;       R0 - R2 not preserved
                     03BC    879 ;--
                     03BC    880
                     03BC    881 LCK$MARK_FOR_RESEND:
                     03BC    882          ASSUME  RSB$L_CVTQFL  EQ  RSB$L_GRQFL+8
                     03BC    883          ASSUME  RSB$L_WTQFL  EQ  RSB$L_CVTQFL+8
                     03BC    884
   51   10 A8   DE  03BC    885          MOVAL   RSB$L_GRQFL(R8),R1      ; Get address of granted queue
        52   03   D0  03C0    886          MOVL    #3,R2                   ; Process all three queues
                     03C3    887
        50   51   D0  03C3    888 10$:     MOVL    R1,R0                   ; R0 will step through queue
                     03C6    889
        50   60   D0  03C6    890 20$:     MOVL    (R0),R0                 ; Get next LKB on queue
        51   50   D1  03C9    891          CMPL    R0,R1                   ; Reached end of queue?
             22   13  03CC    892          BEQL    30$                     ; Yes
          38 A8   B5  03CE    893          TSTW    RSB$L_CSID(R8)          ; Is CSID valid?
             08   18  03D1    894          BGEQ    25$                     ; Yes
        0400 8F   A8  03D3    895          BISW    #LKB$M_RESEND,-         ; Set resend bit in LKB
          F2 A0       03D7    896                  LKB$W_STATUS-LKB$L_SQFL(R0)
```

```
                EB   11  03D9   897          BRB     20$                     ; Move on to next LKB in this queue
                                 03DB   898
                0A   E0  03DB   899 25$:     BBS     #LKB$V_RESEND,-         ; Ok if lock hasn't been rebuilt yet
            E6 F2 A0         03DD   900              LKB$W_STATUS-LKB$L_SQFL(R0),20$
                             03E0   901          DISPATCH        LKB$B_STATE-LKB$L_SQFL(R0),TYPE=B,PREFIX=LKB$K_,-
                             03E0   902                  <-
                             03E0   903                  <RETRY,20$>,-           ; These states haven't been rebuilt
                             03E0   904                  <SCSWAIT,20$>,-         ; and are thus okay too.
                             03E0   905                  <RSPNOTQED,20$>,-
                             03E0   906                  >
                             03EC   907          BUG_CHECK       LOCKMGRERR,FATAL
                             03F0   908
               51    08  C0  03F0   909 30$:     ADDL    #8,R1                   ; Point to next queue
                 CD 52   F5  03F3   910          SOBGTR  R2,10$                  ; Repeat for all three queues
                         05  03F6   911          RSB
```

```
                                03F7   913          .SBTTL  LCK$REBUILD_RSBS
                                03F7   914
                                03F7   915 ;++
                                03F7   916 ; FUNCTIONAL DESCRIPTION:
                                03F7   917 ;
                                03F7   918 ;       This routine makes a pass through the resource hash table
                                03F7   919 ;       to rebuild the RSB database.  For each RSB, it computes new group
                                03F7   920 ;       grant and conversion grant modes, a new BLKASTCNT, and grants
                                03F7   921 ;       any possible unprotected locks.
                                03F7   922 ;
                                03F7   923 ; CALLING SEQUENCE:
                                03F7   924 ;
                                03F7   925 ;       JSB     LCK$REBUILD_RSBS (called from failover table dispatcher)
                                03F7   926 ;       IPL must be at IPL$_SYNCH
                                03F7   927 ;
                                03F7   928 ; INPUT PARAMETERS:
                                03F7   929 ;
                                03F7   930 ;       R5      Address of failover control block
                                03F7   931 ;
                                03F7   932 ; OUTPUT PARAMETERS:
                                03F7   933 ;
                                03F7   934 ;       None
                                03F7   935 ;
                                03F7   936 ; SIDE EFFECTS:
                                03F7   937 ;
                                03F7   938 ;       R0 - R5 are not preserved
                                03F7   939 ;--
                                03F7   940
                                03F7   941 LCK$REBUILD_RSBS::
                0004'CF 8ED0    03F7   942          POPL    W^RETURN_ADDR           ; Save return address
                OFE0 8F   BB    03FC   943          PUSHR   #^M<R5,R6,R7,R8,R9,R10,R11>
                                0400   944
                                0400   945          ; Loop through all RSBs in the resource hash table.  For each
                                0400   946          ; RSB, process it's locks.
                                0400   947
5A   01   00000000'GF    78     0400   948          ASHL    G^LCK$GL_HTBLCNT,#1,R10 ; Get size of hash table
     5B   00000000'GF    D0     0408   949          MOVL    G^LCK$GL_HASHTBL,R11    ; Get address of hash table
                                040F   950
                                040F   951 10$:     ; Start on next hash chain
                                040F   952
                    58   8B     DE     040F   953          MOVAL   (R11)+,R8               ; Get address of next list head
                                0412   954
                                0412   955 20$:     ; Get next RSB in this hash chain.
                                0412   956
                    58   68     D0     0412   957          MOVL    (R8),R8                 ; Get address of next RSB
                         04     13     0415   958          BEQL    30$                     ; Reached end of chain
                         31     10     0417   959          BSBB    PROCESS_RSB             ; Process it
                         F7     11     0419   960          BRB     20$                     ; Repeat
                                041B   961
                                041B   962 30$:     ; Finished one complete hash chain.  Fork to allow SCS to get
                                041B   963          ; a chance to execute and then proceed to next hash chain.
                                041B   964
                    53   5A     7D     041B   965          MOVQ    R10,R3                  ; Save size and address of hash table
                    OFE0 8F     BA     041E   966          POPR    #^M<R5,R6,R7,R8,R9,R10,R11>
                         08     90     0422   967          MOVB    #IPL$_SYNCH,-           ; Store fork IPL
                           0B   A5     0424   968                  FKB$B_FIPL(R5)
                                0426   969          FORK
```

```
                FBD1'  30   042A   970          BSBW    CNX$CHECK_FAILOVER      ; Check for another failover
0FE0 8F         BB     042F   971          PUSHR   #^M<R5,R6,R7,R8,R9,R10,R11>
5A   53         7D     0433   972          MOVQ    R3,R10                  ; Restore size and address of table
     5A         D7     0436   973          DECL    R10                     ; Decr. count of hash chains
     D5         14     0438   974          BGTR    10$                     ; Repeat
                       043A   975
0FE0 8F         BA     043A   976          POPR    #^M<R5,R6,R7,R8,R9,R10,R11>
0004'DF         17     043E   977          JMP     @W^RETURN_ADDR          ; Return to caller via saved ret. addr.
```

```
                                0442   979          .SBTTL  PROCESS_RSB - Process a single RSB during failover
                                0442   980
                                0442   981 ;++
                                0442   982 ; FUNCTIONAL DESCRIPTION:
                                0442   983 ;
                                0442   984 ;       This routine is called during failover to process a single RSB.
                                0442   985 ;       It recomputes the group grant modes, blocking AST count, queues
                                0442   986 ;       blocking ASTs and grants locks, where possible
                                0442   987 ;
                                0442   988 ; CALLING SEQUENCE:
                                0442   989 ;
                                0442   990 ;       BSBW    PROCESS_RSB
                                0442   991 ;       IPL must be at IPL$_SYNCH
                                0442   992 ;
                                0442   993 ; INPUT PARAMETERS:
                                0442   994 ;
                                0442   995 ;       R8      Address of RSB to be processed
                                0442   996 ;
                                0442   997 ; OUTPUT PARAMETERS:
                                0442   998 ;
                                0442   999 ;       None
                                0442  1000 ;
                                0442  1001 ; SIDE EFFECTS:
                                0442  1002 ;
                                0442  1003 ;       R0 - R7, and R9 destroyed
                                0442  1004 ;--
                                0442  1005
                                0442  1006 CSID_ERROR:
                                0442  1007         BUG_CHECK       LOCKMGRERR,FATAL; CSID in RSB not valid or not equal
                                0446  1008                                         ; to parent RSB's CSID
                                0446  1009
                                0446  1010 STATE_NOTZERO:
                                0446  1011         BUG_CHECK       LOCKMGRERR,FATAL;Rebuild state is not 0 or 3
                                044A  1012
                                044A  1013 PROCESS_RSB:
                                044A  1014
                                044A  1015         ; This routine does two functions, depending on the value of
                                044A  1016         ; LCK$GB_REBLD_STATE.  If it's 3 then we do a complete rebuild
                                044A  1017         ; of the RSB.  If it's 0 then we simply try to grant waiting
                                044A  1018         ; locks
                                044A  1019
03   00000000'GF    91   044A  1020         CMPB    G^LCK$GB_REBLD_STATE,#3 ; What phase of failover are we in?
           15    13   0451  1021         BEQL    5$                      ; Do the complete rebuild
     00000000'GF    95   0453  1022         TSTB    G^LCK$GB_REBLD_STATE    ; Verify state is 0
           EB    12   0459  1023         BNEQ    STATE_NOTZERO           ; Error
        38 A8    D5   045B  1024         TSTL    RSB$L_CSID(R8)          ; Only regrant locks if we are
           01    13   045E  1025         BEQL    3$                      ; mastering this resource
                 05   0460  1026         RSB
 55    0C A8    9A   0461  1027 3$:      MOVZBL  RSB$B_GGMODE(R8),R5     ; Get group grant mode
         00E6    31   0465  1028         BRW     90$                     ; Just grant locks
                      0468  1029
                      0468  1030 5$:      ; Verify that the CSID in the RSB is valid and equal to the
                      0468  1031         ; CSID in the parent RSB.
                      0468  1032         ; Then loop through all locks on all three queues.  As we do this
                      0468  1033         ; we will compute new group grant and conversion grant modes,
                      0468  1034         ; a new BLKASTCNT, send blocking ASTs, etc.
                      0468  1035         ; R9 is used to maintain a count of the number of locks on the
```

```
                          0468  1036          ; resource.  This will be used later, just for an integrity check.
                          0468  1037
                          0468  1038          ASSUME  RSB$L_CVTQFL  EQ  RSB$L_GRQFL+8
                          0468  1039          ASSUME  RSB$L_WTQFL   EQ  RSB$L_CVTQFL+8
                          0468  1040
      38 A8    B5    0468  1041          TSTW    RSB$L_CSID(R8)          ; Make sure CSID is not still -1
         D5    19    046B  1042          BLSS    CSID_ERROR              ; Error
50    48 A8    D0    046D  1043          MOVL    RSB$L_PARENT(R8),R0     ; Get address of parent RSB
         07    13    0471  1044          BEQL    8$                      ; No parent
      38 A8    D1    0473  1045          CMPL    RSB$L_CSID(R8),-        ; Verify this CSID matches parent's
      38 A0          0476  1046                  RSB$L_CSID(R0)          ; CSID
         C8    12    0478  1047          BNEQ    CSID_ERROR              ; Error!
         55    D4    047A  1048 8$:      CLRL    R5                      ; Initialize group grant mode
         59    D4    047C  1049          CLRL    R9                      ; Initialize number of locks on resource
      42 A8    B4    047E  1050          CLRW    RSB$W_BLKASTCNT(R8)     ; Initialize blocking AST count
   54    03    D0    0481  1051          MOVL    #3,R4                   ; Process 3 queues
57    10 A8    DE    0484  1052          MOVAL   RSB$L_GRQFL(R8),R7      ; Address of first queue
                          0488  1053
                          0488  1054 10$:     ; Start processing next lock queue
                          0488  1055
   56    57    D0    0488  1056          MOVL    R7,R6                   ; Use R6, save queue header in R7
                          048B  1057
                          048B  1058 20$:     ; Process next lock in this queue
                          048B  1059
   56    66    D0    048B  1060          MOVL    (R6),R6                 ; Get next lock
   57    56    D1    048E  1061          CMPL    R6,R7                   ; Reached end of queue?
         03    12    0491  1062          BNEQ    30$                     ; No
      0091    31    0493  1063          BRW     80$                     ; Yes, move to next queue
   56    38    C2    0496  1064 30$:     SUBL    #LKB$L_SQFL,R6          ; Point to start of LKB
         59    D6    0499  1065          INCL    R9                      ; Incr. count of locks
                          049B  1066
                          049B  1067          ; Do the following code only for the granted queue
                          049B  1068
      03    54    D1    049B  1069          CMPL    R4,#3                   ; Are we doing the granted queue?
            0E    19    049E  1070          BLSS    50$                     ; No
         36 A6    91    04A0  1071          CMPB    LKB$B_STATE(R6),-       ; Yes, is state = GRANTED?
            01          04A3  1072                  #LKB$K_GRANTED
            08    12    04A4  1073          BNEQ    50$                     ; No
         20 A6    D5    04A6  1074          TSTL    LKB$L_BLKASTADR(R6)     ; Is a blocking AST specified?
            03    13    04A9  1075          BEQL    50$                     ; No
         42 A8    B6    04AB  1076          INCW    RSB$W_BLKASTCNT(R8)     ; Yes, incr. blocking AST count
                          04AE  1077
                          04AE  1078 50$:     ; Do the following code only for the granted and conversion queues
                          04AE  1079
      01    54    D1    04AE  1080          CMPL    R4,#1                   ; Are we doing the waiting queue?
            15    15    04B1  1081          BLEQ    60$                     ; Yes
   51    35 A6    9A    04B3  1082          MOVZBL  LKB$B_GRMODE(R6),R1     ; No, get granted mode
63 00000000'GF45    51  E1    04B7  1083          BBC     R1,G^LCK$COMPAT_TBL[R5],75$ ; Verify it's compatible
      55    51    D1    04C0  1084          CMPL    R1,R5                   ; Is it bigger than the current
            03    1B    04C3  1085          BLEQU   60$                     ; group grant mode?
      55    51    D0    04C5  1086          MOVL    R1,R5                   ; Yes, it becomes new gg mode
                          04C8  1087
                          04C8  1088 60$:     ; Do the following code only for the conversion and waiting queues
                          04C8  1089          ; and only if this resource is managed on this system
                          04C8  1090
      03    54    D1    04C8  1091          CMPL    R4,#3                   ; Are we doing the granted queue?
            50    18    04CB  1092          BGEQ    70$                     ; Yes
```

```
                 38 A8    D5  04CD  1093          TSTL    RSB$L_CSID(R8)          ; Are we managing this resource?
                    4B    12  04D0  1094          BNEQ    70$                     ; No
                          04D2  1095
                          04D2  1096          ; Queue blocking ASTs if necessary.  Don't queue blocking ASTs
                          04D2  1097          ; on behalf of locks in any of the SCS or response states.
                          04D2  1098
                          04D2  1099          DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
                          04D2  1100                  <-
                          04D2  1101                  <CONVERT,65$>,-
                          04D2  1102                  <WAITING,65$>,-
                          04D2  1103                  >
                       3F 11    04DC  1104          BRB     70$                     ; Ignore other states
                       A8 11    04DE  1105 63$:     BRB     10$
                 42 A8    B5  04E0  1106 65$:     TSTW    RSB$W_BLKASTCNT(R8)     ; Anyone want a blocking AST?
                       0E 13    04E3  1107          BEQL    68$                     ; No
                   0FF0 8F    BB  04E5  1108          PUSHR   #^M<R4,R5,R6,R7,R8,R9,R10,R11>
              00000000'GF     16  04E9  1109          JSB     G^LCK$QUEUE_BLOCKAST    ; Queue blocking ASTs
                   0FF0 8F    BA  04EF  1110          POPR    #^M<R4,R5,R6,R7,R8,R9,R10,R11>
                          04F3  1111
                          04F3  1112 68$:     ; Insert waiting locks on the timeout queue (if not already on it).
                          04F3  1113          ; We've already determined that the lock is mastered on this system
                          04F3  1114          ; and it is in either CONVERT or WAITING state.
                          04F3  1115
                       09 E0    04F3  1116          BBS     #LCK$V_NODLCKWT,-       ; Don't insert if no deadlock wait
                 25 28 A6       04F5  1117                  LKB$W_FLAGS(R6),70$     ; is specified
           50    00000000'GF     D0  04F8  1118          MOVL    G^LCK$GL_WAITTIME,R0    ; Get lock wait time
                       1C 13    04FF  1119          BEQL    70$                     ; Deadlock checking is disabled
                       06 E2    0501  1120          BBSS    #LKB$V_TIMOUTQ,-        ; Branch if already on the queue;
                    17 2A A6    0503  1121                  LKB$W_STATUS(R6),70$    ; set bit otherwise
              00000000'GF  50   C1  0506  1122          ADDL3   R0,G^EXE$GL_ABSTIM,-    ; Add wait time to current time to
                       18 A6    050D  1123                  LKB$L_DUETIME(R6)       ; get duetime
                       4E A6 94 050F  1124          CLRB    LKB$B_TSLT(R6)          ; Init. timestamp lifetime
           50    00000000'GF     DE  0512  1125          MOVAL   G^LCK$GL_TIMOUTQ,R0
                          66 0E    0519  1126          INSQUE  LKB$L_ASTQFL(R6),-      ; Insert lock on end of timeout queue
                       04 B0    051B  1127                  @4(R0)
                          051D  1128
                    56    38 C0  051D  1129 70$:     ADDL    #LKB$L_SQFL,R6          ; Point to queue links
                       FF68 31    0520  1130          BRW     20$                     ; Go to next LKB in this queue
                          0523  1131
                          0523  1132 75$:     BUG_CHECK       LOCKMGRERR,FATAL; Incompatible granted locks
                          0527  1133                                          ; or resource has no locks
                          0527  1134                                          ; and is not a directory entry
                          0527  1135 80$:     ; Proceed to next queue
                          0527  1136
                    57    08 C0  0527  1137          ADDL    #8,R7                   ; R7 points to next queue header
                       B1 54 F5    052A  1138          SOBGTR  R4,63$                  ; Repeat for all three queues
                          052D  1139
                          052D  1140          ; Store newly computed group grant mode, regardless of
                          052D  1141          ; who is mastering resource.  Also verify that there are
                          052D  1142          ; some locks on this resource.  The only tolerable case of no
                          052D  1143          ; locks is if this is a directory entry resource.
                          052D  1144
                          052D  1145          ASSUME  RSB$V_DIRENTRY  EQ  0
                          052D  1146
                 0C A8    55  90  052D  1147          MOVB    R5,RSB$B_GGMODE(R8)     ; Store group grant mode
                 0D A8    55  90  0531  1148          MOVB    R5,RSB$B_CGMODE(R8)     ; Store conversion grant mode
                          59  D5  0535  1149          TSTL    R9                      ; Are there any locks?
```

```
                     04   12  0537  1150          BNEQ    85$                     ; Yes
                 E6 0E A8   E9  0539  1151          BLBC    RSB$W_STATUS(R8),75$    ; Error if not a directory entry
                            0053D  1152
                            0053D  1153 85$:      ; Do the following code only if we are managing this resource
                            0053D  1154
                        38 A8   D5  0053D  1155          TSTL    RSB$L_CSID(R8)          ; Is resource managed locally?
                           30   12  0540  1156          BNEQ    100$                    ; No
                                    0542  1157
                                    0542  1158          ; Invalidate value block if group grant mode is not greater
                                    0542  1159          ; than CR mode.
                                    0542  1160
                     01    55   91  0542  1161          CMPB    R5,#LCK$K_CRMODE        ; Is group grant mode greater than CR?
                           07   1A  0545  1162          BGTRU   90$                     ; Yes
                           02   A8  0547  1163          BISW    #RSB$M_VALINVLD,-       ; No, set value block invalid bit
                        0E A8       0549  1164                  RSB$W_STATUS(R8)
                        3C A8   D6  054B  1165          INCL    RSB$L_VALSEQNUM(R8)     ; Incr. value block seq. number
                                    054E  1166
                                    054E  1167 90$:      ; Try granting locks.  Note that we may have to
                                    054E  1168          ; temporarily change LCK$GB_STALLREQS in order
                                    054E  1169          ; to get LCK$GRANTCVTS to grant locks.  If LCK$GB_REBLD_STATE is
                                    054E  1170          ; set to 3 we temporarily set LCK$GB_STALLREQS to +1.  Note that we
                                    054E  1171          ; cannot suspend this thread of execution at IPL$_SYNCH while
                                    054E  1172          ; LCK$GB_STALLREQS is changed.
                                    054E  1173
             57   00000000'GF   98  054E  1174          CVTBL   G^LCK$GB_STALLREQS,R7   ; Fetch stall flag and save it
             03   00000000'GF   91  0555  1175          CMPB    G^LCK$GB_REBLD_STATE,#3 ; Is rebuild state = 3?
                           07   12  055C  1176          BNEQ    95$                     ; No
             00000000'GF   01   90  055E  1177          MOVB    #1,G^LCK$GB_STALLREQS   ; Yes, store temp. stall flag
                  00000000'GF   16  0565  1178 95$:      JSB     G^LCK$GRANTCVTS         ; Try granting locks
             00000000'GF   57   90  056B  1179          MOVB    R7,G^LCK$GB_STALLREQS   ; Restore old value of stall flag
                                05  0572  1180 100$:     RSB
```

```
                     0573  1182          .SBTTL  LCK$RESUME_UNPROT - Resume processes waiting for locks
                     0573  1183
                     0573  1184 ;++
                     0573  1185 ; FUNCTIONAL DESCRIPTION:
                     0573  1186 ;
                     0573  1187 ;       These routines resume processes waiting for locks.  The
                     0573  1188 ;       processes are in MWAIT waiting for resource RSN$_CLUSTRAN.
                     0573  1189 ;       The global cell LCK$GB_STALLREQS controls which processes
                     0573  1190 ;       proceed and which go back into MWAIT.
                     0573  1191 ;
                     0573  1192 ; CALLING SEQUENCE:
                     0573  1193 ;
                     0573  1194 ;       JSB     LCK$RESUME_UNPROT - Resume processes waiting for unprotected
                     0573  1195 ;                                   locks
                     0573  1196 ;       JSB     LCK$RESUME_ALL  -   Resume processes waiting for protected
                     0573  1197 ;                                   locks
                     0573  1198 ;       JSB     LCK$STALL_ALL   -   Stall all lock requests
                     0573  1199 ;
                     0573  1200 ;       IPL must be at IPL$_SYNCH
                     0573  1201 ;
                     0573  1202 ; INPUT PARAMATERS:
                     0573  1203 ;
                     0573  1204 ;       None
                     0573  1205 ;
                     0573  1206 ; OUTPUT PARAMETERS:
                     0573  1207 ;
                     0573  1208 ;       None
                     0573  1209 ;
                     0573  1210 ; IMPLICIT OUTPUTS:
                     0573  1211 ;
                     0573  1212 ;       LCK$GB_STALLREQS - Set to 1 by LCK$RESUME_UNPROT
                     0573  1213 ;                          Set to 0 by LCK$RESUME_ALL
                     0573  1214 ;                          set to -1 by LCK$STALL_ALL
                     0573  1215 ;
                     0573  1216 ; SIDE EFFECTS:
                     0573  1217 ;
                     0573  1218 ;       R0 - R5 destroyed
                     0573  1219 ;--
                     0573  1220
                     0573  1221 LCK$RESUME_ALL::
           50   D4   0573  1222          CLRL    R0
           03   11   0575  1223          BRB     RESUME_COM
                     0577  1224
                     0577  1225 LCK$RESUME_UNPROT::
        50   01   D0 0577  1226          MOVL    #1,R0
                     057A  1227
                     057A  1228 RESUME_COM:
00000000'GF 50   90 057A  1229          MOVB    R0,G^LCK$GB_STALLREQS   ; Store new value for stall indicator
        50   0E   D0 0581  1230          MOVL    #RSN$_CLUSTRAN,R0       ; Set resource number
     00000000'GF     16 0584  1231          JSB     G^SCH$RAVAIL            ; Make it available
                     05 058A  1232          RSB
                     058B  1233
                     058B  1234 LCK$STALL_ALL::
00000000'GF 01   8E 058B  1235          MNEGB   #1,G^LCK$GB_STALLREQS   ; Store -1 for stall indicator
                     05 0592  1236          RSB
```

```
                                0593  1238          .SBTTL  LCK$SET_STATEn - Set rebuild state to specified value
                                0593  1239
                                0593  1240 ;++
                                0593  1241 ; FUNCTIONAL DESCRIPTION:
                                0593  1242 ;
                                0593  1243 ;       These routines set the rebuild state to a specified value.
                                0593  1244 ;       The purpose of the rebuild state is to guarantee that all
                                0593  1245 ;       nodes process each step in the failover table in unison
                                0593  1246 ;       without any nodes getting ahead of other nodes.  The rebuild
                                0593  1247 ;       state variable is used by the lock manager's input message
                                0593  1248 ;       dispatcher in routine DSTRLOCK.
                                0593  1249 ;
                                0593  1250 ; CALLING SEQUENCE:
                                0593  1251 ;
                                0593  1252 ;       JSB     LCK$SET_STATEn
                                0593  1253 ;
                                0593  1254 ; INPUT PARAMETERS:
                                0593  1255 ;
                                0593  1256 ;       LCK$GB_REBLD_STATE      (The old value is verified)
                                0593  1257 ;
                                0593  1258 ; OUTPUT PARAMETERS:
                                0593  1259 ;
                                0593  1260 ;       LCK$GB_REBLD_STATE      (The new value is set)
                                0593  1261 ;
                                0593  1262 ; SIDE EFFECTS:
                                0593  1263 ;
                                0593  1264 ;       R0 and R1 not preserved
                                0593  1265 ;--
                                0593  1266
                                0593  1267
                                0593  1268 LCK$SET_STATE1::
00000000'GF     01      90      0593  1269          MOVB    #1,G^LCK$GB_REBLD_STATE ; Old value can be anything
                        05      059A  1270          RSB
                                059B  1271
                                059B  1272 LCK$SET_STATE2::
          50    01      90      059B  1273          MOVB    #1,R0
                03      11      059E  1274          BRB     SET_STATE
                                05A0  1275
                                05A0  1276 LCK$SET_STATE3::
          50    02      90      05A0  1277          MOVB    #2,R0
                                05A3  1278
                                05A3  1279 SET_STATE:
51      00000000'GF     9E      05A3  1280          MOVAB   G^LCK$GB_REBLD_STATE,R1 ; Get address of state variable
          50    61      91      05AA  1281          CMPB    (R1),R0                 ; Verify old value
                03      12      05AD  1282          BNEQ    10$
                61      96      05AF  1283          INCB    (R1)                    ; Set new state
                        05      05B1  1284          RSB
                                05B2  1285
                                05B2  1286 10$:     BUG_CHECK       LOCKMGRERR,FATAL
                                05B6  1287
                                05B6  1288 LCK$SET_STATE0::
          50    03      90      05B6  1289          MOVB    #3,R0
                E8      10      05B9  1290          BSBB    SET_STATE
                61      94      05BB  1291          CLRB    (R1)
                        05      05BD  1292          RSB
                                05BE  1293
                                05BE  1294
```

```
05BE  1295
05BE  1296
05BE  1297
05BE  1298          .END
```

```
$$BASE                          = FFFFFFFF
$$DISPL                         = 00000001
$$GENSW                         = 00000001
$$HIGH                          = 00000000
$$LIMIT                         = 00000001
$$LOW                           = FFFFFFFF
$$MNSW                          = 00000001
$$MXSW                          = 00000001
BUG$_LOCKMGRERR                   ********    X   03
BUG$_RESEXH                       ********    X   03
CA$_MEASURE                     = 00000002
CDRP$L_MSGBLD                   = 0000004C
CDRP$L_VAL1                     = 0000002C
CDRP$L_VAL2                     = 00000030
CDRP$L_VAL3                     = 00000034
CDRP$L_VAL8                     = 00000048
CHECK_FAILOVER                    000002BD R      03
CHECK_FAILOVER2                   000002C1 R      03
CLU$GL_CLUB                       ********    X   03
CLUB$B_CLUFCB                   = 0000010C
CLUB$W_MEMSEQ                   = 000000AC
CNX$ALLOC_WARMCDRP                ********    X   03
CNX$CHECK_FAILOVER                ********    X   03
CNX$DEALL_WARMCDRP_CSB            ********    X   03
CNX$END_FAILOVER                  ********    X   03
CNX$RESOURCE_CHECK                ********    X   03
CNX$SEND_MSG_CSB                  ********    X   03
CSB$L_CLUB                      = 00000064
CSB$L_CSID                      = 0000004C
CSID_ERROR                        00000442 R      03
CURR_LOCKID                       00000000 R      02
DYN$C_RSB                       = 00000036
EXE$DEANONPAGED                   ********    X   03
EXE$FORK                          ********    X   03
EXE$FORK_WAIT                     ********    X   03
EXE$GL_ABSTIM                     ********    X   03
FKB$B_FIPL                      = 0000000B
IPL$_SCS                        = 00000008
IPL$_SYNCH                      = 00000008
LCK$BLD_REBLDLOCK                 ********    X   03
LCK$CHECK_DIRENTRY                00000369 RG     03
LCK$COMPAT_TBL                    ********    X   03
LCK$DEALLOC_LKB                   ********    X   03
LCK$DEALLOC_RSB                   ********    X   03
LCK$GB_REBLD_STATE                ********    X   03
LCK$GB_STALLREQS                  ********    X   03
LCK$GL_DIRVEC                     ********    X   03
LCK$GL_HASHTBL                    ********    X   03
LCK$GL_HTBLCNT                    ********    X   03
LCK$GL_IDTBL                      ********    X   03
LCK$GL_MAXID                      ********    X   03
LCK$GL_TIMOUTQ                    ********    X   03
LCK$GL_TS_CSID                    00000008 RG     02
LCK$GL_WAITTIME                   ********    X   03
LCK$GQ_BITMAP_EXP                 ********    X   03
LCK$GRANTCVTS                     ********    X   03
LCK$INIT_REBUILD                  00000000 RG     03
LCK$K_CRMODE                    = 00000001
LCK$MARK_FOR_RESEND               000003BC R      03
LCK$QUEUE_BLOCKAST                ********      X 03
LCK$REBLD_LOCK                    000002EE RG     03
LCK$REBUILD_LKBS                  000000FD RG     03
LCK$REBUILD_RSBS                  000003F7 RG     03
LCK$RESUME_ALL                    00000573 RG     03
LCK$RESUME_UNPROT                 00000577 RG     03
LCK$SET_STATE0                    000005B6 RG     03
LCK$SET_STATE1                    00000593 RG     03
LCK$SET_STATE2                    0000059B RG     03
LCK$SET_STATE3                    000005A0 RG     03
LCK$STALL_ALL                     0000058B RG     03
LCK$V_CONVERT                   = 00000001
LCK$V_NODLCKWT                  = 00000009
LKB$B_GRMODE                    = 00000035
LKB$B_STATE                     = 00000036
LKB$B_TSLT                      = 0000004E
LKB$B_TYPE                      = 0000000A
LKB$K_CONVERT                   = 00000000
LKB$K_GRANTED                   = 00000001
LKB$K_RETRY                     = FFFFFFFE
LKB$K_RSPDOLOCL                 = FFFFFFF9
LKB$K_RSPGRANTD                 = FFFFFFFA
LKB$K_RSPNOTQED                 = FFFFFFFC
LKB$K_RSPQUEUED                 = FFFFFFFB
LKB$K_RSPRESEND                 = FFFFFFF8
LKB$K_SCSWAIT                   = FFFFFFFD
LKB$K_WAITING                   = FFFFFFFF
LKB$L_ASTQFL                    = 00000000
LKB$L_BLKASTADR                 = 00000020
LKB$L_DUETIME                   = 00000018
LKB$L_PARENT                    = 00000048
LKB$L_PID                       = 0000000C
LKB$L_REMLKID                   = 00000054
LKB$L_RSB                       = 00000050
LKB$L_SQFL                      = 00000038
LKB$M_ASYNC                     = 00000004
LKB$M_RESEND                    = 00000400
LKB$M_TIMOUTQ                   = 00000040
LKB$V_MSTCPY                    = 00000004
LKB$V_RESEND                    = 0000000A
LKB$V_TIMOUTQ                   = 00000006
LKB$W_FLAGS                     = 00000028
LKB$W_REFCNT                    = 0000004C
LKB$W_RQSEQNM                   = 00000010
LKB$W_STATUS                    = 0000002A
LKMSG$B_LCKSTATE                = 0000006A
LKMSG$B_LSTATUS                 = 00000068
LKMSG$B_RSTATUS                 = 00000069
LKMSG$B_STATE                   = 0000001E
LKMSG$L_CSID                    = 00000018
LKMSG$L_MSTLKID                 = 00000010
LKMSG$L_VALSEQALT               = 00000064
LKMSG$Q_VALBLKALT               = 00000054
LKMSG$W_MEMSEQ                  = 0000000C
LKMSG$W_RQSEQALT                = 00000050
```

```
LOCKS_DONE                      000000F5 R      03
NEXT_LOCKID                     0000010A R      03
NEXT_LOCKID_SAVE                00000106 R      03
PMS$GL_DIR_OUT                  ******** X      03
PMS$GL_ENQNEW_OUT               ******** X      03
PROCESS_RSB                     0000044A R      03
REBLD_DOLOCL                    0000028A R      03
REBLD_GRANTD                    00000298 R      03
REBLD_NOTQED                    00000286 R      03
REBLD_RESEND                    000002CB R      03
REBLD_RETRY                     00000293 R      03
REBUILD                         000001C3 R      03
RESUME_COM                      0000057A R      03
RETURN_ADDR                     00000004 R      02
RSB$B_CGMODE                  = 0000000D
RSB$B_GGMODE                  = 0000000C
RSB$L_CSID                    = 00000038
RSB$L_CVTQFL                  = 00000018
RSB$L_GRQFL                   = 00000010
RSB$L_HSHCHNBK                = 00000004
RSB$L_PARENT                  = 00000048
RSB$L_VALSEQNUM               = 0000003C
RSB$L_WTQFL                   = 00000020
RSB$M_DIRENTRY                = 00000001
RSB$M_VALINVLD                = 00000002
RSB$Q_VALBLK                  = 00000028
RSB$V_DIRENTRY                = 00000000
RSB$V_VALINVLD                = 00000001
RSB$W_BLKASTCNT               = 00000042
RSB$W_HASHVAL                 = 00000044
RSB$W_REFCNT                  = 00000040
RSB$W_RQSEQNM                 = 00000046
RSB$W_STATUS                  = 0000000E
RSN$_CLUSTRAN                 = 0000000E
SAME_LOCKID                     00000119 R      03
SCH$GL_PCBVEC                   ******** X      03
SCH$RAVAIL                      ******** X      03
SET_STATE                       000005A3 R      03
SS$_INSFMEM                     ******** X      03
STATE_NOTZERO                   00000446 R      03
STORE_CSID                      000002AA R      03
```

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

```
PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           00000000  (    0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
$$$040                          0000000C  (   12.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC LONG
$$$020                          000005BE  ( 1470.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
```

### Performance indicators

| Phase | Page faults | CPU Time | Elapsed Time |
|---|---|---|---|
| Initialization | 29 | 00:00:00.03 | 00:00:01.94 |
| Command processing | 113 | 00:00:00.46 | 00:00:02.34 |
| Pass 1 | 421 | 00:00:11.68 | 00:00:40.93 |
| Symbol table sort | 0 | 00:00:01.46 | 00:00:06.00 |
| Pass 2 | 245 | 00:00:02.91 | 00:00:08.54 |
| Symbol table output | 19 | 00:00:00.11 | 00:00:00.11 |
| Psect synopsis output | 2 | 00:00:00.02 | 00:00:00.02 |
| Cross-reference output | 0 | 00:00:00.00 | 00:00:00.00 |
| Assembler run totals | 831 | 00:00:16.69 | 00:01:00.31 |

The working set limit was 1800 pages.
96403 bytes (189 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1373 non-local and 70 local symbols.
1298 source lines were read in Pass 1, producing 22 object records in Pass 2.
32 pages of virtual memory were used to define 30 macros.

### Macro library statistics

| Macro library name | Macros defined |
|---|---|
| _$255$DUA28:[SYSLOA.OBJ]CLUSTER.MLB;1 | 1 |
| _$255$DUA28:[SYS.OBJ]LIB.MLB;1 | 17 |
| _$255$DUA28:[SYSLIB]STARLET.MLB;2 | 6 |
| TOTALS (all libraries) | 24 |

1477 GETS were required to define 24 macros.

There were no errors, warnings or information messages.

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

MOUNTVER
LIS

OPDRVWS1
LIS

QUORUM
LIS

OPDRV790
LIS

OPDRIVER
CIS

REBLDLOCK
LIS

UTILKEY.LIS

TECOLBR.LIS

TECOCMD.LIS

SCSLOA.LIS

TECO

TECO.MAP

TECONAT.LIS