```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 000000000        AAAAAAAAA
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 000000000        AAAAAAAAA
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  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
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     000000000     AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     000000000     AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     000000000     AAA         AAA
```

**FILE**ID**DSTRLOCK

```
DDDDDDDD      SSSSSSSS  TTTTTTTTTT  RRRRRRRR    LL            OOOOOO      CCCCCCCC  KK      KK
DDDDDDDD      SSSSSSSS  TTTTTTTTTT  RRRRRRRR    LL            OOOOOO      CCCCCCCC  KK      KK
DD      DD  SS              TT      RR      RR  LL          OO      OO  CC          KK      KK
DD      DD  SS              TT      RR      RR  LL          OO      OO  CC          KK      KK
DD      DD  SS              TT      RR      RR  LL          OO      OO  CC          KK    KK
DD      DD  SS              TT      RR      RR  LL          OO      OO  CC          KK    KK
DD      DD    SSSSSS        TT      RRRRRRRR    LL          OO      OO  CC          KKKKKK
DD      DD    SSSSSS        TT      RRRRRRRR    LL          OO      OO  CC          KKKKKK
DD      DD          SS      TT      RR  RR      LL          OO      OO  CC          KK    KK
DD      DD          SS      TT      RR  RR      LL          OO      OO  CC          KK    KK
DD      DD          SS      TT      RR    RR    LL          OO      OO  CC          KK      KK      ....
DD      DD          SS      TT      RR    RR    LL          OO      OO  CC          KK      KK      ....
DDDDDDDD    SSSSSSSS        TT      RR      RR  LLLLLLLLLL    OOOOOO      CCCCCCCC  KK      KK      ....
DDDDDDDD    SSSSSSSS        TT      RR      RR  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          .TITLE  DSTRLOCK - Distributed Lock Manager Loadable Code
0000          .IDENT  'V04-001'
0000
0000  ;
0000  ;****************************************************************************
0000  ;*                                                                          *
0000  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000  ;*  ALL RIGHTS RESERVED.                                                    *
0000  ;*                                                                          *
0000  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000  ;*  TRANSFERRED.                                                            *
0000  ;*                                                                          *
0000  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000  ;*  CORPORATION.                                                            *
0000  ;*                                                                          *
0000  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000  ;*                                                                          *
0000  ;*                                                                          *
0000  ;****************************************************************************
0000  ;
0000
0000  ;++
0000  ;
0000  ; FACILITY:     Executive, system services and fork level code
0000  ;
0000  ; ABSTRACT:
0000  ;       This module contains routines used to implement distributed
0000  ;       lock manager functions.
0000  ;
0000  ; ENVIRONMENT:  Kernel mode, fork level, loadable code
0000  ;
0000  ;--
0000  ;
0000  ; AUTHOR: Steve Beckhardt,       CREATION DATE: 11-Oct-1982
0000  ;
0000  ; MODIFIED BY:
0000  ;
0000  ;       V04-001 SRB0144         Steve Beckhardt                 6-Sep-1984
0000  ;               Fixed race condition in conversion code that could
0000  ;               ultimately clobber value blocks.  Also fixed bug in
0000  ;               CVTRETRY_PROC code for locks in RETRY state that are
0000  ;               ultimately not queued.
0000  ;
0000  ;       V03-023 SRB0140         Steve Beckhardt                 2-Aug-1984
0000  ;               Change SRB0136 was incomplete.  Fixed one more place
0000  ;               to bugcheck this system rather than returning an error.
0000  ;
0000  ;       V03-022 SRB0136         Steve Beckhardt                 9-Jul-1984
0000  ;               Changed handling of insufficient resources during failover
0000  ;               to bugcheck the system with insufficient resources.
```

```
0000    ;
0000    ;       V03-021 SRB0134         Steve Beckhardt                 26-Jun-1984
0000    ;               Simplified message build routine for rebuild lock message
0000    ;               to not handle transient SCS states (RSPxxx).  This is
0000    ;               a result of a corresponding change in the rebuild code
0000    ;               to put all those locks into RETRY state.
0000    ;
0000    ;       V03-020 SRB0132         Steve Beckhardt                 25-May-1984
0000    ;               Fixed bug where cleanup path from building an LKB error
0000    ;               didn't decrement parent's sub-RSB reference count.
0000    ;
0000    ;       V03-019 SRB0120         Steve Beckhardt                 20-Apr-1984
0000    ;               Fixed race conditions in LCK$RCV_RMVDIR code.
0000    ;               Fixed bug where locks in SCS states were rebuilt incorrectly.
0000    ;
0000    ;       V03-018 SRB0119         Steve Beckhardt                 2-Apr-1984
0000    ;               Added support for waiting for pool.  Fixed several bugs.
0000    ;
0000    ;       V03-017 SRB0117         Steve Beckhardt                 19-Mar-1984
0000    ;               Added support for distributed deadlock detection.
0000    ;
0000    ;       V03-016 SRB0115         Steve Beckhardt                 23-Feb-1984
0000    ;               Added support for maintaining deadlock priority field in
0000    ;               master-copy locks.  Fixed bug in conversion code.
0000    ;
0000    ;       V03-015 SRB0110         Steve Beckhardt                 27-Jan-1984
0000    ;               Added checking of MEMSEQ of incoming REBLDLOCK messages.
0000    ;               Added checking of rebuild state in message dispatching.
0000    ;
0000    ;       V03-014 SRB0108         Steve Beckhardt                 11-Jan-1984
0000    ;               Added support for new failin/out design with hashed root
0000    ;               directory.  Added support for sending HASHVAL in lock
0000    ;               messages.  Added support for maintaining EPID in locks.
0000    ;               Changed PMS counters.
0000    ;
0000    ;       V03-013 SRB0106         Steve Beckhardt                 6-Dec-1983
0000    ;               Changed LKB$L_REFCNT,RSB$L_REFCNT, and RSB$L_BLKASTCNT
0000    ;               to word fields.
0000    ;
0000    ;       V03-012 SRB0104         Steve Beckhardt                 17-Oct-1983
0000    ;               Fixed bug in LCK$SND_BLKING.
0000    ;
0000    ;       V03-011 SRB0100         Steve Beckhardt                 18-Jul-1983
0000    ;               Fixed PMS counters.  Fixed handling of receiving a lock
0000    ;               granted message to handle canceled locks with blocking ASTs.
0000    ;
0000    ;       V03-010 SRB0097         Steve Beckhardt                 23-Jun-1983
0000    ;               Added support for RECOVER bit.  Made several changes
0000    ;               to support multi-node failover.  Added support for
0000    ;               converting new locks to be system owned.
0000    ;
0000    ;       V03-009 SRB0093         Steve Beckhardt                 31-May-1983
0000    ;               Simplified use of LKB state codes.
0000    ;
0000    ;       V03-008 SRB0090         Steve Beckhardt                 18-May-1983
0000    ;               Added support for extending the lock id. table.  Added
0000    ;               support for multi-node failover.  Added PMS counters.
```

```
0000  ;               Fixed bug involving system owned locks.  Added support
0000  ;               for stalling requests during failover.
0000  ;
0000  ;       V03-007 SRB0083         Steve Beckhardt                 20-Apr-1983
0000  ;               Finished adding support for INVVALBLK and CANCEL flags.
0000  ;               Rewrote conversion code to send messages without RSPIDs
0000  ;               in some cases.
0000  ;
0000  ;       V03-006 SRB0077         Steve Beckhardt                 14-Apr-1983
0000  ;               Removed use of LKB$M_PRCCPY bit.
0000  ;
0000  ;       V03-005 SRB0073         Steve Beckhardt                 29-Mar-1983
0000  ;               Changed references to CSB to CSID in LKB and RSB.
0000  ;               Started adding support for $DEQ flags INVVALBLK and CANCEL.
0000  ;
0000  ;       V03-004 SRB0069         Steve Beckhardt                 7-Mar-1983
0000  ;               Changed HALT instructions to BUG_CHECKs.  Changed handling
0000  ;               of value blocks on conversions to return value block on
0000  ;               conversions to same lock mode.
0000  ;
0000  ;       V03-003 SRB0063         Steve Beckhardt                 21-Jan-1983
0000  ;               Made a number of bug fixes.
0000  ;
0000  ;       V03-002 SRB0062         Steve Beckhardt                 11-Jan-1983
0000  ;               Changed BSBW to JSB.
0000  ;
0000  ;
```

```
    0000         .SBTTL  DECLARATIONS
    0000   ;
    0000   ; INCLUDE FILES:
    0000   ;
    0000
    0000   ;
    0000   ; MACROS:
    0000   ;
    0000
    0000         $ACBDEF                              ; ACB offsets
    0000         $CADEF                               ; Conditional assembly switches
    0000         $CDRPDEF                             ; CDRP offsets
    0000         $CLSMSGDEF                           ; Cluster message offsets
    0000         $CLUBDEF                             ; CLUB offsets
    0000         $CSBDEF                              ; CSB offsets
    0000         $DYNDEF                              ; Data structure names
    0000         $FKBDEF                              ; Fork block offsets
    0000         $IPLDEF                              ; IPL definitions
    0000         $LCKDEF                              ; LCK definitions
    0000         $LKBDEF                              ; LKB offsets
    0000         $PCBDEF                              ; PCB offsets
    0000         $PRIDEF                              ; Priority definitions
    0000         $PSLDEF                              ; PSL definitions
    0000         $RSBDEF                              ; RSB offsets
    0000         $RSNDEF                              ; Resource numbers
    0000
    0000   ;
    0000   ; EQUATED SYMBOLS:
    0000   ;
    0000
    0000
    0000   ;
    0000   ; OWN STORAGE:
    0000   ;
    0000
00000000         .PSECT  $$$020
    0000
    0000   ;***********************************************************************
    0000   ;
    0000   ; NOTE: The following assumption is in effect for this entire module.
    0000   ;
    0000   ;***********************************************************************
    0000
    0000         ASSUME  IPL$_SYNCH  EQ  IPL$_SCS
```

```
         0000   ;
         0000   ;                         CONVERSION RESPONSE TABLE
         0000   ;
         0000   ; This table indicates which conversions require a response message .
         0000   ;
         0000   ;                                       TO
         0000   ;
         0000   ;              |  NL   |  CR   |  CW   |  PR   |  PW   |  EX
         0000   ;              | NL/SW | R/SW  | W/SW  | R/SR  | W/SR  | W/NL
         0000   ;     ---------+-------+-------+-------+-------+-------+-------
         0000   ;     NL NL/SW |  yes  |  yes  |  yes  |  yes  |  yes  |  yes
         0000   ;     ---------+-------+-------+-------+-------+-------+-------
         0000   ;     CR  R/SW |  no   |  yes  |  yes  |  yes  |  yes  |  yes
         0000   ;     ---------+-------+-------+-------+-------+-------+-------
         0000   ;     CW  W/SW |  no   |  no   |  no   |  yes  |  yes  |  yes
         0000   ; FROM---------+-------+-------+-------+-------+-------+-------
         0000   ;     PR  R/SR |  no   |  no   |  yes  |  no   |  yes  |  yes
         0000   ;     ---------+-------+-------+-------+-------+-------+-------
         0000   ;     PW  W/SR |  no   |  no   |  no   |  no   |  no   |  yes
         0000   ;     ---------+-------+-------+-------+-------+-------+-------
         0000   ;     EX  W/NL |  no   |  no   |  no   |  no   |  no   |  no
         0000
         0000
         0000   LCK$CVTRSP_TBL::
         0000
    3F   0000           .BYTE   ^B 111111
    3E   0001           .BYTE   ^B 111110
    38   0002           .BYTE   ^B 111000
    34   0003           .BYTE   ^B 110100
    20   0004           .BYTE   ^B 100000
    00   0005           .BYTE   ^B 000000
```

```
                        0006         .SBTTL  LCK$DISPATCH - Dispatch incoming lock message
                        0006
                        0006 ;++
                        0006 ; FUNCTIONAL DESCRIPTION:
                        0006 ;
                        0006 ;       This routine dispatches incoming lock manager messages tc
                        0006 ;       the appropriate routine.  It also verifies that incoming messages
                        0006 ;       are stamped with a membership sequence number that is the same as
                        0006 ;       the current one on this system.  If not, the messages are either
                        0006 ;       returned with a LKB$K_RETRY response (RSPID messages) or are ignored
                        0006 ;       (non-RSPID messages).
                        0006 ;
                        0006 ; CALLING SEQUENCE:
                        0006 ;
                        0006 ;       JSB     LCK$DISPATCH    (called from connection manager received
                        0006 ;                                message routine)
                        0006 ;       IPL is at IPL$_SYNCH
                        0006 ;
                        0006 ; INPUT PARAMETERS:
                        0006 ;
                        0006 ;       R2      Address of message
                        0006 ;       R3      Address of CSB
                        0006 ;       R4      Address of PDT
                        0006 ;       R5      Address of CDRP (if this message needs a response)
                        0006 ;
                        0006 ; OUTPUT PARAMETERS:
                        0006 ;
                        0006 ;       None
                        0006 ;
                        0006 ; SIDE EFFECTS:
                        0006 ;
                        0006 ;       R0 - R5 destroyed
                        0006 ;--
                        0006
                        0006 LCK$DISPATCH::
                        0006
   50    64 A3     D0   0006         MOVL    CSB$L_CLUB(R3),R0          ; Get address of CLUB
       00AC C0     A3   000A         SUBW3   CLUB$W_MEMSEQ(R0),-        ; Compare sequence numbers
   50    0C A2          000E                 LKMSG$W_MEMSEQ(R2),R0
            58     19   0011         BLSS    50$                        ; Message is old, ignore or retry
            2A     14   0013         BGTR    20$                        ; MEMSEQ must be in error!
50  00000000'GF    9A   0015         MOVZBL  G^LCK$GB_REBLD_STATE,R0    ; Are we in any phase of failover?
            25     12   001C         BNEQ    30$                        ; Yes
                        001E         DISPATCH        CLSMSG$B_FUNC(R2),TYPE=B,PREFIX=LKMSG$K_,-
                        001E                 <-
                        001E                 <NEWLOCK,LCK$RCV_LOCKREQ>,-        ; New lock request
                        001E                 <GRANTED,LCK$RCV_GRANTED>,-        ; Lock granted message
                        001E                 <DEQ,LCK$RCV_DEQ>,-                ; Dequeue request
                        001E                 <RMVDIR,LCK$RCV_RMVDIR>,-          ; Remove directory entry
                        001E                 <BLKING,LCK$RCV_BLKING>,-          ; Lock blocking message
                        001E                 <CVTLCKM,LCK$RCV_CVTMSG>,-         ; Convert lock message
                        001E                 <CVTLCKR,LCK$RCV_CVTREQ>,-         ; Convert lock req. (needs RSP)
                        001E                 <TSRQST,LCK$RCV_TIMESTAMP_RQST>,-  ; Time stamp request
                        001E                 <SRCHDLCK,LCK$RCV_SRCHDLCK>,-      ; Search for deadlock
                        001E                 <DLCKFND,LCK$RCV_DLCKFND>,-        ; Deadlock found
                        001E                 <REDO_SRCH,LCK$RCV_REDO_SRCH>,-    ; Redo deadlock search
                        001E                 >
```

```
                        003B
                        003B              BUG_CHECK      LOCKMGRERR,FATAL        ; Illegal function code
                        003F
                        003F     20$:     BUG_CHECK      LOCKMGRERR,FATAL        ; MEMSEQ error
                        0043
                        0043     30$:     ; We are in the middle of a membership state change.  Dispatch
                        0043              ; on phase of state change.
                        0043
                        0043              DISPATCH       R0,TYPE=B,-
                        0043                        <-
                        0043                        <2,35$>,-
                        0043                        <3,45$>,-
                        0043                        >
                        004B
                        004B              BUG_CHECK      LOCKMGRERR,FATAL        ; Unrecognized state or
                        004F                                                     ; no messages allowed in
                        004F                                                     ; this state
                        004F
                        004F     35$:     DISPATCH       CLSMSG$B_FUNC(R2),TYPE=B,PREFIX=LKMSG$K_,-
                        004F                        <-
                        004F                        <REBLDLOCK,LCK$RCV_LOCKREQ>,-   ; Rebuild lock (failover)
                        004F                        >
                        0056
                        0056              BUG_CHECK      LOCKMGRERR,FATAL        ; Illegal function code
                        005A
                        005A     45$:     DISPATCH       CLSMSG$B_FUNC(R2),TYPE=B,PREFIX=LKMSG$K_,-
                        005A                        <-
                        005A                        <GRANTED,LCK$RCV_GRANTED>,-     ; Lock granted message
                        005A                        <BLKING,LCK$RCV_BLKING>,-       ; Lock blocking message
                        005A                        >
                        0067
                        0067              BUG_CHECK      LOCKMGRERR,FATAL        ; Illegal function code
                        006B
                        006B     50$:     ; Message should either be ignored or returned with LKB$K_RETRY
                        006B              ; response (the latter if a RSPID is provided).
                        006B
          04 A2    D5   006B              TSTL    CLSMSG$L_RSPID(R2)       ; Is there a RSPID?
             03    12   006E              BNEQ    70$                      ; Yes
          FF8D'    31   0070              BRW     CNX$DEALL_MSG_BUF_CSB    ; No, deallocate message bfr and return
                        0073
                        0073     70$:     ; Send back LKB$K_RETRY response.
                        0073
   53     4C A3    D0   0073              MOVL    CSB$L_CSID(R3),R3        ; Get CSID
          FF86'    30   0077              BSBW    CNX$INIT_CDRP            ; Initialize CDRP
          FE 8F    90   007A              MOVB    #LKB$K_RETRY,-           ; Store retry response
          32 A5         007D                      CDRP$L_VAL2+2(R5)
      0A5A'CF      9E   007F              MOVAB   W^BLD_RSPMSG,-           ; Store address of message build routine
          4C A5         0083                      CDRP$L_MSGBLD(R5)
          FF78'    30   0085              BSBW    CNX$SEND_MSG_RESP        ; Send message response
   50      55      D0   0088              MOVL    R5,R0                    ; Address of CDRP
   00000000'GF     17   008B              JMP     G^EXE$DEANONPAGED        ; Deallocate it and return
```

```
                                  0091          .SBTTL  LCK$SND_CVTREQ - Send a conversion request to remote system
                                  0091
                                  0091  ;++
                                  0091  ; FUNCTIONAL DESCRIPTION:
                                  0091  ;
                                  0091  ;       This routine handles conversion requests that must be forwarded
                                  0091  ;       to the master system for this lock.  This system is the process
                                  0091  ;       system.
                                  0091  ;
                                  0091  ; CALLING SEQUENCE:
                                  0091  ;
                                  0091  ;       JMP     LCK$SND_CVTREQ (This routine does not return to its
                                  0091  ;                               caller.  Rather, it jumps to an
                                  0091  ;                               appropriate destination.)
                                  0091  ;       IPL must be at IPL$_SYNCH
                                  0091  ;
                                  0091  ; INPUT PARAMETERS:
                                  0091  ;
                                  0091  ;       R1      Requested lock mode
                                  0091  ;       R3      CSID of destination system
                                  0091  ;       R6      Address of LKB
                                  0091  ;       R7      Old status from LKB
                                  0091  ;       R8      Address of RSB
                                  0091  ;       R9      Flags
                                  0091  ;       R11     Current granted mode
                                  0091  ;
                                  0091  ; OUTPUT PARAMETERS:
                                  0091  ;
                                  0091  ;       R0      Completion code (on some exit paths)
                                  0091  ;
                                  0091  ; SIDE EFFECTS:
                                  0091  ;
                                  0091  ;       The process will go into MWAIT until the response from the remote
                                  0091  ;       system arrives.
                                  0091  ;       R1 - R5 destroyed
                                  0091  ;--
                                  0091
                                  0091          .ENABL  LSB
                                  0091
                                  0091  LCK$SND_CVTREQ::
                                  0091
                        00000002  0091          .IF NE  CA$_MEASURE
             00000000'GF      D6  0091          INCL    G^PMS$GL_ENQCVT_OUT
                                  0097          .ENDC
                                  0097
      54     00000000'GF      D0  0097          MOVL    G^SCH$GL_CURPCB,R4      ; Get our PCB address
                34 A6    51   90  009E          MOVB    R1,LKB$B_RQMODE(R6)     ; Store requested mode
    33 FF58 CF4B         51   E0  00A2          BBS     R1,LCK$CVTRSP_TBL[R11],20$ ; Branch if response needed
                                  00A9
                                  00A9          ; No response needed.  Allocate CDRP, store value block (if
                                  00A9          ; specified), and send message.
                                  00A9
                                  00A9          ASSUME  LCK$V_VALBLK  EQ  0
                                  00A9
                     FF54'    30  00A9          BSBW    CNX$ALLOC_CDRP          ; Alloc. a CDRP and convert CSID to CSB
                    30 50     E9  00AC          BLBC    R0,25$                  ; No CDRPs or CSID error
                    0A 59     E9  00AF          BLBC    R9,10$                  ; No value block
```

```
      28 A8   7D  00B2          MOVQ    RSB$Q_VALBLK(R8),-           ; Copy value block
      38 A5       00B5                  CDRP$L_VAL4(R5)
      30 A8   7D  00B7          MOVQ    RSB$Q_VALBLK+8(R8),-
      40 A5       00BA                  CDRP$L_VAL6(R5)
      60 A4   D0  00BC  10$:    MOVL    PCB$L_PID(R4),-              ; Store PID in CDRP
      48 A5       00BF                  CDRP$L_VAL8(R5)
      34 A6   B0  00C1          MOVW    LKB$B_RQMODE(R6),-           ; Store req. and granted mode in CDRP
      5C A5       00C4                  CDRP$L_VAL9(R5)
         50   D4  00C6          CLRL    R0                           ; Initialize blocking AST flag
      20 A6   D5  00C8          TSTL    LKB$L_BLKASTADR(R6)          ; Test for a blocking AST address
         02   13  00CB          BEQL    15$                          ; No blocking AST
         50   D6  00CD          INCL    R0                           ; Set blocking AST flag
5E A5    50   90  00CF  15$:    MOVB    R0,CDRP$L_VAL9+2(R5)         ; Store blocking AST flag
   50    56   D0  00D3          MOVL    R6,R0                        ; Move address of LKB
       00DC   30  00D6          BSBW    SENDCVTMSG                   ; Send message
       0084   31  00D9          BRW     CGRNTD_PROC                  ; Join granted code
                  00DC
                  00DC  20$:    ; Send a message and wait for response.
                  00DC
      FF21'   30  00DC          BSBW    CNX$ALLOC_WARMCDRP           ; Alloc. a CDRP with RSPID and cvt CSID
      52 50   E9  00DF  25$:    BLBC    R0,60$                       ; No CDRPs or CSID error
   50    56   D0  00E2          MOVL    R6,R0                        ; Move address of LKB
48 A5    54   D0  00E5          MOVL    R4,CDRP$L_VAL8(R5)           ; Store PCB address in CDRP
       00F4   30  00E9          BSBW    SENDCVTREQ                   ; Send convert request
   54  48 A5  D0  00EC          MOVL    CDRP$L_VAL8(R5),R4           ; Restore address of PCB
                  00F0
                  00F0  CVT_SCS_WAIT:
   50    0D   D0  00F0          MOVL    #RSN$_SCS,R0                 ; Go into MWAIT for resource RSN$_SCS
                  00F3
                  00F3  CVT_WAIT:
                  00F3          ; Put the process into MWAIT until the response arrives or the
                  00F3          ; resource becomes available.  Resource number is in R0.
                  00F3          ; PCB address is in R4.  If this is a system owned lock,
                  00F3          ; then the process's PID is temporarily stored in
                  00F3          ; the LKB.  This serves two purposes.  First, it allows the process
                  00F3          ; to be made executable when the response arrives and secondly, it
                  00F3          ; prevents any other processes from manipulating this lock
                  00F3          ; (VERIFYLOCKID will fail) while we wait for the response.
                  00F3          ; Note that the only two conversions that require a response message
                  00F3          ; yet will allow a system lock are a CR to CR conversion or a NL to
                  00F3          ; NL conversion and that these are always synchronously grantable.
                  00F3          ; This also occurs if we go into MWAIT for non-paged pool.
                  00F3
      60 A4   D0  00F3          MOVL    PCB$L_PID(R4),-              ; Make sure PID is stored in case lock
      0C A6       00F6                  LKB$L_PID(R6)                ; is system owned
         10   78  00F8          ASHL    #PSL$V_IPL,-                 ; Create a PSL on stack with IPL
   7E    02       00FA                  #IPL$_ASTDEL,-(SP)           ; set to ASTDEL
 00000000'GF  16  00FC          JSB     G^SCH$RWAIT                  ; Wait
                  0102
                  0102          ; Upon reawakening the LKB$B_STATE field tells us what action
                  0102          ; to perform.  But we first must determine if we are stalling requests.
                  0102
                  0102          SETIPL  #IPL$_SYNCH                  ; Raise IPL
 00000000'GF  95  0105          TSTB    G^LCK$GB_STALLREQS           ; Are we stalling?
         1B   12  010B          BNEQ    40$                          ; Yes
03 59    06   E1  010D  30$:    BBC     #LCK$V_CVTSYS,R9,33$         ; Branch if not system owned
      0C A6   D4  0111          CLRL    LKB$L_PID(R6)                ; Clear PID of system owned lock
```

```
                        0114
                        0114  33$:    DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
                        0114                  <-
                        0114                  <RSPGRANTD,CGRNTD_PROC>,-       ; Granted
                        0114                  <RSPQUEUED,CVTQED_PROC>,-       ; Queued
                        0114                  <RSPNOTQED,CNOTQD_PROC>,-       ; Not queued
                        0114                  <SCSWAIT,CVT_SCS_WAIT>,-        ; Spurious wakeup
                        0114                  <RETRY,CVTRETRY_PROC>,-         ; Retry
                        0114                  >
                        0124
                        0124          BUG_CHECK       LOCKMGRERR,FATAL        ; Invalid lock state
                        0128
                        0128  40$:    ; We are stalling some or all lock requests.  See which ones.
                        0128
            05    19    0128          BLSS    50$                     ; All requests are being stalled
            09    E1    012A          BBC     #LKB$V_PROTECT,-        ; Only protected locks are being stalled
      DE 2A A6          012C                  LKB$W_STATUS(R6),30$    ; Branch if this is not a protected lock
      50    0E    D0    012F  50$:    MOVL    #RSN$_CLUSTRAN,R0       ; Resource to wait for
            BF    11    0132          BRB     CVT_WAIT                ; Go into MWAIT for RSN$_CLUSTRAN
                        0134
                        0134  60$:    ; Either no memory or CSID to CSB conversion failed with
                        0134          ; SS$_NOSUCHNODE.  Note that if we wait, system owned locks
                        0134          ; must have a non-zero PID.
                        0134
 0000'8F    50    B1    0134          CMPW    R0,#SS$_INSFMEM         ; Is it insuff. memory?
            0A    12    0139          BNEQ    70$                     ; No
      FE    8F    90    013B          MOVB    #LKB$K_RETRY,-          ; Store lock state
      36    A6          013E                  LKB$B_STATE(R6)
      50    03    D0    0140          MOVL    #RSN$_NPDYNMEM,R0       ; Resource to wait for
            AE    11    0143          BRB     CVT_WAIT                ; Go into MWAIT for RSN$_NPDYNMEM
                        0145
                        0145  70$:    BUG_CHECK       LOCKMGRERR,FATAL; CSID conversion error
                        0149
                        0149          .DSABL  LSB
                        0149
                        0149  ;***********************************************************************
                        0149  ;
                        0149  ;       Process level routines for handling remote conversion requests
                        0149  ;
                        0149  ;***********************************************************************
                        0149
                        0149  CVTRETRY_PROC:
                        0149          ; Retry this conversion due to the destination being removed
                        0149          ; from the cluster or we were unable to allocate a CDRP.
                        0149
   51    34 A6    9A    0149          MOVZBL  LKB$B_RQMODE(R6),R1     ; Get requested lock mode
   53    38 A8    D0    014D          MOVL    RSB$L_CSID(R8),R3       ; Get CSID of destination system
            0A    12    0151          BNEQ    10$                     ; Resend to destination
            01    90    0153          MOVB    #LKB$K_GRANTED,-        ; Store correct lock state
      36    A6          0155                  LKB$B_STATE(R6)
00000000'GF       17    0157          JMP     G^LCK$LOCAL_CVT         ; Do it locally
          FF31    31    015D  10$:    BRW     LCK$SND_CVTREQ
                        0160
                        0160  CGRNTD_PROC:
                        0160          ; Conversion request was granted
                        0160
            03    E1    0160          BBC     #LKB$V_BLKASTQED,-      ; Branch if a blocking AST
```

```
      0D 2A A6        0162                     LKB$W_STATUS(R6),20$      ; should not be queued
         02    A8     0165           BISW      #LKB$M_DBLKAST,-          ; Set deliver blocking AST status
         2A A6        0167                     LKB$W_STATUS(R6)
         10    88     0169           BISB      #LKB$M_PKAST,-            ; Set piggyback kernel AST bit
         0B A6        016B                     LKB$B_RMOD(R6)
         20 A6  D0    016D           MOVL      LKB$L_BLKASTADR(R6),-     ; Store address of blocking AST routine
         10 A6        0170                     LKB$L_AST(R6)
   50    38 A6  0F    0172   20$:    REMQUE    LKB$L_SQFL(R6),R0         ; Remove lock from RSB's granted queue
         34 A6  90    0176           MOVB      LKB$B_RQMODE(R6),-        ; Set granted mode
         35 A6        0179                     LKB$B_GRMODE(R6)
           00'  D0    017B           MOVL      S^#SS$_NORMAL,-           ; Set status for lock status block
         2C A6        017D                     LKB$L_LKST1(R6)
         20 A6  D5    017F           TSTL      LKB$L_BLKASTADR(R6)       ; Blocking AST specified?
         03    13     0182           BEQL      40$                       ; No
         42 A8  B6    0184           INCW      RSB$W_BLKASTCNT(R8)       ; Yes, adjust count
  00000000'GF   16    0187   40$:    JSB       G^LCK$GRANT_REM           ; Finish granting the lock
  00000000'GF   17    018D           JMP       G^LCK$CVT_GRANTED         ; Return value block and exit
                      0193
                      0193   CVTQED_PROC:
                      0193           ; Conversion was queued
                      0193
   50    38 A6  0F    0193           REMQUE    LKB$L_SQFL(R6),R0         ; Remove lock from granted queue
  00000000'GF   16    0197           JSB       G^LCK$QUEUECVT            ; Queue it to conversion queue
  00000000'GF   17    019D           JMP       G^LCK$QUEUED_EXIT
                      01A3
                      01A3   CNOTQD_PROC:
                      01A3           ; Conversion was not queued.  However, we have to handle the case
                      01A3           ; where we receive a blocking AST message before we get here.
                      01A3
         03    E1     01A3           BBC       #LKB$V_BLKASTQED,-        ; Branch if we didn't get a blocking AST
      03 2A A6        01A5                     LKB$W_STATUS(R6),60$
    57   02    A8     01A8           BISW      #LKB$M_DBLKAST,R7         ; Set deliver BLKAST bit in old status
         01    90     01AB   60$:    MOVB      #LKB$K_GRANTED,-          ; Store correct lock state
         36 A6        01AD                     LKB$B_STATE(R6)
  00000000'GF   17    01AF           JMP       G^LCK$CVTNOTQED
```

```
                        01B5           .SBTTL  SENDCVTREQ - Send convert request to remote system
                        01B5
                        01B5  ;++
                        01B5  ; FUNCTIONAL DESCRIPTION:
                        01B5  ;
                        01B5  ;        This routine takes care of building and sending the convert lock
                        01B5  ;        request.  It also handles the fork level processing of the response.
                        01B5  ;        We are the process system; the remote system is the master system.
                        01B5  ;        SENDCVTREQ is called to send a convert request that requires a
                        01B5  ;        response.  SENDCVTMSG is called to send a message that doesn't
                        01B5  ;        require a response.
                        01B5  ;
                        01B5  ; CALLING SEQUENCE:
                        01B5  ;
                        01B5  ;        BSBW    SENDCVTREQ
                        01B5  ;        BSBW    SENDCVTMSG
                        01B5  ;        IPL must be at SCS fork IPL (8)
                        01B5  ;        This routine operates as a fork process so it may return
                        01B5  ;        to its caller before completing.
                        01B5  ;
                        01B5  ; INPUT PARAMETERS:
                        01B5  ;
                        01B5  ;        R0      LKB address
                        01B5  ;        R3      CSB address of destination system
                        01B5  ;        R5      CDRP address
                        01B5  ;
                        01B5  ; OUTPUT PARAMETERS:
                        01B5  ;
                        01B5  ;        NONE
                        01B5  ;
                        01B5  ; SIDE EFFECTS:
                        01B5  ;
                        01B5  ;        R0 - R4 destroyed
                        01B5  ;
                        01B5  ;--
                        01B5           .ENABL  LSB
                        01B5
                        01B5  SENDCVTMSG:
        54 A0     D0    01B5           MOVL    LKB$L_REMLKID(R0),-     ; Store master lockid in CDRP
        2C A5           01B8                   CDRP$L_VAL1(R5)
        30 A0     D0    01BA           MOVL    LKB$L_LKID(R0),-        ; Store process lockid in CDRP
        30 A5           01BD                   CDRP$L_VAL2(R5)
        28 A0     B0    01BF           MOVW    LKB$W_FLAGS(R0),-       ; Store flags in CDRP
        36 A5           01C2                   CDRP$L_VAL3+2(R5)
  51    64 A3     D0    01C4           MOVL    CSB$L_CLUB(R3),R1       ; Get address of CLUB
      00AC C1     B0    01C8           MOVW    CLUB$W_MEMSEQ(R1),-     ; Store membership seq. number
        34 A5           01CC                   CDRP$L_VAL3(R5)
     0289'CF      9E    01CE           MOVAB   W^BLD_CVTMSG,-          ; Store address of message build routine
        4C A5           01D2                   CDRP$L_MSGBLD(R5)
        FE29'     30    01D4           BSBW    CNX$SEND_MSG_CSB        ; Send the message
     50    55     D0    01D7           MOVL    R5,R0                   ; Address of CDRP
 00000000'GF      17    01DA           JMP     G^EXE$DEANONPAGED       ; Deallocate it
                        01E0
                        01E0  SENDCVTREQ:
  2C A5    50     D0    01E0           MOVL    R0,CDRP$L_VAL1(R5)      ; Store LKB address
  51    64 A3     D0    01E4           MOVL    CSB$L_CLUB(R3),R1       ; Get address of CLUB
      00AC C1     B0    01E8           MOVW    CLUB$W_MEMSEQ(R1),-     ; Store membership seq. number
```

```
          34 A5       01EC                CDRP$L_VAL3(R5)
       02D0'CF    9E  01EE        MOVAB   W^BLD_CVTMSGR,-          ; Store address of message build
          4C A5       01F2                CDRP$L_MSGBLD(R5)        ; routine
          FD 8F   90  01F4        MOVB    #LKB$K_SCSWAIT,-         ; Store lock state
          36 A0       01F7                LKB$B_STATE(R0)
          FE04'   30  01F9        BSBW    CNX$SEND_MSG_CSB         ; Send the message
          39 50   E9  01FC        BLBC    R0,CVT_RETRY             ; Dest. system is no longer in cluster
                      01FF
                      01FF        ; We are resumed here when the response message arrives.
                      01FF        ; Registers contain:
                      01FF        ;       R2      Address of message buffer
                      01FF        ;       R3      CSB address
                      01FF        ;       R4      Address of PDT
                      01FF        ;       R5      Address of CDRP
                      01FF        ; First validate membership sequence number.  I.e. ignore response
                      01FF        ; and retry request if we did a membership state change while
                      01FF        ; this message was in transit.  Otherwise store response code
                      01FF        ; in LKB and then dispatch on it.
                      01FF
    50    64 A3   D0  01FF        MOVL    CSB$L_CLUB(R3),R0        ; Get address of CLUB
        00AC C0   B1  0203        CMPW    CLUB$W_MEMSEQ(R0),-      ; Compare sequence numbers
          0C A2       0207                LKMSG$W_MEMSEQ(R2)
             05   13  0209        BEQL    20$                      ; They match
          FE 8F   90  020B        MOVB    #LKB$K_RETRY,-           ; No match - change state to RETRY
          1E A2       020E                LKMSG$B_STATE(R2)
                      0210 20$:
    50    2C A5   D0  0210        MOVL    CDRP$L_VAL1(R5),R0       ; R0 = LKB address
          36 A0   91  0214        CMPB    LKB$B_STATE(R0),-        ; Make sure LKB is in SCSWAIT state
          FD 8F       0217                #LKB$K_SCSWAIT
             19   12  0219        BNEQ    30$                      ; It's not
          1E A2   90  021B        MOVB    LKMSG$B_STATE(R2),-      ; Store state
          36 A0       021E                LKB$B_STATE(R0)
                      0220
                      0220        DISPATCH        LKB$B_STATE(R0),TYPE=B,PREFIX=LKB$K_,-
                      0220                <-
                      0220                <RETRY,DEALL_WARMCDRP>,-         ; Retry request
                      0220                <RSPGRANTD,CGRNTD_FORK>,-        ; Granted
                      0220                <RSPQUEUED,CVTQED_FORK>,-        ; Queued
                      0220                <RSPNOTQED,CNOTQD_FORK>,-        ; Not queued
                      0220                >
                      0230
                      0230        BUG_CHECK       LOCKMGRERR,FATAL         ; Invalid lock state
                      0234
                      0234 30$:   BUG_CHECK       LOCKMGRERR,FATAL         ; LKB not in SCSWAIT state
                      0238
                      0238 CVT_RETRY:
                      0238        ; Remote system is being removed from cluster
                      0238
    54    2C A5   D0  0238        MOVL    CDRP$L_VAL1(R5),R4       ; Get LKB address
          FE 8F   90  023C        MOVB    #LKB$K_RETRY,-           ; Store retry state
          36 A4       023F                LKB$B_STATE(R4)
       50    55   D0  0241        MOVL    R5,R0                    ; Address of CDRP
   00000000'GF    16  0244        JSB     G^EXE$DEANONPAGED        ; Deallocate CDRP
           0496   31  024A        BRW     WAKE_PROCESS             ; Wake process
                      024D
                      024D CNOTQD_FORK:
                      024D CGRNTD_FORK:
```

```
                    024D  650          ; Conversion request was granted or not queued.  Store returned
                    024D  651          ; value block only if it's sequence number is later than the
                    024D  652          ; one in the RSB.  Then set BLKASTQED bit in LKB if necessary.
                    024D  653
   51    50 A0  D0  024D  654          MOVL    LKB$L_RSB(R0),R1            ; Get RSB address
         3C A1  C3  0251  655          SUBL3   RSB$L_VALSEQNUM(R1),-       ; Compare sequence numbers
   54    30 A2      0254  656                  LKMSG$L_VALSEQNUM(R2),R4
            1C  15  0257  657          BLEQ    40$                         ; We already have a newer one
         20 A2  7D  0259  658          MOVQ    LKMSG$Q_VALBLK(R2),-        ; Store value block
         28 A1      025C  659                  RSB$Q_VALBLK(R1)
         28 A2  7D  025E  660          MOVQ    LKMSG$Q_VALBLK+8(R2),-
         30 A1      0261  661                  RSB$Q_VALBLK+8(R1)
         30 A2  D0  0263  662          MOVL    LKMSG$L_VALSEQNUM(R2),- ; Store sequence number
         3C A1      0266  663                  RSB$L_VALSEQNUM(R1)
            02  AA  0268  664          BICW    #RSB$M_VALINVLD,-           ; Clear value block invalid flag
         0E A1      026A  665                  RSB$W_STATUS(R1)
            01  E1  026C  666          BBC     #RSB$V_VALINVLD,-           ; and conditionally set it
      04 1A A2      026E  667                  LKMSG$W_RSBSTATUS(R2),40$
            02  A8  0271  668          BISW    #RSB$M_VALINVLD,-
         0E A1      0273  669                  RSB$W_STATUS(R1)
                    0275  670
            03  E1  0275  671 40$:     BBC     #LKB$V_BLKASTQED,-          ; Branch if a blocking AST wasn't queued
      04 18 A2      0277  672                  LKMSG$W_LKBSTATUS(R2),50$
            08  A8  027A  673          BISW    #LKB$M_BLKASTQED,-          ; Set BLKASTQED in LKB
         2A A0      027C  674                  LKB$W_STATUS(R0)
          045C  31  027E  675 50$:     BRW     DEALL_WARMCDRP              ; Deallocate CDRP etc. and wake process
                    0281  676
                    0281  677 CVTQED_FORK:
                    0281  678          ; Conversion request was queued
                    0281  679
         30 A2  B0  0281  680          MOVW    LKMSG$W_RQSEQNM(R2),-       ; Store request sequence number
         10 A0      0284  681                  LKB$W_RQSEQNM(R0)
          0454  31  0286  682          BRW     DEALL_WARMCDRP              ; Deallocate CDRP etc. and wake process
                    0289  683
                    0289  684          .DSABL  LSB
```

```
                           0289   ;************************************************************************
                           0289   ;
                           0289   ;              Build convert message action routines
                           0289   ;
                           0289   ;************************************************************************
                           0289
                           0289           ; Action routine to build the actual convert lock request message
                           0289           ; Inputs are:
                           0289           ;       R2      Address of message buffer
                           0289           ;       R5      Address of CDRP
                           0289           ; NOTE: CDRP$L_VAL contains a PID for BLD_CVTMSG and a PCB
                           0289           ;       address for BLD_CVTMSGR.  This is because the PCB address
                           0289           ;       must be revalidated for the former case as the process
                           0289           ;       can get deleted before the message build routine is called.
                           0289           ; All registers except R0 and R1 must be preserved
                           0289
                           0289   BLD_CVTMSG:
                           0289           ; This entry point is for conversions that don't require a response
                           0289
                           0289           ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
                           0289           ASSUME  LKMSG$L_PRCLKID  EQ  LKMSG$L_MSTLKID+4
                           0289           ASSUME  LKMSG$L_DLCKPRI_CVT  EQ  LKMSG$L_EPIDCVT+4
                           0289
   08 A2    0602 8F   B0   0289           MOVW    #LKMSG$K_CVTLCKM@8-        ; Store function and facility codes
                           028F                   !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
             2C A5    7D   028F           MOVQ    CDRP$L_VAL1(R5),-          ; Store master and process lockids
             10 A2         0292                   LKMSG$L_MSTLKID(R2)        ; in message
             34 A5    B0   0294           MOVW    CDRP$L_VAL3(R5),-          ; Store memseq in message
             0C A2         0297                   LKMSG$W_MEMSEQ(R2)
             36 A5    B0   0299           MOVW    CDRP$L_VAL3+2(R5),-        ; Store flags in message
             18 A2         029C                   LKMSG$W_FLAGS(R2)
                           029E           ASSUME  LCK$V_VALBLK  EQ  0
          0A 18 A2    E9   029E           BLBC    LKMSG$W_FLAGS(R2),10$      ; Branch if no value block specified
             38 A5    7D   02A2           MOVQ    CDRP$L_VAL4(R5),-          ; Store value block in message
             20 A2         02A5                   LKMSG$Q_VALBLK(R2)
             40 A5    7D   02A7           MOVQ    CDRP$L_VAL6(R5),-
             28 A2         02AA                   LKMSG$Q_VALBLK+8(R2)
             5C A5    9B   02AC   10$:    MOVZBW  CDRP$L_VAL9(R5),-          ; Store requested mode, and clear
             1A A2         02AF                   LKMSG$B_RQMODE(R2)         ; granted mode
             5E A5    9A   02B1           MOVZBL  CDRP$L_VAL9+2(R5),-        ; Store blocking AST flag
             1C A2         02B4                   LKMSG$L_BLKASTFLG(R2)
       51    48 A5    3C   02B6           MOVZWL  CDRP$L_VAL8(R5),R1         ; Get PID index
  50   00000000'GF    D0   02BA           MOVL    G^SCH$GL_PCBVEC,R0         ; Get address of PCB vector
          50   6041   D0   02C1           MOVL    (R0)[R1],R0                ; Get address of PCB
             48 A5    D1   02C5           CMPL    CDRP$L_VAL8(R5),-          ; Validate PCB
             60 A0         02C8                   PCB$L_PID(R0)
                2E    13   02CA           BEQL    BLD_CVTCOM                 ; It's ok
             30 A2    7C   02CC           CLRQ    LKMSG$L_EPIDCVT(R2)        ; Process was deleted; clear
                      05   02CF           RSB                                ; EPID and deadlock priority fields
                           02D0
                           02D0   BLD_CVTMSGR:
                           02D0           ; This entry point is for conversions that require a response
                           02D0
                           02D0           ASSUME  LKMSG$L_MSTLKID  EQ  LKMSG$W_MEMSEQ+4
                           02D0           ASSUME  LKMSG$L_PRCLKID  EQ  LKMSG$L_MSTLKID+4
                           02D0           ASSUME  LKMSG$W_FLAGS  EQ  LKMSG$L_PRCLKID+4
                           02D0           ASSUME  LKMSG$B_RQMODE  EQ  LKMSG$W_FLAGS+2
```

```
                         02D0            ASSUME  LKMSG$L_BLKASTFLG  EQ  LKMSG$B_RQMODE+2
                         02D0            ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
                         02D0
  08 A2   0702 8F    B0  02D0            MOVW    #LKMSG$K_CVTLCKR@8-        ; Store function and facility codes
                         02D6                    !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
       50   2C A5    D0  02D6            MOVL    CDRP$L_VAL1(R5),R0         ; Get LKB address
       51   0C A2    3E  02DA            MOVAW   LKMSG$W_MEMSEQ(R2),R1      ; Point into message buffer
       81   34 A5    3C  02DE            MOVZWL  CDRP$L_VAL3(R5),(R1)+      ; Store seq. number
       81   54 A0    D0  02E2            MOVL    LKB$L_REMLKID(R0),(R1)+    ; Store master lockid
       81   30 A0    D0  02E6            MOVL    LKB$L_LKID(R0),(R1)+       ; Store process lockid
       81   28 A0    B0  02EA            MOVW    LKB$W_FLAGS(R0),(R1)+      ; Store flags
       81   34 A0    9B  02EE            MOVZBW  LKB$B_RQMODE(R0),(R1)+     ; Store req. mode and clear granted mode
       81   20 A0    D0  02F2            MOVL    LKB$L_BLKASTADR(R0),(R1)+  ; Store blocking AST flag
       50   48 A5    D0  02F6            MOVL    CDRP$L_VAL8(R5),R0         ; Get PCB address
                         02FA
                         02FA    BLD_CVTCOM:
            64 A0    D0  02FA            MOVL    PCB$L_EPID(R0),-           ; Store EPID
            30 A2        02FD                    LKMSG$L_EPIDCVT(R2)
          010C C0    D0  02FF            MOVL    PCB$L_DLCKPRI(R0),-        ; Store deadlock priority
            34 A2        0303                    LKMSG$L_DLCKPRI_CVT(R2)
                     05  0305            RSB
```

```
                     0306          .SBTTL  LKC$RCV_CVTREQ - Receive convert request
                     0306
                     0306   ;++
                     0306   ; FUNCTIONAL DESCRIPTION:
                     0306   ;
                     0306   ;       This routine receives conversion requests from the remote system.
                     0306   ;       The remote system is the process system and we are the master
                     0306   ;       system for this lock.  The conversion request is performed and
                     0306   ;       a response message is returned if called at entry point
                     0306   ;       LCK$RCV_CVTREQ.
                     0306   ;
                     0306   ; CALLING SEQUENCE:
                     0306   ;
                     0306   ;       JSB     LCK$RCV_CVTREQ (called from SCS received message routine)
                     0306   ;       IPL must be at IPL$_SYNCH
                     0306   ;
                     0306   ; INPUT PARAMETERS:
                     0306   ;
                     0306   ;       R2      Address of message buffer
                     0306   ;       R3      Address of CSB
                     0306   ;       R5      Address of CDRP if a response is needed
                     0306   ;
                     0306   ; OUTPUT PARAMETERS:
                     0306   ;
                     0306   ;       None
                     0306   ;
                     0306   ; SIDE EFFECTS:
                     0306   ;
                     0306   ;       A message may be sent back as a response
                     0306   ;       R0 - R5 destroyed
                     0306   ;--
                     0306
                     0306           ASSUME  LKMSG$L_PRCLKID  EQ  LKMSG$L_MSTLKID+4
                     0306           ASSUME  LKMSG$W_FLAGS  EQ  LKMSG$L_PRCLKID+4
                     0306           ASSUME  LKMSG$B_RQMODE  EQ  LKMSG$W_FLAGS+2
                     0306           ASSUME  LKMSG$L_BLKASTFLG  EQ  LKMSG$B_RQMODE+2
                     0306           ASSUME  LKMSG$Q_VALBLK  EQ  LKMSG$L_BLKASTFLG+4
                     0306
                     0306   CVT_INVLKID:
                     0306           BUG_CHECK       LOCKMGRERR,FATAL
                     030A
                     030A   LCK$RCV_CVTMSG::
           55   D4   030A           CLRL    R5                      ; Indicate no CDRP
                     030C
                     030C   LCK$RCV_CVTREQ::
                     030C
      00000002       030C           .IF NE  CA$_MEASURE
  00000000'GF   D6   030C           INCL    G^PMS$GL_ENQCVT_IN
                     0312           .ENDC
                     0312
    0FCC 8F     BB   0312           PUSHR   #^M<R2,R3,R6,R7,R8,R9,R10,R11>
           55   DD   0316           PUSHL   R5                      ; Save CDRP address or 0
           03   13   0318           BEQL    5$                      ; No CDRP
        FCE3'   30   031A           BSBW    CNX$INIT_CDRP           ; Initialize CDRP
  57   52  10   C1   031D   5$:     ADDL3   #LKMSG$L_MSTLKID,R2,R7  ; Point R7 into message buffer
       51  87   D0   0321           MOVL    (R7)+,R1                ; Get master lockid (id on this system)
       50  87   D0   0324           MOVL    (R7)+,R0                ; Get process lockid (for verify)
```

```
            0CAD   30  0327          BSBW    VERIFYREMLKID             ; Convert to LKB address
           D9 50   E9  032A          BLBC    R0,CVT_INVLKID            ; Invalid id
                       032D
                       032D          ; Have LKB address in R6, pointer into message in R7.  Set
                       032D          ; up other registers as needed.
                       032D
        58   50 A6 D0  032D          MOVL    LKB$L_RSB(R6),R8          ; Get RSB address
          59    87 3C  0331          MOVZWL  (R7)+,R9                  ; Get flags
       28 A6    59 B0  0334          MOVW    R9,LKB$W_FLAGS(R6)        ; Store them
           2A A6   DD  0338          PUSHL   LKB$W_STATUS(R6)          ; Save old status bits
                   AA  033B          BICW    #LKB$M_DCPLAST-           ; Clear relevant status bits
                       033C                  !LKB$M_DBLKAST-           ; The following bits retain their
                       033C                  !LKB$M_ASYNC-             ; old state:
                       033C                  !LKB$M_BLKASTQED-         ;       MSTCPY
                       033C                  !LKB$M_TIMOUTQ-           ;       NOQUOTA (should not be set)
                       033C                  !LKB$M_WASSYSOWN-         ;       PROTECT
                       033C                  !LKB$M_CVTTOSYS,-
   2A A6   01CF 8F     033C                  LKB$W_STATUS(R6)
          51    87 9A  0341          MOVZBL  (R7)+,R1                  ; Get requested mode, skip granted
                57 D6  0344          INCL    R7                        ; mode field
        5B   35 A6 9A  0346          MOVZBL  LKB$B_GRMODE(R6),R11      ; Get granted mode
                       034A
                       034A          ; Store EPID of owner of lock unless LCK$M_CVTSYS is set in
                       034A          ; which case, store 0.  R7 points to BLKASTFLG field of message.
                       034A          ; Also store deadlock priority in LKB if not system owned.
                       034A
                50 D4  034A          CLRL    R0                        ; Assume system owned
       09 59    06 E0  034C          BBS     #LCK$V_CVTSYS,R9,8$       ; Branch if it is system owned
        50   14 A7 D0  0350          MOVL    LKMSG$L_EPIDCVT-LKMSG$L_BLKASTFLG(R7),R0  ; Get EPID
             18 A7 D0  0354          MOVL    LKMSG$L_DLCKPRI_CVT-LKMSG$L_BLKASTFLG(R7),-
             24 A6     0357                  LKB$L_DLCKPRI(R6)         ; Store deadlock priority
       14 A6    50 D0  0359  8$:     MOVL    R0,LKB$L_EPID(R6)         ; Store EPID or 0
                       035D
                       035D          ; Decrement blocking AST count if a blocking AST was specified
                       035D          ; and then store new blocking AST flag.
                       035D          ; NOTE: the store of the new flag MUST be done after the test
                       035D          ; of the old flag and decrement of the count.
                       035D
             20 A6 D0  035D          MOVL    LKB$L_BLKASTADR(R6),-     ; Save old blocking AST flag
             5C A6     0360                  LKB$L_OLDBLKAST(R6)
                03 13  0362          BEQL    10$                       ; No
             42 A8 B7  0364          DECW    RSB$W_BLKASTCNT(R8)       ; Yes, adjust count
       20 A6    87 D0  0367  10$:    MOVL    (R7)+,LKB$L_BLKASTADR(R6) ; Store new flag
                       036B
                       036B          ; If we are converting down (or same) from PW or EX then copy
                       036B          ; the caller's value block (if specified) into the RSB.
                       036B          ; R7 points to value block in message.
                       036B
                       036B          ASSUME  LCK$V_VALBLK  EQ  0
                       036B
          19    59 E9  036B          BLBC    R9,20$                    ; Branch if no value block specified
       04       5B 91  036E          CMPB    R11,#LCK$K_PWMODE         ; Is grmode mode PW or EX?
                14 1F  0371          BLSSU   20$                       ; No
        5B      51 91  0373          CMPB    R1,R11                    ; Is rqmode greater than grmode?
                0F 1A  0376          BGTRU   20$                       ; Yes
       28 A8    87 7D  0378          MOVQ    (R7)+,RSB$Q_VALBLK(R8)    ; Copy caller's value block to RSB
       30 A8    67 7D  037C          MOVQ    (R7),RSB$Q_VALBLK+8(R8)
```

```
            3C A8   D6 0380          INCL    RSB$L_VALSEQNUM(R8)       ; Increment value block seq. number
               02   AA 0383          BICW    #RSB$M_VALINVLD,-         ; Clear invalid bit
            0E A8      0385                  RSB$W_STATUS(R8)
                       0387
                       0387     20$: ; Remove this lock from the granted queue.  If it was the only one and
                       0387          ; if the conversion queue is also empty, then the conversion request
                       0387          ; can be granted immediately.  This path is special cased because it
                       0387          ; is the normal case.
                       0387
         50    38 A6   0F 0387       REMQUE  LKB$L_SQFL(R6),R0         ; Remove lock from granted queue
               1F   12 038B          BNEQ    40$                       ; Not the only one
         5A    18 A8   DE 038D       MOVAL   RSB$L_CVTQFL(R8),R10      ; It's the only granted lock
            5A    6A   D1 0391       CMPL    (R10),R10                 ; Is conversion queue empty?
               16   12 0394          BNEQ    40$                       ; It's not - must check the longer way
      00000000'GF   16 0396          JSB     G^LCK$GRANT_LOCK_ALT      ; It is - grant lock
            5A    08   C0 039C       ADDL    #8,R10                    ; Point to wait queue
            5A    6A   D1 039F       CMPL    (R10),R10                 ; Is wait queue empty?
               46   13 03A2          BEQL    60$                       ; Yes, exit
      00000000'GF   16 03A4          JSB     G^LCK$GRANTWTRS           ; Try granting waiting locks
               3E   11 03AA          BRB     60$
                       03AC
                       03AC     40$: ; There was at least one other holder of the resource so we have
                       03AC          ; to check for compatibility the longer way.  The granted mode
                       03AC          ; of this lock is compared with the conversion grant mode.  If,
                       03AC          ; other than the head of the conversion queue, there are granted
                       03AC          ; locks with higher lock modes than this lock, then there
                       03AC          ; is no need to recompute the group grant mode or attempt to
                       03AC          ; grant waiting conversions.
                       03AC
         0D A8    5B   91 03AC       CMPB    R11,RSB$B_CGMODE(R8)      ; Is granted mode = conv. grant mode?
               15   13 03B0          BEQL    45$                       ; Yes
         55    0C A8   9A 03B2       MOVZBL  RSB$B_GGMODE(R8),R5       ; No, get group grant mode
 54 00000000'GF45 51   E1 03B6       BBC     R1,G^[CK$COMPAT_TBL[R5],80$ ; Branch if not compatible
      00000000'GF   16 03BF          JSB     G^LCK$GRANT_LOCK          ; Grant the lock
               23   11 03C5          BRB     60$                       ; Exit with completion status in R0
                       03C7
      00000000'GF   16 03C7     45$: JSB     G^LCK$COMP_GGMODE         ; Compute new group grant mode in R5
 3D 00000000'GF45 51   E1 03CD       BBC     R1,G^LCK$COMPAT_TBL[R5],80$ ; Branch if not compatible
         0C A8    55   90 03D6       MOVB    R5,RSB$B_GGMODE(R8)       ; Store group grant mode in RSB
         0D A8    55   90 03DA       MOVB    R5,RSB$B_CGMODE(R8)       ; Also store conversion grant mode
      00000000'GF   16 03DE          JSB     G^LCK$GRANT_LOCK          ; Grant lock
      00000000'GF   16 03E4          JSB     G^LCK$GRANTCVTS           ; Try granting conversions and waiters
                       03EA
                       03EA     60$: ; Set response state to be granted
                       03EA
            5E    04   C0 03EA       ADDL    #4,SP                     ; Remove old status from stack
               02   AA 03ED          BICW    #LKB$M_DBLKAST,-          ; Got set spuriously by common code
            2A A6      03EF                  LKB$W_STATUS(R6)
         50    FA 8F   90 03F1       MOVB    #LKB$K_RSPGRANTD,R0       ; Set state
            55  8ED0      03F5       POPL    R5                        ; Get response CDRP or 0
            58    12      03F8       BNEQ    SEND_CVTRSP               ; Have a CDRP - send response
                       03FA
                       03FA     70$: ; Since a response is not required, we have to deallocate the
                       03FA          ; message buffer.  However, it may be necessary to send a blocking
                       03FA          ; message.
                       03FA
               0C   BA 03FA          POPR    #^M<R2,R3>                ; Get buffer and CSB addresses
```

```
        FC01'   30  03FC            BSBW    CNX$DEALL_MSG_BUF_CSB     ; Deallocate message buffer
           03   E1  03FF            BBC     #LKB$V_BLKASTQED,-        ; Branch if no blocking AST queued
     0A 2A A6       0401                    LKB$W_STATUS(R6),75$
   53  58 A6    D0  0404            MOVL    LKB$L_CSID(R6),R3         ; Get CSID of other system
      54   56   D0  0408            MOVL    R6,R4                     ; Move LKB address
         090F   30  040B            BSBW    SEND_BLOCKING             ; Send blocking message
     0FC0 8F    BA  040E    75$:    POPR    #^M<R6,R7,R8,R9,R10,R11>
                05  0412            RSB
                    0413
                    0413    80$:    ; The conversion cannot be granted.  Queue the request
                    0413            ; unless the noqueue bit is set.  If the conversion queue is empty
                    0413            ; then R5 contains the new conversion grant mode.
                    0413
        57 8ED0     0413            POPL    R7                        ; Restore old LKB status bits
   18 59   02   E0  0416            BBS     #LCK$V_NOQUEUE,R9,85$     ; Branch if noqueue is set
                    041A
                    041A            ; Queue the conversion
                    041A
   34 A6   51   90  041A            MOVB    R1,LKB$B_RQMODE(R6)       ; Store requested mode
        46 A8   B0  041E            MOVW    RSB$W_RQSEQNM(R8),-       ; Store request sequence number
        10 A6       0421                    LKB$W_RQSEQNM(R6)
        46 A8   B6  0423            INCW    RSB$W_RQSEQNM(R8)         ; Increment sequence number
  00000000'GF   16  0426            JSB     G^LCK$QUEUECVT            ; Insert onto conversion queue, etc.
   50   FB 8F   90  042C            MOVB    #LKB$K_RSPQUEUED,R0       ; Set response state
           1D   11  0430            BRB     98$                       ; Send response
                    0432
                    0432    85$:    ; The request is not to be queued.  Insert back onto the
                    0432            ; granted queue.
                    0432
         00000002   0432            .IF NE  CA$_MEASURE
  00000000'GF   D6  0432            INCL    G^PMS$GL_ENQNOTQD
                    0438            .ENDC
                    0438
   2A A6   57   B0  0438            MOVW    R7,LKB$W_STATUS(R6)       ; Put back old status bits
     38 A6      0E  043C            INSQUE  LKB$L_SQFL(R6),-          ; Put lock back on granted queue
     10 A8          043F                    RSB$L_GRQFL(R8)
     5C A6      D0  0441            MOVL    LKB$L_OLDBLKAST(R6),-     ; Restore old blocking AST flag
     20 A6          0444                    LKB$L_BLKASTADR(R6)
        03      13  0446            BEQL    95$                       ; No
     42 A8      B6  0448            INCW    RSB$W_BLKASTCNT(R8)       ; Incr. blocking AST count
   50   FC 8F   90  044B    95$:    MOVB    #LKB$K_RSPNOTQED,R0       ; Set response state
        55 8ED0     044F    98$:    POPL    R5                        ; Restore CDRP address
                    0452
                    0452    SEND_CVTRSP:
                    0452            ; Send response message.  CDRP address is in R5.
                    0452
   2C A5   56   D0  0452            MOVL    R6,CDRP$L_VAL1(R5)        ; Store LKB address
   32 A5   50   90  0456            MOVB    R0,CDRP$L_VAL2+2(R5)      ; Store response state
       88'AF    9E  045A            MOVAB   B^BLD_CVTRSP,-            ; Store address of message build
        4C A5       045D                    CDRP$L_MSGBLD(R5)         ; routine
     0FCC 8F    BA  045F            POPR    #^M<R2,R3,R6,R7,R8,R9,R10,R11>
   51  64 A3    D0  0463            MOVL    CSB$L_CLUB(R3),R1         ; Get address of CLUB
     00AC C1    B0  0467            MOVW    CLUB$W_MEMSEQ(R1),-       ; Store membership seq. number
        34 A5       046B                    CDRP$L_VAL3(R5)
   53  4C A3    D0  046D            MOVL    CSB$L_CSID(R3),R3         ; Get CSID out of CSB
        FB8C'   30  0471            BSBW    CNX$SEND_MSG_RESP         ; Send message response
  0000'8F  50   B1  0474            CMPW    R0,#SS$_NOSUCHNODE        ; Is status SS$_NOSUCHNODE?
```

```
             09  13 0479           BEQL    10$                         ; Yes, error
          50 55  D0 047B           MOVL    R5,R0                       ; No, SS$_NODELEAVE is okay
     00000000'GF 17 047E           JMP     G^EXE$DEANONPAGED           ; Deallocate CDRP
                    0484
                    0484      10$: BUG_CHECK       LOCKMGRERR,FATAL; CSID invalid
                    0488
                    0488      BLD_CVTRSP:
                    0488           ; This routine builds a response message for conversion requests
                    0488           ; Inputs:
                    0488           ;       R2      Address of message buffer
                    0488           ;       R3      Address of CSB
                    0488           ;       R5      Address of CDRP
                    0488           ; All registers except R0 and R1 must be preserved
                    0488           ;
                    0488           ; Note:  This message build routine uses pointers (to LKB and RSB)
                    0488           ; that will not be valid if we started a state change.  Therefore,
                    0488           ; it is necessary to validiate the membership sequence number
                    0488           ; before using any pointers stored in the CDRP.
                    0488
                    0488           ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
                    0488
  08 A2    0782 8F  B0 0488        MOVW    #LKMSG$K_CVTLCKR@8-         ; Store function and facility codes
                    048E                   !CLSMSG$M_RESPMSG-
                    048E                   !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
          34 A5  B0 048E           MOVW    CDRP$L_VAL3(R5),-           ; Store membership seq. num.
          0C A2     0491                   LKMSG$W_MEMSEQ(R2)
       51 64 A3  D0 0493           MOVL    CSB$L_CLUB(R3),R1           ; Get address of CLUB
        00AC C1  B1 0497           CMPW    CLUB$W_MEMSEQ(R1),-         ; Have we done a state change?
          34 A5     049B                   CDRP$L_VAL3(R5)
             3D  12 049D           BNEQ    20$                         ; Yes
       50 2C A5  D0 049F           MOVL    CDRP$L_VAL1(R5),R0          ; Get address of LKB
          30 A0  D0 04A3           MOVL    LKB$L_LKID(R0),-            ; Store master lockid
          10 A2     04A6                   LKMSG$L_MSTLKID(R2)
          54 A0  D0 04A8           MOVL    LKB$L_REMLKID(R0),-         ; Store process lockid
          14 A2     04AB                   LKMSG$L_PRCLKID(R2)
          32 A5  90 04AD           MOVB    CDRP$L_VAL2+2(R5),-         ; Store response state
          1E A2     04B0                   LKMSG$B_STATE(R2)
          36 A0  91 04B2           CMPB    LKB$B_STATE(R0),-           ; Is lock state = GRANTED
             01     04B5                   #LKB$K_GRANTED
             1E  12 04B6           BNEQ    10$                         ; No
          2A A0  B0 04B8           MOVW    LKB$W_STATUS(R0),-          ; Yes, store LKB status
          18 A2     04BB                   LKMSG$W_LKBSTATUS(R2)
       50 50 A0  D0 04BD           MOVL    LKB$L_RSB(R0),R0            ; Get RSB address
          28 A0  7D 04C1           MOVQ    RSB$Q_VALBLK(R0),-          ; Store value block
          20 A2     04C4                   LKMSG$Q_VALBLK(R2)
          30 A0  7D 04C6           MOVQ    RSB$Q_VALBLK+8(R0),-
          28 A2     04C9                   LKMSG$Q_VALBLK+8(R2)
          3C A0  D0 04CB           MOVL    RSB$L_VALSEQNUM(R0),-       ; Store value block seq. number
          30 A2     04CE                   LKMSG$L_VALSEQNUM(R2)
          0E A0  B0 04D0           MOVW    RSB$W_STATUS(R0),-          ; Store RSB status
          1A A2     04D3                   LKMSG$W_RSBSTATUS(R2)
                 05 04D5           RSB
                    04D6
          10 A0  B0 04D6      10$: MOVW    LKB$W_RQSEQNM(R0),-         ; Store request sequence number
          30 A2     04D9                   LKMSG$W_RQSEQNM(R2)         ; (not used if state = RSPCNOTQD)
                 05 04DB           RSB
                    04DC
```

```
                04DC   1050 20$:    ; We've done a membership state change between our SEND_MSG
                04DC   1051         ; and calling the message build routine.  Just send back a RETRY
                04DC   1052         ; status.
                04DC   1053
  FE 8F    90   04DC   1054         MOVB    #LKB$K_RETRY,-          ; Store RETRY status
  1E A2         04DF   1055                 LKMSG$B_STATE(R2)
           05   04E1   1056         RSB
```

```
                        04E2          .SBTTL  LCK$SND_LOCKREQ - Send lock request to remote system
                        04E2
                        04E2   ;++
                        04E2   ; FUNCTIONAL DESCRIPTION:
                        04E2   ;
                        04E2   ;       This routine handles lock requests that must be forwarded
                        04E2   ;       to the master system for this lock.  This system is the process
                        04E2   ;       system.  In the case of root locks, the master system has not
                        04E2   ;       been determined yet so this request effectively functions as
                        04E2   ;       a directory lookup.
                        04E2   ;
                        04E2   ; CALLING SEQUENCE:
                        04E2   ;
                        04E2   ;       JMP     LCK$SND_LOCKREQ (This routine does not return to its
                        04E2   ;                               caller.  Instead it jumps to an
                        04E2   ;                               appropriate destination.)
                        04E2   ;       IPL must be at IPL$_SYNCH
                        04E2   ;
                        04E2   ; INPUT PARAMETERS:
                        04E2   ;
                        04E2   ;       R3      CSID of destination system
                        04E2   ;       R6      Address of LKB
                        04E2   ;       R8      Address of RSB
                        04E2   ;       R9      Flags
                        04E2   ;
                        04E2   ; OUTPUT PARAMETERS:
                        04E2   ;
                        04E2   ;       R0      Completion code (on some exit paths)
                        04E2   ;
                        04E2   ; SIDE EFFECTS:
                        04E2   ;
                        04E2   ;       The process will go into MWAIT state until the response from the
                        04E2   ;       remote system arrives.
                        04E2   ;
                        04E2   ;       R0 - R5 destroyed
                        04E2   ;--
                        04E2
                        04E2          .ENABL  LSB
                        04E2
                        04E2   LCK$SND_LOCKREQ::
                        04E2
                        04E2          ; The LKB is inserted on the RSB's wait queue until the response
                        04E2          ; arrives.  Note:  Normally, the LKB state field should get stored
                        04E2          ; adjacent to the INSQUE.  Here, it gets stored shortly; either
                        04E2          ; in SENDLOCKREQ or in the code that waits for pool.
                        04E2
        38 A6      0E   04E2          INSQUE  LKB$L_SQFL(R6),-          ; Insert LKB onto tail of RSB's
        24 B8           04E5                  @RSB$L_WTQBL(R8)          ; wait queue.
                        04E7
                        04E7   RESEND: ; Allocate a CDRP and send the message
                        04E7
54  00000000'GF    D0   04E7          MOVL    G^SCH$GL_CURPCB,R4        ; Get our PCB address
          FB0F'    30   04EE          BSBW    CNX$ALLOC_WARMCDRP        ; Alloc. a CDRP with RSPID and CVT CSID
        5D 50      E9   04F1          BLBC    R0,WAITFORMEM             ; No CDRP or CSID error
     50    56      D0   04F4          MOVL    R6,R0                     ; Move LKB address
     51    58      D0   04F7          MOVL    R8,R1                     ; Move RSB address
  48 A5    54      D0   04FA          MOVL    R4,CDRP$L_VAL8(R5)        ; Store PCB address in CDRP
```

```
           00C4  30 04FE         BSBW    SENDLOCKREQ             ; Send the lock request
   54    48 A5   D0 0501         MOVL    CDRP$L_VAL8(R5),R4      ; Restore PCB address
                    0505
                    0505 LCK_SCS_WAIT:
      50    0D   D0 0505         MOVL    #RSN$_SCS,R0            ; Go into MWAIT for resource RSN$_SCS
                    0508
                    0508 LCK_WAIT:
                    0508         ; Put the process into MWAIT until the response arrives or
                    0508         ; the resource becomes available.  Resource number is in R0.
                    0508         ; PCB address is in R4.
                    0508
         60 A4   D0 0508         MOVL    PCB$L_PID(R4),-         ; Make sure PID is stored in case lock
         0C A6      050B                 LKB$L_PID(R6)           ; is system owned
            10   78 050D         ASHL    #PSL$V_IPL,-            ; Create a PSL on stack with IPL
         7E 02      050F                 #IPL$_ASTDEL,-(SP)      ; set to IPL$_ASTDEL
    00000000'GF  16 0511         JSB     G^SCH$RWAIT             ; Wait
                    0517
                    0517         ; Upon reawakening the LKB$B_STATE field tells us what action
                    0517         ; to perform.  But first, we have to make sure that we're not
                    0517         ; stalling lock requests.
                    0517
                    0517         SETIPL  #IPL$_SYNCH             ; Raise IPL
    00000000'GF  95 051A         TSTB    G^LCK$GB_STALLREQS      ; Are we stalling?
            1D   12 0520         BNEQ    30$                     ; Yes
      03 59 06   E1 0522 15$:    BBC     #LCK$V_CVTSYS,R9,20$    ; Branch if not system owned
         0C A6   D4 0526         CLRL    LKB$L_PID(R6)           ; Clear PID of system owned lock
                    0529
                    0529 20$:    DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
                    0529                 <-
                    0529                 <RSPDOLOCL,DOLOCL_PROC>,-       ; Do locally
                    0529                 <RSPGRANTD,GRANTD_PROC>,-       ; Granted
                    0529                 <RSPQUEUED,WAITNG_PROC>,-       ; Waiting
                    0529                 <RSPNOTQED,NOTQED_PROC>,-       ; Queued
                    0529                 <SCSWAIT,LCK_SCS_WAIT>,-        ; Spurious wakeup
                    0529                 <RETRY,LCKRETRY_PROC>,-         ; Retry lock
                    0529                 >
                    053B
                    053B         BUG_CHECK       LOCKMGRERR,FATAL        ; Invalid lock state
                    053F
                    053F 30$:    ; We are stalling some or all lock requests.  See which ones.
                    053F
            05   19 053F         BLSS    40$                     ; All requests are being stalled
            09   E1 0541         BBC     #LKB$V_PROTECT,-        ; Only protected locks are being stalled
      DC 2A A6      0543                 LKB$W_STATUS(R6),15$    ; Branch if this is not a protected lock
         50 0E   D0 0546 40$:    MOVL    #RSN$_CLUSTRAN,R0       ; Resource to wait for
            BD   11 0549         BRB     LCK_WAIT                ; Go into MWAIT for RSN$_CLUSTRAN
                    054B
                    054B RESEND_BRANCH:
            9A   11 054B         BRB     RESEND                  ; *** TEMPORARY ***
                    054D
                    054D CSID_ERROR:                             ; CSID to CSB conversion error
                    054D         BUG_CHECK       LOCKMGRERR,FATAL
                    0551
                    0551 WAITFORMEM:
                    0551         ; CSID conversion error or insufficient pool to allocate
                    0551         ; CDRP.  If not CSID error then wait for pool and try again.
                    0551
```

```
   0000'8F     50   B1  0551          CMPW    R0,#SS$_INSFMEM         ; Is it insuff. memory?
               F5   12  0556          BNEQ    CSID_ERROR              ; No
            FE 8F   90  0558          MOVB    #LKB$K_RETRY,-          ; Store lock state
            36 A6       055B                  LKB$B_STATE(R6)
        50     03   D0  055D          MOVL    #RSN$_NPDYNMEM,R0       ; Resource to wait for
               A6   11  0560          BRB     LCK_WAIT                ; Go into MWAIT for RSN$_NPDYNMEM
                        0562
                        0562          .DSABL  LSB
                        0562
                        0562  ;*******************************************************************************
                        0562  ;
                        0562  ;       Process level routines for handling remote lock requests
                        0562  ;
                        0562  ;*******************************************************************************
                        0562
                        0562  LCKRETRY_PROC:
        53     38 A8    D0  0562      MOVL    RSB$L_CSID(R8),R3       ; Get CSID of destination system
               E3   12  0566          BNEQ    RESEND_BRANCH           ; Try again
                        0568
                        0568          ; This system became the manager of the resource while we
                        0568          ; were waiting.  Fall through to ...
                        0568
                        0568  DOLOCL_PROC:
                        0568          ; Handle the request locally.
                        0568
        50     38 A6    0F  0568      REMQUE  LKB$L_SQFL(R6),R0       ; Remove from wait queue
   00000000'GF      17  056C          JMP     G^LCK$LOCAL_LOCK
                        0572
                        0572  GRANTD_PROC:
                        0572          ; Lock request was granted
                        0572
               03   E1  0572          BBC     #LKB$V_BLKASTQED,-      ; Branch if blocking AST should
         0D 2A A6       0574                  LKB$W_STATUS(R6),10$    ; not be queued
               02   A8  0577          BISW    #LKB$M_DBLKAST,-        ; Set deliver blocking AST status
            2A A6       0579                  LKB$W_STATUS(R6)
               10   88  057B          BISB    #LKB$M_PKAST,-          ; Set piggyback kernel AST bit
            0B A6       057D                  LKB$B_RMOD(R6)
            20 A6   D0  057F          MOVL    LKB$L_BLKASTADR(R6),-   ; Store address of blocking AST routine
            10 A6       0582                  LKB$L_AST(R6)
        50     38 A6    0F  0584  10$: REMQUE  LKB$L_SQFL(R6),R0      ; Remove lock from RSB's wait queue
            34 A6   90  0588          MOVB    LKB$B_RQMODE(R6),-      ; Set granted mode
            35 A6       058B                  LKB$B_GRMODE(R6)
              00'   D0  058D          MOVL    S^#SS$_NORMAL,-         ; Set status for lock status block
            2C A6       058F                  LKB$L_LKST1(R6)
            20 A6   D5  0591          TSTL    LKB$L_BLKASTADR(R6)     ; Blocking AST specified?
               03   13  0594          BEQL    15$                     ; No
            42 A8   B6  0596          INCW    RSB$W_BLKASTCNT(R8)     ; Yes, adjust count
   00000000'GF      16  0599  15$:    JSB     G^LCK$GRANT_REM         ; Finish granting the lock
   00000000'GF      17  059F          JMP     G^LCK$SYNC_EXIT         ; Exit the service
                        05A5
                        05A5  WAITNG_PROC:
                        05A5          ; Lock request had to wait.  PCB address is still in R4.
                        05A5
   00000000'GF      16  05A5          JSB     G^LCK$QUEUE_REM         ; Queue the lock request
   00000000'GF      17  05AB          JMP     G^LCK$QUEUED_EXIT       ; Exit the service
                        05B1
                        05B1  NOTQED_PROC:
```

```
                          05B1          ; Lock request was not queued due to an error.  Status code
                          05B1          ; returned by remote system is in LKB.  Typical values are
                          05B1          ;  SS$_NOTQUEUED or SS$_NOLOCKIDS.
                          05B1
     5B    2C A6    3C    05B1          MOVZWL  LKB$L_LKST1(R6),R11      ; Get status
                          05B5
                          05B5  CHECK_RSB:
                          05B5          ; LKB is to be deallocated.  Check if RSB should be deallocated too.
                          05B5          ; Status is in R11.
                          05B5
     50    38 A6    0F    05B5          REMQUE  LKB$L_SQFL(R6),R0        ; Remove lock from RSB's wait queue
   00000000'GF      16    05B9          JSB     G^LCK$CHECK_RSB          ; Deallocate RSB if necessary
   00000000'GF      17    05BF          JMP     G^LCK$NOT_QUEUED         ; Cleanup and exit service
                          05C5
```

```
                        05C5            .SBTTL  SENDLOCKREQ - Send a (new) lock request
                        05C5
                        05C5    ;++
                        05C5    ; FUNCTIONAL DESCRIPTION:
                        05C5    ;
                        05C5    ;       This routine takes care of building and sending the new lock
                        05C5    ;       request message to the remote system.  It also handles
                        05C5    ;       fork level processing of the response.  We are the process system;
                        05C5    ;       the remote system is the master system.
                        05C5    ;
                        05C5    ; CALLING SEQUENCE:
                        05C5    ;
                        05C5    ;       BSBW    SENDLOCKREQ
                        05C5    ;       IPL must be at SCS fork IPL (8)
                        05C5    ;       This routine operates as a fork process so it may return
                        05C5    ;       to its caller before completing.
                        05C5    ;
                        05C5    ; INPUT PARAMETERS:
                        05C5    ;
                        05C5    ;       R0      LKB address
                        05C5    ;       R1      RSB Address
                        05C5    ;       R3      CSB address
                        05C5    ;       R5      CDRP address
                        05C5    ;
                        05C5    ; IMPLICIT INPUTS:
                        05C5    ;
                        05C5    ;       CDRP$L_VAL8     Contains PCB address of process requesting lock
                        05C5    ;
                        05C5    ; OUTPUT PARAMETERS:
                        05C5    ;
                        05C5    ;       R0 - R4 destroyed
                        05C5    ;
                        05C5    ; SIDE EFFECTS:
                        05C5    ;
                        05C5    ;       NONE
                        05C5    ;
                        05C5    ;--
                        05C5            .ENABL  LSB
                        05C5
                        05C5    SENDLOCKREQ:
                        05C5            ; Put everything we need to build the message into the CDRP.
                        05C5
   2C A5    50    7D    05C5            MOVQ    R0,CDRP$L_VAL1(R5)      ; Store LKB and RSB addresses
51    64 A3       D0    05C9            MOVL    CSB$L_CLUB(R3),R1       ; Get address of CLUB
   00AC C1        B0    05CD            MOVW    CLUB$Q_MEMSEQ(R1),-     ; Store membership seq. number
      34 A5             05D1                    CDRP$L_VAL3(R5)
   06FC'CF        9E    05D3            MOVAB   W^BLD_LOCKMSG,-         ; Store address of routine that will
      4C A5             05D7                    CDRP$L_MSGBLD(R5)       ; actually build message
      FD 8F       90    05D9            MOVB    #LKB$K_SCSWAIT,-        ; Set LKB state to SCSWAIT
      36 A0             05DC                    LKB$B_STATE(R0)
                        05DE
                        05DE            ; Send the message
                        05DE
      FA1F'       30    05DE            BSBW    CNX$SEND_MSG_CSB
      41 50       E9    05E1            BLBC    R0,LCK_SENDERR          ; Destination system is being
                        05E4                                            ; removed from cluster
                        05E4
```

```
                    05E4          ; We are resumed here when the response message arrives.
                    05E4          ; Registers contain:
                    05E4          ;         R2      Address of message buffer
                    05E4          ;         R3      Address of CSB
                    05E4          ;         R4      Address of PDT
                    05E4          ;         R5      Address of CDRP
                    05E4          ; First validate membership sequence number.  I.e. ignore response
                    05E4          ; and retry request if we did a membership state change while
                    05E4          ; this message was in transit.  Otherwise store response code
                    05E4          ; in LKB and then dispatch on it.
                    05E4
50   64 A3    D0    05E4          MOVL    CSB$L_CLUB(R3),R0        ; Get address of CLUB
  00AC C0     B1    05E8          CMPW    CLUB$W_MEMSEQ(R0),-      ; Compare sequence numbers
     0C A2          05EC                  LKMSG$W_MEMSEQ(R2)
        05    13    05EE          BEQL    20$                      ; They match
     FE 8F    90    05F0          MOVB    #LKB$K_RETRY,-           ; No match - change state to RETRY
     1E A2          05F3                  LKMSG$B_STATE(R2)
50   2C A5    7D    05F5   20$:   MOVQ    CDRP$L_VAL1(R5),R0       ; R0 = LKB address;  R1 = RSB address
     36 A0    91    05F9          CMPB    LKB$B_STATE(R0),-        ; Make sure LKB is in SCSWAIT state
     FD 8F          05FC                  #LKB$K_SCSWAIT
        1D    12    05FE          BNEQ    30$                      ; It's not
     1E A2    90    0600          MOVB    LKMSG$B_STATE(R2),-      ; Store state
     36 A0          0603                  LKB$B_STATE(R0)
                    0605
                    0605          DISPATCH        LKB$B_STATE(R0),TYPE=B,PREFIX=LKB$K_,-
                    0605                  <-
                    0605                  <RETRY,DEALL_WARMCDRP>,-            ; Retry this request
                    0605                  <RSPRESEND,RESEND_FORK>,-           ; Resend to specified node
                    0605                  <RSPDOLOCL,DOLOCL_FORK>,-           ; Do locally
                    0605                  <RSPGRANTD,GRANTD_FORK>,-           ; Granted
                    0605                  <RSPQUEUED,WAITNG_FORK>,-           ; Waiting
                    0605                  <RSPNOTQED,NOTQED_FORK>,-           ; Not queued
                    0605                  >
                    0619
                    0619          BUG_CHECK       LOCKMGRERR,FATAL            ; Invalid lock state
                    061D
                    061D   30$:   BUG_CHECK       LOCKMGRERR,FATAL            ; LKB not in SCSWAIT state
                    0621   35$:   BUG_CHECK       LOCKMGRERR,FATAL            ; Other system returned
                    0625                                                      ; invalid CSID
                    0625   LCK_SENDERR:
                    0625          ; Remote system is being removed from cluster
                    0625
54   2C A5    D0    0625          MOVL    CDRP$L_VAL1(R5),R4       ; Get LKB address
  50    55    D0    0629          MOVL    R5,R0                    ; Address of CDRP
00000000'GF   16    062C          JSB     G^EXE$DEANONPAGED        ; Deallocate CDRP
        4C    11    0632          BRB     LCK_RETRY                ; Wake process and retry request
                    0634
                    0634   ;****************************************************************************
                    0634   ;
                    0634   ;      Fork level routines for handling remote lock request responses
                    0634   ;
                    0634   ;****************************************************************************
                    0634
                    0634   RESEND_FORK:
                    0634          ; Resend request to another system (specified by CSID).
                    0634          ; New destination CSID is in message.  But before we use it we
                    0634          ; have to make sure the CSID address in the RSB didn't change out from
```

```
                          0634          ; underneath us by a different thread.  If it did, we have to use the
                          0634          ; stored CSID address because if we were to change the stored one,
                          0634          ; we could confuse the other thread.   Once we determine
                          0634          ; the correct CSID we deallocate the old
                          0634          ; CDRP and message buffer and allocate a new ones.  The deallocate
                          0634          ; and allocate are necessary since CRDPs and message buffers are
                          0634          ; maintained on a per CSB basis.  Also, the CDRP will get
                          0634          ; initialized when it's reallocated.  R3 still contains old
                          0634          ; CSB address
                          0634
              00000002    0634          .IF NE  CA$_MEASURE
   00000000'GF       D6   0634          INCL    G^PMS$GL_DIR_OUT
                          063A          .ENDC
                          063A
     54    18 A2     D0   063A          MOVL    LKMSG$L_CSID(R2),R4      ; Get specified CSID
           38 A1     D1   063E          CMPL    RSB$L_CSID(R1),-         ; Has the CSID in the RSB changed?
           4C A3          0641                  CSB$L_CSID(R3)
              06     13   0643          BEQL    40$                      ; No
     54    38 A1     D0   0645          MOVL    RSB$L_CSID(R1),R4        ; Yes, get new CSID
              3C     13   0649          BEQL    50$                      ; It's now being done locally
   38 A1      54     D0   064B  40$:    MOVL    R4,RSB$L_CSID(R1)        ; Store new CSID
           48 A5     DD   064F          PUSHL   CDRP$L_VAL8(R5)          ; Save PCB address
        7E    50     7D   0652          MOVQ    R0,-(SP)                 ; Save LKB and RSB addresses
           F9A8'     30   0655          BSBW    CNX$DEALL_WARMCDRP_CSB   ; Deallocate old CDRP and message buffer
        53    54     D0   0658          MOVL    R4,R3                    ; Move new CSID
   00000000'GF       95   065B          TSTB    G^LCK$GB_STALLREQS       ; Are we stalling lock requests?
              17     12   0661          BNEQ    47$                      ; Yes, wake process
           F99A'     30   0663          BSBW    CNX$ALLOC_WARMCDRP       ; Alloc. a CDRP with RSPID and cvt CSID
           0A 50     E9   0666          BLBC    R0,45$                   ; Allocation or CSID convert failure
     50    8E        7D   0669          MOVQ    (SP)+,R0                 ; Restore LKB and RSB addresses
        48 A5      8ED0   066C          POPL    CDRP$L_VAL8(R5)          ; Restore PCB address into new CDRP
           FF52      31   0670          BRW     SENDLOCKREQ              ; Resend it
                          0673
                          0673  45$:    ; Unable to allocate a CDRP or CSID returned by other system
                          0673          ; was invalid.  If it's insufficient memory then let process
                          0673          ; do the allocate so that it can wait, if necessary.  If CSID
                          0673          ; returned by the other system is invalid, then bugcheck.
                          0673
   0000'8F    50     B1   0673          CMPW    R0,#SS$_INSFMEM          ; Is it insuff. memory?
              A7     12   0678          BNEQ    35$                      ; No, error
     54       8E     7D   067A  47$:    MOVQ    (SP)+,R4                 ; Restore LKB and RSB addresses
     5E       04     C0   067D          ADDL    #4,SP                    ; Throw away PCB address
                          0680
                          0680  LCK_RETRY:
                          0680          ; Set lock state to RETRY and wake process.  LKB address is in R4.
                          0680
        FE 8F        90   0680          MOVB    #LKB$K_RETRY,-           ; Set RETRY state
        36 A4             0683                  LKB$B_STATE(R4)
           5C        11   0685          BRB     WAKE_PROCESS
                          0687
        F9 8F        90   0687  50$:    MOVB    #LKB$K_RSPDOLOCL,-       ; Change state to handle request
        36 A0             068A                  LKB$B_STATE(R0)          ; locally and fall through to ...
                          068C
                          068C  DOLOCL_FORK:
                          068C          ; Handle request locally
                          068C
              00000002    068C          .IF NE  CA$_MEASURE
```

```
 00000000'GF   D6  068C         INCL    G^PMS$GL_DIR_OUT
                   0692         .ENDC
                   0692
         38 A1 D4  0692         CLRL    RSB$L_CSID(R1)          ; Indicate we will handle this resource
            46 11  0695         BRB     DEALL_WARMCDRP          ; Deallocate CDRP etc. and wake process
                   0697
                   0697 NOTQED_FORK:
                   0697         ; Lock request was not queued due to error or incompatibility
                   0697
         1C A2 B0  0697         MOVW    LKMSG$W_STATUS(R2),-    ; Move completion status to LKB
         2C A0     069A                 LKB$L_LKST1(R0)
            39 11  069C         BRB     70$                     ; Deallocate CDRP etc. and wake process
                   069E
                   069E WAITNG_FORK:
                   069E         ; Lock request has to wait
                   069E
         30 A2 B0  069E         MOVW    LKMSG$W_RQSEQNM(R2),-   ; Store request sequence number
         10 A0     06A1                 LKB$W_RQSEQNM(R0)
            2D 11  06A3         BRB     60$
                   06A5
                   06A5 GRANTD_FORK:
                   06A5         ; Set BLKASTQED bit in LKB if appropriate.  Then store value
                   06A5         ; block from message unless one in RSB is newer.
                   06A5
            01 E1  06A5         BBC     #LKB$V_DBLKAST,-        ; Branch if blocking AST shouldn't
      04 18 A2     06A7                 LKMSG$B_LKBSTATUS(R2),55$ ; be queued
            02 A8  06AA         BISW    #LKB$M_DBLKAST,-        ; Set deliver blocking AST bit
         2A A0     06AC                 LKB$W_STATUS(R0)
         3C A1 C3  06AE 55$:    SUBL3   RSB$L_VALSEQNUM(R1),-   ; Compare sequence numbers
   54    30 A2     06B1                 LKMSG$L_VALSEQNUM(R2),R4
            1C 15  06B4         BLEQ    60$                     ; We already have a newer one
         20 A2 7D  06B6         MOVQ    LKMSG$Q_VALBLK(R2),-    ; Store value block
         28 A1     06B9                 RSB$Q_VALBLK(R1)
         28 A2 7D  06BB         MOVQ    LKMSG$Q_VALBLK+8(R2),-
         30 A1     06BE                 RSB$Q_VALBLK+8(R1)
         30 A2 D0  06C0         MOVL    LKMSG$L_VALSEQNUM(R2),- ; Store sequence number
         3C A1     06C3                 RSB$L_VALSEQNUM(R1)
            02 AA  06C5         BICW    #RSB$M_VALINVLD,-       ; Clear value block invalid flag
         0E A1     06C7                 RSB$W_STATUS(R1)
            01 E1  06C9         BBC     #RSB$V_VALINVLD,-       ; and conditionally set it
      04 1A A2     06CB                 LKMSG$B_RSBSTATUS(R2),60$
            02 A8  06CE         BISW    #RSB$M_VALINVLD,-
         0E A1     06D0                 RSB$W_STATUS(R1)
                   06D2
         10 A2 D0  06D2 60$:    MOVL    LKMSG$L_MSTLKID(R2),-   ; Store remote lock id
         54 A0     06D5                 LKB$L_REMLKID(R0)
                   06D7
       00000002    06D7 70$:    .IF NE  CA$_MEASURE
 00000000'GF   D6  06D7         INCL    G^PMS$GL_ENQNEW_OUT
                   06DD         .ENDC
                   06DD
                   06DD DEALL_WARMCDRP:
                   06DD         ; Deallocate CDRP (R5), message buffer (R2), and RSPID (in CDRP).
                   06DD         ; LKB address is in R0.  CSB address is in R3.
                   06DD
      54    50 D0  06DD         MOVL    R0,R4                   ; Move LKB address
        F91D' 30   06E0         BSBW    CNX$DEALL_WARMCDRP_CSB  ; Deallocate the package
```

```
                                06E3  1472
                                06E3  1473 WAKE_PROCESS:
                                06E3  1474         ; LKB address is in R4.
                                06E3  1475
           51    0C A4     3C   06E3  1476         MOVZWL  LKB$L_PID(R4),R1        ; Get process index
   50    00000000'GF       D0   06E7  1477         MOVL    G^SCH$GL_PCBVEC,R0      ; Get address of PCB vector
           54    6041      D0   06EE  1478         MOVL    (R0)[R1],R4             ; Get address of PCB
                                06F2  1479
                                06F2  1480         ; Change process state to executable
                                06F2  1481
                 52        D4   06F2  1482         CLRL    R2                      ; No priority increment
         00000000'GF       16   06F4  1483         JSB     G^SCH$RSE               ; Report event (RPTEVT not used
                           00'  06FA  1484         .BYTE   EVT$_AST                ; because we need to JSB G^)
                           05   06FB  1485         RSB
                                06FC  1486
                                06FC  1487         .DSABL  LSB
```

```
                        06FC  ;***********************************************************************
                        06FC  ;
                        06FC  ;               Build message action routines for new lock requests
                        06FC  ;               and rebuild lock during failin/out
                        06FC  ;
                        06FC  ;***********************************************************************
                        06FC
                        06FC  BLD_LOCKMSG:
                        06FC          ; Action routine to build the actual lock request message
                        06FC          ; Inputs are:
                        06FC          ;       R2      Address of message buffer
                        06FC          ;       R5      Address of CDRP
                        06FC          ; All registers except R0 and R1 must be preserved
                        06FC
                        06FC          ASSUME  LKMSG$W_HASHVAL  EQ  LKMSG$W_MEMSEQ+2
                        06FC          ASSUME  LKMSG$L_EPIDNEW  EQ  LKMSG$W_HASHVAL+2
                        06FC          ASSUME  LKMSG$L_PRCLKID  EQ  LKMSG$L_EPIDNEW+4
                        06FC          ASSUME  LKMSG$W_FLAGS  EQ  LKMSG$L_PRCLKID+4
                        06FC          ASSUME  LKMSG$B_RQMODE  EQ  LKMSG$W_FLAGS+2
                        06FC          ASSUME  LKMSG$B_GRMODE  EQ  LKMSG$B_RQMODE+1
                        06FC          ASSUME  LKMSG$L_BLKASTFLG  EQ  LKMSG$B_GRMODE+1
                        06FC          ASSUME  LKMSG$L_PARPRCLKID  EQ  LKMSG$L_BLKASTFLG+4
                        06FC          ASSUME  LKMSG$L_PARMSTLKID  EQ  LKMSG$L_PARPRCLKID+4
                        06FC          ASSUME  LKMSG$W_GROUP  EQ  LKMSG$L_PARMSTLKID+4
                        06FC          ASSUME  LKMSG$B_RMOD  EQ  LKMSG$W_GROUP+2
                        06FC          ASSUME  LKMSG$B_RSNLEN  EQ  LKMSG$B_RMOD+1
                        06FC          ASSUME  LKMSG$T_RESNAM  EQ  LKMSG$B_RSNLEN+1
                        06FC
                        06FC          ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
08 A2     0102 8F  B0   06FC          MOVW    #LKMSG$K_NEWLOCK@8-        ; Store function and facility codes
                        0702                  !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
                        0702  BLD_LOCKCOM:
               3C  BB   0702          PUSHR   #^M<R2,R3,R4,R5>           ; Save registers
     50     2C A5  7D   0704          MOVQ    CDRP$L_VAL1(R5),R0         ; R0 = LKB address;  R1 = RSB address
        52     0C  C0   0708          ADDL    #LKMSG$W_MEMSEQ,R2         ; Point inside message
     82     34 A5  B0   070B          MOVW    CDRP$L_VAL3(R5),(R2)+      ; Store rebuild seq. number
     82     44 A1  B0   070F          MOVW    RSB$W_HASHVAL(R1),(R2)+    ; Store resource hash value
     54     48 A5  D0   0713          MOVL    CDRP$L_VAL8(R5),R4         ; Get PCB address
               3C  13   0717          BEQL    20$                        ; Branch if system owned (rebuild only)
     82     64 A4  D0   0719          MOVL    PCB$L_EPID(R4),(R2)+       ; Store EPID
          010C C4  D0   071D          MOVL    PCB$L_DLCKPRI(R4),-        ; Store deadlock priority
            38 A2       0721                  LKMSG$L_DLCKPRI_NEW-LKMSG$L_PRCLKID(R2)
     82     30 A0  D0   0723  5$:     MOVL    LKB$L_LKID(R0),(R2)+       ; Store process lock id
     82     28 A0  B0   0727          MOVW    LKB$W_FLAGS(R0),(R2)+      ; Store input flags
     82     34 A0  B0   072B          MOVW    LKB$B_RQMODE(R0),(R2)+     ; Store lock modes
            20 A0  D0   072F          MOVL    LKB$L_BLKASTADR(R0),-      ; Set blocking AST indicator.  Note:
               82       0732                  (R2)+                      ; address is not used - just zero or not
               53  D4   0733          CLRL    R3                         ; Assume no parent lock id
     54     48 A0  D0   0735          MOVL    LKB$L_PARENT(R0),R4        ; Get address of parent LKB
               08  13   0739          BEQL    10$                        ; No parent
     53     30 A4  D0   073B          MOVL    LKB$L_LKID(R4),R3          ; Get parent's process lock id
     54     54 A4  D0   073F          MOVL    LKB$L_REMLKID(R4),R4       ; and master lock id
        82     53  7D   0743  10$:    MOVQ    R3,(R2)+                   ; Store both parent lock ids
     50     4F A1  9A   0746          MOVZBL  RSB$B_RSNLEN(R1),R0        ; Get resource name length
        50     04  C0   074A          ADDL    #4,R0                      ; Account for adjacent fields
62   4C A1  50     28   074D          MOVC3   R0,RSB$W_GROUP(R1),(R2)    ; Move group number, access mode,
                        0752                                             ; res. name length and resource name
```

```
               3C  BA 0752            POPR    #^M<R2,R3,R4,R5>        ; Restore registers
                   05 0754            RSB
                      0755
                      0755  20$:      ; Have a zero for PCB address.  This should only occur doing rebuild
                      0755            ; for system owned locks.  Verify CVTSYS bit is set.
                      0755
         52   04   C0 0755            ADDL    #4,R2                   ; Move past EPID field
              06   E0 0758            BBS     #LCK$V_CVTSYS,-         ; Continue if CVTSYS is set
         C6 28 A0     075A                    LKB$W_FLAGS(R0),5$
                      075D            BUG_CHECK       LOCKMGRERR,FATAL; CVTSYS not set and PCB address = 0
                      0761
                      0761  LCK$BLD_REBLDLOCK::
 08 A2   0802 8F   B0 0761            MOVW    #LKMSG$K_REBLDLOCK@8-   ; Store function and facility codes
                      0767                    !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
              99   10 0767            BSBB    BLD_LOCKCOM             ; Build normal portion of message
         50   2C A5   7D 0769         MOVQ    CDRP$L_VAL1(R5),R0      ; R0 = LKB address;  R1 = RSB address
              10 A0   B0 076D         MOVW    LKB$W_RQSEQNM(R0),-     ; Store request sequence number
              50 A2      0770                 LKMSG$W_RQSEQALT(R2)
              28 A1   7D 0772         MOVQ    RSB$Q_VALBLK(R1),-      ; Store value block
              54 A2      0775                 LKMSG$Q_VALBLKALT(R2)
              30 A1   7D 0777         MOVQ    RSB$Q_VALBLK+8(R1),-
              5C A2      077A                 LKMSG$Q_VALBLKALT+8(R2)
              3C A1   D0 077C         MOVL    RSB$L_VALSEQNUM(R1),-   ; Store value block sequence number
              64 A2      077F                 LKMSG$L_VALSEQALT(R2)
                         0781
                         0781         ASSUME  LKMSG$B_RSTATUS  EQ LKMSG$B_LSTATUS+1
                         0781         ASSUME  LKB$V_ASYNC  LE  7
                         0781         ASSUME  LKB$V_BLKASTQED  LE  7
                         0781         ASSUME  RSB$V_VALINVLD  LE  7
                         0781
              68 A2   B4 0781         CLRW    LKMSG$B_LSTATUS(R2)     ; Clear LKB and RSB status fields in msg
 68 A2  2A A0  F3 8F  8B 0784         BICB3   #^C<LKB$M_BLKASTQED!-   ; Set specified LKB status bits
                         078B                 LKB$M_ASYNC>,LKB$W_STATUS(R0),LKMSG$B_LSTATUS(R2)
               FD 8F  8B 078B         BICB3   #^C<RSB$M_VALINVLD>,-   ; Set specified RSB status bits
        69 A2  0E A1     078E                 RSB$W_STATUS(R1),LKMSG$B_RSTATUS(R2)
           51  36 A0  90 0792         MOVB    LKB$B_STATE(R0),R1      ; Get lock state
                         0796
                         0796         ; Handle special-cased lock states
                         0796
                         0796         DISPATCH        R1,TYPE=B,PREFIX=LKB$K_,-
                         0796                 <-
                         0796                 <GRANTED,50$>,-
                         0796                 <CONVERT,50$>,-
                         0796                 <WAITING,50$>,-
                         0796                 <RETRY,20$>,-
                         0796                 <SCSWAIT,20$>,-
                         0796                 >
                         07A5
                         07A5         BUG_CHECK       LOCKMGRERR,FATAL        ; Illegal lock state
                         07A9
                         07A9  20$:   ; Lock state is RETRY or SCSWAIT.  This must be a
                         07A9         ; conversion request as new lock requests in this state aren't even
                         07A9         ; rebuilt.  Store the old blocking AST address in the message because
                         07A9         ; that is what the master copy should hold at this point.  Then change
                         07A9         ; the lock state to GRANTED.
                         07A9
              5C A0   D0 07A9         MOVL    LKB$L_OLDBLKAST(R0),-   ; Store old blocking AST address
```

```
      1C A2        07AC  1603                LKMSG$L_BLKASTFLG(R2)
   51    01   90   07AE  1604        MOVB    #LKB$K_GRANTED,R1         ; Set state to GRANTED
                   07B1  1605
6A A2    51   90   07B1  1606 50$:   MOVB    R1,LKMSG$B_LCKSTATE(R2)  ; Store lock state
              05   07B5  1607        RSB
```

```
07B6          .SBTTL  LCK$RCV_LOCKREQ - Receive lock request
07B6
07B6    ;++
07B6    ; FUNCTIONAL DESCRIPTION
07B6    ;
07B6    ;       This routine receives new lock requests from the remote
07B6    ;       system.  The other system is the process system and this system is
07B6    ;       the master system.  Root lock requests are special-cased in that
07B6    ;       the root directory logic is involved (see below).
07B6    ;
07B6    ; CALLING SEQUENCE:
07B6    ;
07B6    ;       JSB     LCK$RCV_LOCKREQ (called from SCS received message routine)
07B6    ;       IPL must be at IPL$_SYNCH
07B6    ;
07B6    ; INPUT PARAMETERS:
07B6    ;
07B6    ;       R2      Address of message buffer
07B6    ;       R3      Address of CSB
07B6    ;       R5      Address of CDRP for response message
07B6    ;
07B6    ; OUTPUT PARAMETERS:
07B6    ;
07B6    ;       None
07B6    ;
07B6    ; SIDE EFFECTS:
07B6    ;
07B6    ;       A message may be sent back as a response
07B6    ;
07B6    ;
07B6    ; DESCRIPTION OF DIRECTORY PROCESSING LOGIC:
07B6    ;
07B6    ;
07B6    ;       If this is a sub-resource
07B6    ;           then
07B6    ;               Get address of parent resource;
07B6    ;           else;
07B6    ;       Search resource hash table for a matching resource;
07B6    ;       If resource found
07B6    ;           then
07B6    ;               If resource is managed locally (CSB address = 0)
07B6    ;                   then
07B6    ;                       Handle lock request here;
07B6    ;                       Send response (granted, waiting, blocking, or notqueued);
07B6    ;                   else
07B6    ;                       Bugcheck if this is a sub-resource
07B6    ;                       If system managing the resource is the same
07B6    ;                       system that sent us this request      [DIRECTORY LOOKUP]
07B6    ;                           then
07B6    ;                               Send response (do locally);
07B6    ;                           else
07B6    ;                               Send response (resend to specified
07B6    ;                               system);
07B6    ;           else
07B6    ;               If this is a root resource (PARENT = 0)
07B6    ;                   then
07B6    ;                       If this is the directory node for this resource
```

```
                      07B6  ;                                       then
                      07B6  ;                                           Create directory entry      [ENTER IN DIRECTORY]
                      07B6  ;                                           Send response (do locally);
                      07B6  ;                                       else
                      07B6  ;                                           Send response (re-lookup in directory);
                      07B6  ;                                                                      [DIRECTORY RACE]
                      07B6  ;                               else
                      07B6  ;                                   Bugcheck if parent is not managed here (CSID not 0)
                      07B6  ;                                   Handle lock request here
                      07B6  ;                                   Send response (granted, blocking, waiting or
                      07B6  ;                                   not queued);
                      07B6  ;
                      07B6  ;--
                      07B6
                      07B6  LCK$RCV_LOCKREQ::
     51   53    D0    07B6          MOVL    R3,R1                           ; Move CSB address
   0FC6   8F    BB    07B9          PUSHR   #^M<R1,R2,R6,R7,R8,R9,R10,R11>
          55    DD    07BD          PUSHL   R5                              ; Save CDRP address
       4C A3    DD    07BF          PUSHL   CSB$L_CSID(R3)                  ; Convert CSB address to CSID and save
     59   52    D0    07C2          MOVL    R2,R9                           ; Save input message address
      F838'     30    07C5          BSBW    CNX$INIT_CDRP                   ; Initialize CDRP
                      07C8
                      07C8          ; If this is not a root resource request then get address of parent
                      07C8          ; RSB (needed for hash table search) and verify that the resource
                      07C8          ; is managed on this system.
                      07C8
          57    D4    07C8          CLRL    R7                              ; Assume no parent RSB
  56   24 A9    D0    07CA          MOVL    LKMSG$L_PARMSTLKID(R9),R6       ; Get parent lock id
          11    13    07CE          BEQL    20$                             ; No parent - this is a root resource
     51   56    D0    07D0          MOVL    R6,R1                           ; Move parent id
  50   20 A9    D0    07D3          MOVL    LKMSG$L_PARPRCLKID(R9),R0       ; Get remote parent lock id for verify
        07FD    30    07D7          BSBW    VERIFYREMLKID                   ; Verify lockid and convert to LKB addr.
       23 50    E9    07DA          BLBC    R0,80$                          ; Invalid lock id
  57   50 A6    D0    07DD          MOVL    LKB$L_RSB(R6),R7                ; Get parent RSB address
  24 A9   57    D0    07E1  20$:    MOVL    R7,LKMSG$L_PARMSTLKID(R9)       ; Store in message for convenience
                      07E5
                      07E5          ; Search hash table for a matching resource.  The input message
                      07E5          ; is used as the template to match.
                      07E5
                      07E5          ASSUME  LKMSG$W_GROUP   EQ  LKMSG$L_PARMSTLKID+4
                      07E5          ASSUME  LKMSG$B_RMOD    EQ  LKMSG$W_GROUP+2
                      07E5          ASSUME  LKMSG$B_RSNLEN  EQ  LKMSG$B_RMOD+1
                      07E5          ASSUME  LKMSG$T_RESNAM  EQ  LKMSG$B_RSNLEN+1
                      07E5
  54   24 A9    9E    07E5          MOVAB   LKMSG$L_PARMSTLKID(R9),R4       ; Point to start of fields to match
  5A   2B A9    9A    07E9          MOVZBL  LKMSG$B_RSNLEN(R9),R10          ; Get length of resource name
     5A   08    C0    07ED          ADDL    #8,R10                          ; Account for other fields to match
  51   0E A9    3C    07F0          MOVZWL  LKMSG$W_HASHVAL(R9),R1          ; Get resource hash value
 00000000'GF    16    07F4          JSB     G^LCK$SRCH_HSHTBL               ; Search hash table
       19 50    E9    07FA          BLBC    R0,EXISTING_RESOURCE
        0150    31    07FD          BRW     NEW_RESOURCE
                      0800
                      0800  80$:    ; Other system sent us an invalid lock id
                      0800
                      0800          BUG_CHECK       INVLOCKID,FATAL
```

```
                        0804  ;*************************************************************************
                        0804  ;
                        0804  ;                  Existing resource
                        0804  ;
                        0804  ;*************************************************************************
                        0804
                        0804          .ENABL  LSB
                        0804
                        0804  5$:     ; It looks like we should perform a directory lookup.  However,
                        0804          ; if we are rebuilding a lock during failover, then additional
                        0804          ; checking must be performed.
                        0804
          09 A9    91   0804          CMPB    CLSMSG$B_FUNC(R9),-       ; Is this a rebuild lock message
             08         0807                  #LKMSG$K_REBLDLOCK        ; (only sent during failover)?
             09    12   0808          BNEQ    10$                       ; No
       55    6E    D0   080A          MOVL    (SP),R5                   ; Yes, get CSID of requesting system
           F7F0'   30   080D          BSBW    LCK$CHECK_DIRENTRY        ; Check if it's really a directory entry
          0C 50    E9   0810          BLBC    R0,15$                    ; It's not - build the lock
            0096   31   0813  10$:    BRW     DIR_LOOKUP                ; Do a directory lookup
                        0816
                        0816  EXISTING_RESOURCE:
                        0816          ; We've found a matching RSB.  If the resource is being
                        0816          ; managed by another system (CSID non-zero) then treat
                        0816          ; this request as a directory lookup.  Otherwise, handle it
                        0816          ; in this system.  Registers contain:
                        0816          ;
                        0816          ;       R5      Address of matching RSB
                        0816          ;       R6      Address of parent LKB (or 0)
                        0816          ;       R7      Address of parent RSB (or 0)
                        0816          ;       R9      Address of input message
                        0816
          58 55    D0   0816          MOVL    R5,R8                     ; R8 will be used to point to RSB
     54   38 A8    D0   0819          MOVL    RSB$L_CSID(R8),R4         ; Get CSID
             E5    12   081D          BNEQ    5$                        ; Do a directory lookup
                        081F
        00000002        081F  15$:    .IF NE  CA$_MEASURE
   00000000'GF     D6   081F          INCL    G^PMS$GL_ENQNEW_IN
                        0825          .ENDC
                        0825
            02AF   30   0825          BSBW    BUILD_LKB                 ; Build the lock block
          17 50    E9   0828          BLBC    R0,19$                    ; Exit on any error
    58 A6    6E    D0   082B          MOVL    (SP),LKB$L_CSID(R6)       ; Store CSID of requesting system
          09 A9    91   082F          CMPB    CLSMSG$B_FUNC(R9),-       ; Is this a normal lock message?
             01         0832                  #LKMSG$K_NEWLOCK          ; (as opposed to a failover message)
             10    13   0833          BEQL    20$                       ; Yes
          09 A9    91   0835          CMPB    CLSMSG$B_FUNC(R9),-       ; Is this a rebuild lock message?
             08         0838                  #LKMSG$K_REBLDLOCK
             03    12   0839          BNEQ    18$                       ; No, must be a directory entry message
            01C0   31   083B          BRW     REBLD_LOCK                ; Yes, rebuild the lock
                        083E  18$:    BUG_CHECK       LOCKMGRERR,FATAL; Disagreement regarding who's mastering
                        0842                                            ; this lock tree
            01D1   31   0842  19$:    BRW     BLDLKB_ERROR2             ; Branch extender
                        0845
                        0845  20$:    ; Now see if the lock can be granted.  To be granted the conversion
                        0845          ; and wait queues must be empty and the requested lock mode must be
                        0845          ; compatible with the group grant mode of the resource.  (Actually,
                        0845          ; any locks on the wait queue in states other than LKB$W_WAITING
```

```
                            0845          ; are ignored).
                            0845
                            0845          ASSUME  RSB$L_WTQFL  EQ  RSB$L_CVTQFL+8
                            0845
          50   18 A8    DE  0845          MOVAL   RSB$L_CVTQFL(R8),R0     ; Get address of conversion queue
               60   50  D1  0849          CMPL    R0,(R0)                 ; Queue empty?
                    2F  12  084C          BNEQ    50$                     ; No
               50   08  C0  084E          ADDL    #8,R0                   ; Get address of wait queue
               51   50  D0  0851          MOVL    R0,R1                   ; Save in R1
               51   60  D1  0854 30$:     CMPL    (R0),R1                 ; Queue empty?
                    1A  12  0857          BNEQ    40$                     ; No
                            0859
                            0859 35$:     ; No waiting requests (or RECOVER bit is set); is the lock compatible?
                            0859
          51   34 A6    9A  0859          MOVZBL  LKB$B_RQMODE(R6),R1     ; Get lock mode
          55   0C A8    9A  085D          MOVZBL  RSB$B_GGMODE(R8),R5     ; Get group grant mode
 18 00000000'GF45  51   E1  0861          BBC     R1,G^LCK$COMPAT_TBL[R5],52$ ; Branch if incompatible
                            086A
                            086A          ; Lock can be granted
                            086A
        00000000'GF     16  086A          JSB     G^LCK$GRANT_LOCK                ; Grant the lock
                016C    31  0870          BRW     LOCK_GRANTED
                            0873
                            0873 40$:     ; Wait queue was not empty.  Skip over locks not in LKB$K_WAITING
                            0873          ; state.  If we find even one lock in LKB$K_WAITING state, then
                            0873          ; this lock is not grantable.
                            0873
               50   60  D0  0873          MOVL    (R0),R0                 ; Get next lock on wait queue
               FF 8F    91  0876          CMPB    #LKB$K_WAITING,-        ; Is it in LKB$K_WAITING state?
               FE A0        0879                  LKB$B_STATE-LKB$L_SQFL(R0)
                  D7    12  087B          BNEQ    30$                     ; No, skip over it
                            087D
                            087D 50$:     ; Lock cannot be granted now due to other locks in the waiting
                            087D          ; or conversion queue.  However, if the RECOVER bit is set, then
                            087D          ; ignore these locks and try granting it anyway
                            087D
                  07    E0  087D          BBS     #LCK$V_RECOVER,-        ; Branch if recovering a lock
            D7 28 A6        087F                  LKB$W_FLAGS(R6),35$
                            0882
                            0882 52$:     ; Lock cannot be granted now.  Queue it unless LCK$M_NOQUEUE
                            0882          ; is set.
                            0882
                  02    E0  0882          BBS     #LCK$V_NOQUEUE,-        ; Branch if the NOQUEUE bit is set
            15 28 A6        0884                  LKB$W_FLAGS(R6),55$
               46 A8    B0  0887          MOVW    RSB$W_RQSEQNM(R8),-     ; Store request sequence number
               10 A6        088A                  LKB$W_RQSEQNM(R6)
               46 A8    B6  088C          INCW    RSB$W_RQSEQNM(R8)       ; Increment sequence number
        00000000'GF     16  088F          JSB     G^LCK$QUEUEWAIT         ; Queue it
          50   FB 8F    90  0895          MOVB    #LKB$K_RSPQUEUED,R0     ; Set response state
                014B    31  0899          BRW     LOCK_QUEUED
                            089C
                            089C 55$:     ; Lock should not be queued.  Clean up and exit.
                            089C
            00000002        089C          .IF NE  CA$_MEASURE
        00000000'GF     D6  089C          INCL    G^PMS$GL_ENQNOTQD
                            08A2          .ENDC
                            08A2
```

```
             02C0   30  08A2  1836        BSBW    LCK$DEALLOC_LKB         ; Deallocate LKB, lockid, etc.
     50    0000'8F  3C  08A5  1837        MOVZWL  #SS$_NOTQUEUED,R0       ; Return status
               67   11  08AA  1838 60$:   BRB     RESPOND_NOTQED
                        08AC  1839
                        08AC  1840        .DSABL  LSB
```

```
                       08AC  ;*****************************************************************************
                       08AC  ;
                       08AC  ;                  Directory handling and error routines
                       08AC  ;
                       08AC  ;*****************************************************************************
                       08AC
                       08AC          .ENABL  LSB
                       08AC
                       08AC  DIR_LOOKUP:
                       08AC          ; Treat this request as a directory lookup request by returning
                       08AC          ; the CSID of the system managing this resource.  If the
                       08AC          ; CSID matches that of the requestor then return a different
                       08AC          ; response state to indicate this.  R4 contains CSID of
                       08AC          ; system managing resource.
                       08AC
         00000002      08AC          .IF NE  CA$_MEASURE
      00000000'GF  D6  08AC          INCL    G^PMS$GL_DIR_IN
                       08B2          .ENDC
                       08B2
            48 A8  D5  08B2          TSTL    RSB$L_PARENT(R8)        ; Directory entries must be roots
               17  12  08B5          BNEQ    40$                     ; Error
                       08B7
                       08B7  RESPOND_RESEND:
            53 8ED0    08B7          POPL    R3                      ; Restore CSID of incoming message
            55 8ED0    08BA          POPL    R5                      ; Restore CDRP address
         54    53  D1  08BD          CMPL    R3,R4                   ; Is requestor managing resource?
               4A  13  08C0          BEQL    RESPOND_DOLOCL          ; Yes, do locally (on remote system)
      2C A5    54  D0  08C2          MOVL    R4,CDRP$L_VAL1(R5)      ; No, return CSID of other system
         F8 8F     90  08C6          MOVB    #LKB$K_RSPRESEND,-      ; Set response state
            32 A5      08C9                  CDRP$L_VAL2+2(R5)
          0054     31  08CB          BRW     NOTQUEUED_EXIT
                       08CE
                       08CE  40$:    BUG_CHECK       LOCKMGRERR,FATAL
                       08D2
                       08D2  RELOOKUP:
                       08D2          ; Send a response telling requesting system to resend this
                       08D2          ; request to directory system (CSID in R4).  Note that it is
                       08D2          ; possible for the requesting system to be the directory system
                       08D2          ; for this resource in which case a "do locally" response is sent.
                       08D2          ; This situation is caused by the following race:
                       08D2          ;       o  This system masters the resource
                       08D2          ;       o  The other system (directory system) sends a request here
                       08D2          ;       o  This system removes the directory entry
                       08D2          ;       o  This system receives the request and returns "do locally"
                       08D2          ; The danger with this race is that the other system doesn't realize
                       08D2          ; it has a directory entry so doesn't set the DIRENTRY bit.  What
                       08D2          ; saves us is that it is already set and LCK$RCV_RMVDIR will not
                       08D2          ; delete the RSB.
                       08D2
               6B  D4  08D2          CLRL    RSB$L_HSHCHN(R11)       ; Remove RSB from hash chain
         50    58  D0  08D4          MOVL    R8,R0
      00000000'GF  16  08D7          JSB     G^EXE$DEANONPAGED       ; Deallocate RSB
               D8  11  08DD          BRB     RESPOND_RESEND
                       08DF
                       08DF  DIR_ENTER:
                       08DF          ; If we are the directory system for this resource, then handle
                       08DF          ; this request by making a directory entry.  Otherwise, tell requesting
```

```
                              08DF              ; system to re-lookup the request in the directory as we've been
                              08DF              ; bagged by a directory lookup race.
                              08DF
      53  00000000'GF    D0   08DF          MOVL    G^LCK$GL_DIRVEC,R3          ; Get address of directory vector
          51   44 A8     3C   08E6          MOVZWL  RSB$W_HASHVAL(R8),R1        ; Get hash value
                   52    D4   08EA          CLRL    R2                          ; Clear high order hash value
   51   50   51   F4 A3  7B   08EC          EDIV    -12(R3),R1,R0,R1            ; Compute hash index (in R1)
             54   6341   D0   08F2          MOVL    (R3)[R1],R4                 ; Get directory system
                   DA    12   08F6          BNEQ    RELOOKUP                    ; It's not us
                   01    A8   08F8          BISW    #RSB$M_DIRENTRY,-           ; Set directory entry bit
                0E A8         08FA                  RSB$W_STATUS(R8)
                              08FC
             00000002         08FC          .IF NE  CA$_MEASURE
         00000000'GF     D6   08FC          INCL    G^PMS$GL_DIR_IN
                              0902          .ENDC
                              0902
               53 8ED0        0902          POPL    R3                          ; Restore CSID
               55 8ED0        0905          POPL    R5                          ; Restore CDRP address
          38 A8    53    D0   0908          MOVL    R3,RSB$L_CSID(R8)           ; Store CSID in RSB to form dir. entry
                              090C
                              090C  RESPOND_DOLOCL:
                              090C          ; Send a response that tells the requestor to handle the lock request
                              090C          ; locally.
                              090C
                F9 8F    90   090C          MOVB    #LKB$K_RSPDOLOCL,-          ; Store response state
                32 A5         090F                  CDRP$L_VAL2+2(R5)
                   0F    11   0911          BRB     NOTQUEUED_EXIT
                              0913
                              0913  RESPOND_NOTQED:
                              0913          ; Lock was not queued due to an error.  Status is in R0
                              0913          ;   (e.g. SS$_NOTQUEUED)
                              0913
               53 8ED0        0913          POPL    R3                          ; Restore CSID
               55 8ED0        0916          POPL    R5                          ; Restore CDRP address
          30 A5    50    B0   0919          MOVW    R0,CDRP$L_VAL2(R5)          ; Store error status
                FC 8F    90   091D          MOVB    #LKB$K_RSPNOTQED,-          ; Store response state
                32 A5         0920                  CDRP$L_VAL2+2(R5)           ; and fall thru to ...
                              0922
                              0922  NOTQUEUED_EXIT:
            0A5A'CF      9E   0922          MOVAB   W^BLD_RSPMSG,-              ; Store address of message build
                4C A5         0926                  CDRP$L_MSGBLD(R5)           ; routine
                              0928  QUEUED_EXIT:
      50  00000000'GF    D0   0928          MOVL    G^CLU$GL_CLUB,R0            ; Get address of CLUB
             00AC C0     B0   092F          MOVW    CLUB$W_MEMSEQ(R0),-         ; Store membership seq. num.
                34 A5         0933                  CDRP$L_VAL3(R5)
              0FC6 8F    BA   0935          POPR    #^M<R1,R2,R6,R7,R8,R9,R10,R11>
                 F6C4'   30   0939          BSBW    CNX$SEND_MSG_RESP           ; Send message response
         0000'8F   50    B1   093C          CMPW    R0,#SS$_NOSUCHNODE          ; Is status SS$_NOSUCHNODE?
                   09    13   0941          BEQL    60$                         ; Yes, error
              50   55    D0   0943          MOVL    R5,R0                       ; No, SS$_NODELEAVE is okay
         00000000'GF     17   0946          JMP     G^EXE$DEANONPAGED           ; Deallocate CDRP
                              094C
                              094C  60$:    BUG_CHECK       LOCKMGRERR,FATAL; CSID invalid
                              0950
                              0950          .DSABL  LSB
```

```
                          0950  ;*********************************************************************
                          0950  ;
                          0950  ;                  New Resource
                          0950  ;
                          0950  ;*********************************************************************
                          0950
                          0950          ; Resource was not found so it must be created.  If it's a root
                          0950          ; resource and we are the directory system, then a directory
                          0950          ; entry will be made but the lock will be handled by the remote
                          0950          ; system.  If it's a root resource but we are not the directory
                          0950          ; system then the remote system will be told to re-lookup this
                          0950          ; resource in the directory.  If it's not a root resource then the
                          0950          ; lock request is handled here.  Registers contain:
                          0950          ;
                          0950          ;       R6      Address of parent LKB (or 0)
                          0950          ;       R7      Address of parent RSB (or 0)
                          0950          ;       R9      Address of input message
                          0950          ;       R10     Length of resource name+8
                          0950          ;       R11     Address of last RSB in resource hash chain
                          0950
                          0950  NEW_RESOURCE:
      51   48 AA    9E    0950          MOVAB   RSB$K_LENGTH-8(R10),R1   ; Compute size of RSB
  00000000'GF       16    0954          JSB     G^EXE$ALONONPAGED        ; Allocate RSB
         4B 50      E9    095A          BLBC    R0,40$                   ; Didn't get one
   08 A2    51      3C    095D          MOVZWL  R1,RSB$W_SIZE(R2)        ; Store size and zero depth
   0A A2    36      90    0961          MOVB    #DYN$C_RSB,RSB$B_TYPE(R2) ; Store type
      58    52      D0    0965          MOVL    R2,R8                    ; R8 points to RSB
                          0968
                          0968          ; Initialize various fields in RSB
                          0968
                          0968          ASSUME  RSB$B_CGMODE  EQ  RSB$B_GGMODE+1
                          0968          ASSUME  RSB$W_STATUS  EQ  RSB$B_CGMODE+1
                          0968          ASSUME  RSB$W_REFCNT  EQ  RSB$L_VALSEQNUM+4
                          0968          ASSUME  RSB$W_BLKASTCNT  EQ RSB$W_REFCNT+2
                          0968          ASSUME  RSB$L_CVTQFL  EQ  RSB$L_GRQFL+8
                          0968          ASSUME  RSB$L_WTQFL  EQ  RSB$L_CVTQFL+8
                          0968          ASSUME  RSB$W_RQSEQNM  EQ  RSB$W_HASHVAL+2
                          0968
      6B    58      D0    0968          MOVL    R8,RSB$L_HSHCHN(R11)     ; Make last RSB point to this one
            68      D4    096B          CLRL    RSB$L_HSHCHN(R8)         ; This one ends the chain
   04 A8    5B      D0    096D          MOVL    R11,RSB$L_HSHCHNBK(R8)   ; Point to previous one
   50    10 A8      DE    0971          MOVAL   RSB$L_GRQFL(R8),R0       ; Initialize all three queue headers
      51    50      D0    0975          MOVL    R0,R1
      81    50      D0    0978          MOVL    R0,(R1)+                 ; Granted queue
      81    80      7E    097B          MOVAQ   (R0)+,(R1)+
      81    50      D0    097E          MOVL    R0,(R1)+                 ; Conversion queue
      81    80      7E    0981          MOVAQ   (R0)+,(R1)+
      81    50      D0    0984          MOVL    R0,(R1)+                 ; Waiting queue
      61    50      D0    0987          MOVL    R0,(R1)
         28 A8      7C    098A          CLRQ    RSB$Q_VALBLK(R8)         ; Clear value block
         30 A8      7C    098D          CLRQ    RSB$Q_VALBLK+8(R8)
         3C A8      7C    0990          CLRQ    RSB$L_VALSEQNUM(R8)      ; Clear value block seq. number,
                          0993                                           ; ref. count and blocking AST cnt.
         0C A8      D4    0993          CLRL    RSB$B_GGMODE(R8)         ; Clear modes, status
   0E A9            3C    0996          MOVZWL  LKMSG$W_HASHVAL(R9),-    ; Store hash value, clear request
   44 A8                  0999                  RSB$W_HASHVAL(R8)        ; sequence number
            5A      28    099B          MOVC3   R10,-                    ; Move parent RSB address, group number,
```

```
                        099D                  LKMSG$L_PARMSTLKID(R9),-; access mode, res. name length and
        48 A8   24 A9   099D                  RSB$L_PARENT(R8)        ; resource name to RSB
                        09A1
                        09A1           ; If this is a root resource, then just create a directory entry
                        09A1           ; and don't build a LKB.
                        09A1
             57   D5    09A1           TSTL    R7                      ; Is this a root resource?
             0A   12    09A3           BNEQ    60$                     ; No
           FF37   31    09A5           BRW     DIR_ENTER               ; Yes
                        09A8
           006B   31    09A8 40$:      BRW     BLDLKB_ERROR2           ; Branch extender
                        09AB
                        09AB 50$:      ; Received a lock request for a lock whose parent is not managed
                        09AB           ; by this system.
                        09AB
                        09AB           BUG_CHECK        LOCKMGRERR,FATAL
                        09AF
     0B A7   01   81    09AF 60$:      ADDB3   #1,RSB$B_DEPTH(R7),-    ; Set depth
          0B A8         09B3                   RSB$B_DEPTH(R8)
          40 A7   B6    09B5           INCW    RSB$W_REFCNT(R7)        ; Incr. parent's reference count
          38 A7   D0    09B8 65$:      MOVL    RSB$L_CSID(R7),-        ; Clear CSID by copying parent's
          38 A8         09BB                   RSB$L_CSID(R8)          ; and verifying it's zero
             EC   12    09BD           BNEQ    50$                     ; Error
                        09BF
                        09BF           ; Build the LKB.
                        09BF
       00000002         09BF           .IF NE  CA$_MEASURE
    00000000'GF   D6    09BF           INCL    G^PMS$GL_ENQNEW_IN
                        09C5           .ENDC
                        09C5
           010F   30    09C5           BSBW    BUILD_LKB
          38 50   E9    09C8           BLBC    R0,BLDLKB_ERROR         ; Unable to build LKB.  Status in R0
    58 A6    6E   D0    09CB           MOVL    (SP),LKB$L_CSID(R6)     ; Store CSID or requesting system
       09 A9      91    09CF           CMPB    CLSMSG$B_FUNC(R9),-     ; Is this a rebuild lock message
          08            09D2                   #LKMSG$K_REBLDLOCK      ; (only sent during failover)?
          29      13    09D3           BEQL    REBLD_LOCK              ; Yes
                        09D5
                        09D5           ; Grant the lock.
                        09D5
    51   34 A6    9A    09D5           MOVZBL  LKB$B_RQMODE(R6),R1     ; Get requested mode
    00000000'GF   16    09D9           JSB     G^LCK$GRANT_LOCK_ALT    ; Grant the lock
                        09DF
                        09DF LOCK_GRANTED:
             02   AA    09DF           BICW    #LKB$M_DBLKAST,-        ; Got set spuriously by common code
          2A A6         09E1                   LKB$W_STATUS(R6)
    50   FA 8F    90    09E3           MOVB    #LKB$K_RSPGRANTD,R0     ; Set response state
                        09E7
                        09E7 LOCK_QUEUED:
                        09E7           ; Lock response state is in R0.  It is either RSPGRANTD or
                        09E7           ; RSPQUEUED.  I.e. it is one of the states that actually
                        09E7           ; built and queued a LKB.  Send response message.
                        09E7
          53 8ED0       09E7           POPL    R3                      ; Restore CSID
          55 8ED0       09EA           POPL    R5                      ; Restore CDRP address
       0A6B'CF    9E    09ED           MOVAB   W^BLD_LOCKRSP,-         ; Store address of message build routine
          4C A5         09F1                   CDRP$L_MSGBLD(R5)
    2C A5    56   D0    09F3           MOVL    R6,CDRP$L_VAL1(R5)      ; Store LKB address
```

```
      32 A5    50    90 09F7           MOVB    R0,CDRP$L_VAL2+2(R5)      ; Store response state
        FF2A   31       09FB           BRW     QUEUED_EXIT
                        09FE
                        09FE   REBLD_LOCK:
                        09FE           ; Rebuild lock (only performed during failover)
                        09FE
        F5FF'  30       09FE           BSBW    LCK$REBLD_LOCK            ; Rebuild the lock
          DC   11       0A01           BRB     LOCK_GRANTED              ; Send the response
                        0A03
                        0A03   BLDLKB_ERROR:
                        0A03           ; Error status in R0.  Parent RSB address in R7.
                        0A03           ; Address of previous RSB in hash chain in R11.  Deallocate RSB.
                        0A03
          50   DD       0A03           PUSHL   R0                        ; Save R0
       40 A7   B7       0A05           DECW    RSB$W_REFCNT(R7)          ; Decrement's parent's reference count
          6B   D4       0A08           CLRL    RSB$L_HSHCHN(R11)         ; Remove from hash chain
      50  58   D0       0A0A           MOVL    R8,R0
 00000000'GF   16       0A0D           JSB     G^EXE$DEANONPAGED         ; Deallocate RSB
       50 8ED0          0A13           POPL    R0                        ; Restore R0
                        0A16
                        0A16   BLDLKB_ERROR2:
                        0A16           ; Error handling depends on whether this is a rebuild lock
                        0A16           ; (during failover) or a lock request during normal operation.
                        0A16           ; During normal operation, all errors other than SS$_INSFMEM
                        0A16           ; are sent back to the other system.  SS$_INSFMEM is handled
                        0A16           ; by breaking our connection as a form of waiting.
                        0A16           ; During failover, we bugcheck here with an appropriate
                        0A16           ; type of resource exhausted bugcheck.
                        0A16
       09 A9   91       0A16           CMPB    CLSMSG$B_FUNC(R9),-       ; Is this a rebuild lock message?
          08            0A19                   #LKMSG$K_REBLDLOCK
          24   13       0A1A           BEQL    40$                       ; Yes
  0000'8F 50   B1       0A1C           CMPW    R0,#SS$_INSFMEM           ; Is it insufficient memory?
          03   13       0A21           BEQL    30$                       ; Yes
        FEED   31       0A23           BRW     RESPOND_NOTQED            ; No, send error back
       53 8ED0          0A26   30$:    POPL    R3                        ; Restore CSID
       55 8ED0          0A29           POPL    R5                        ; Restore CDRP address
      50  55   D0       0A2C           MOVL    R5,R0
 00000000'GF   16       0A2F           JSB     G^EXE$DEANONPAGED         ; Deallocate CDRP
   0FC6 8F     BA       0A35           POPR    #^M<R1,R2,R6,R7,R8,R9,R10,R11>
      53  51   D0       0A39           MOVL    R1,R3                     ; Move CSB address
        F5C1'  30       0A3C           BSBW    CNX$RCV_REJECT            ; Reject this message
               05       0A3F           RSB
                        0A40
                        0A40   40$:    ; Unable to rebuild a lock during failover.  Bugcheck
                        0A40           ; with a (hopefully) useful bugcheck message.
                        0A40
  0000'8F 50   B1       0A40           CMPW    R0,#SS$_INSFMEM           ; Is it insufficient non-paged pool?
          04   12       0A45           BNEQ    50$                       ; No
                        0A47           BUG_CHECK       INSFPOOL,FATAL    ; Yes
  0000'8F 50   B1       0A4B   50$:    CMPW    R0,#SS$_NOLOCKID          ; Is it no lockids?
          04   12       0A50           BNEQ    60$                       ; No
                        0A52           BUG_CHECK       INSFLOCKID,FATAL; Yes
                        0A56   60$:    BUG_CHECK       LOCKMGRERR,FATAL; Unknown error (status in R0)
                        0A5A
```

```
                        0A5A  ;*************************************************************************
                        0A5A  ;
                        0A5A  ;                 Message Build Routines
                        0A5A  ;
                        0A5A  ;*************************************************************************
                        0A5A
                        0A5A          ; BLD_RSPMSG is a generic message building routine that simply
                        0A5A          ; copies a fixed quadword from the CDRP into the message
                        0A5A          ; buffer.  Since this is a response message, the RSPID sent
                        0A5A          ; to us is returned.  Registers contain:
                        0A5A          ;       R2      Address of message buffer
                        0A5A          ;       R5      Address of CDRP
                        0A5A
                        0A5A          ASSUME  LKMSG$B_STATE  EQ  LKMSG$L_CSID+6
                        0A5A
                        0A5A  BLD_RSPMSG:
                        0A5A          ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
 08 A2   0182 8F   B0   0A5A          MOVW    #LKMSG$K_NEWLOCK@8-         ; Store function and facility codes
                        0A60                  !CLSMSG$M_RESPMSG-
                        0A60                  !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
      2C A5        7D   0A60          MOVQ    CDRP$L_VAL1(R5),-           ; Copy value into message buffer
      18 A2             0A63                  LKMSG$L_CSID(R2)
      34 A5        B0   0A65          MOVW    CDRP$L_VAL3(R5),-           ; Store membership seq. num.
      0C A2             0A68                  LKMSG$W_MEMSEQ(R2)
                   05   0A6A          RSB
                        0A6B
                        0A6B
                        0A6B          ; BLD_LOCKRSP builds a response message for lock requests.
                        0A6B          ; This response message is only used if this system actually
                        0A6B          ; created (and queued) a LKB.  In all other cases, (e.g.
                        0A6B          ; directory lookups) a simple response is built by BLD_RSPMSG.
                        0A6B          ; Registers contain:
                        0A6B          ;       R2      Address of message buffer
                        0A6B          ;       R3      Address of CSB
                        0A6B          ;       R5      Address of CDRP
                        0A6B          ;       CDRP$L_VAL1 contains address of the LKB
                        0A6B          ; All registers except R0 and R1 must be preserved
                        0A6B          ;
                        0A6B          ; Note:  This message build routine uses pointers (to LKB and RSB)
                        0A6B          ; that will not be valid if we started a state change.  Therefore,
                        0A6B          ; it is necessary to validate the membership sequence number
                        0A6B          ; before using any pointers stored in the CDRP.
                        0A6B
                        0A6B  BLD_LOCKRSP:
                        0A6B          ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
 08 A2   0182 8F   B0   0A6B          MOVW    #LKMSG$K_NEWLOCK@8-         ; Store function and facility codes
                        0A71                  !CLSMSG$M_RESPMSG-
                        0A71                  !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
      34 A5        B0   0A71          MOVW    CDRP$L_VAL3(R5),-           ; Store membership seq. num.
      0C A2             0A74                  LKMSG$W_MEMSEQ(R2)
   51 64 A3        D0   0A76          MOVL    CSB$L_CLUB(R3),R1           ; Get address of CLUB
    00AC C1        B1   0A7A          CMPW    CLUB$W_MEMSEQ(R1),-         ; Have we done a state change?
      34 A5             0A7E                  CDRP$L_VAL3(R5)
         3C        12   0A80          BNEQ    20$                         ; Yes
   50 2C A5        D0   0A82          MOVL    CDRP$L_VAL1(R5),R0          ; Get address of LKB
      30 A0        D0   0A86          MOVL    LKB$L_CKID(R0),-            ; Store master lockid
      10 A2             0A89                  LKMSG$L_MSTLKID(R2)
```

```
      54 A0  D0  0A8B             MOVL    LKB$L_REMLKID(R0),-      ; Store process lockid
      14 A2      0A8E                     LKMSG$L_PRCLKID(R2)
      32 A5  90  0A90             MOVB    CDRP$L_VAL2+2(R5),-      ; Store response state
      1E A2      0A93                     LKMSG$B_STATE(R2)
      2A A0  B0  0A95             MOVW    LKB$W_STATUS(R0),-       ; Store LKB status
      18 A2      0A98                     LKMSG$W_LKBSTATUS(R2)
                 0A9A             ASSUME  LKB$K_GRANTED  GT  0
                 0A9A             ASSUME  LKB$K_WAITING  LT  0
      36 A0  95  0A9A             TSTB    LKB$B_STATE(R0)          ; Is lock granted?
         19  15  0A9D             BLEQ    10$                      ; No
51    50 A0  D0  0A9F             MOVL    LKB$L_RSB(R0),R1         ; Get RSB address
      28 A1  7D  0AA3             MOVQ    RSB$Q_VALBLK(R1),-       ; Store value block
      20 A2      0AA6                     LKMSG$Q_VALBLK(R2)
      30 A1  7D  0AA8             MOVQ    RSB$Q_VALBLK+8(R1),-
      28 A2      0AAB                     LKMSG$Q_VALBLK+8(R2)
      3C A1  D0  0AAD             MOVL    RSB$L_VALSEQNUM(R1),-    ; Store sequence number
      30 A2      0AB0                     LKMSG$L_VALSEQNUM(R2)
      0E A1  B0  0AB2             MOVW    RSB$W_STATUS(R1),-       ; Store RSB status
      1A A2      0AB5                     LKMSG$W_RSBSTATUS(R2)
         05      0AB7             RSB
                 0AB8
      10 A0  B0  0AB8       10$:  MOVW    LKB$W_RQSEQNM(R0),-      ; Store request sequence number
      30 A2      0ABB                     LKMSG$W_RQSEQNM(R2)
         05      0ABD             RSB
                 0ABE
                 0ABE       20$:  ; We've done a membership state change between our SEND_MSG
                 0ABE             ; and calling the message build routine.  Just send back a RETRY
                 0ABE             ; status.
                 0ABE
      FE 8F  90  0ABE             MOVB    #LKB$K_RETRY,-           ; Store RETRY status
      1E A2      0AC1                     LKMSG$B_STATE(R2)
         05      0AC3             RSB
```

```
                                  0AC4           .SBTTL  BUILD_LKB - Build lock block
                                  0AC4
                                  0AC4  ;++
                                  0AC4  ; FUNCTIONAL DESCRIPTION:
                                  0AC4  ;
                                  0AC4  ;        This routine builds LKBs for received lock requests that are
                                  0AC4  ;        being handled by this system.
                                  0AC4  ;
                                  0AC4  ; CALLING SEQUENCE:
                                  0AC4  ;
                                  0AC4  ;        BSBW    BUILD_LKB
                                  0AC4  ;
                                  0AC4  ; INPUT PARAMETERS:
                                  0AC4  ;
                                  0AC4  ;        R6      Address of parent LKB (or 0)
                                  0AC4  ;        R8      Address of RSB
                                  0AC4  ;        R9      Address of input message
                                  0AC4  ;
                                  0AC4  ; IMPLICIT INPUTS:
                                  0AC4  ;
                                  0AC4  ;        Various fields in the input message are used
                                  0AC4  ;
                                  0AC4  ; OUTPUT PARAMETERS:
                                  0AC4  ;
                                  0AC4  ;        R0      Completion code
                                  0AC4  ;        R6      Address of LKB
                                  0AC4  ;
                                  0AC4  ; COMPLETION CODES:
                                  0AC4  ;
                                  0AC4  ;        SS$_NORMAL      Successful completion
                                  0AC4  ;        SS$_INSFMEM     Insufficent memory to allocate LKB
                                  0AC4  ;        SS$_NOLOCKID    No lock ids
                                  0AC4  ;
                                  0AC4  ; SIDE EFFECTS:
                                  0AC4  ;
                                  0AC4  ;        R0 - R4 are destroyed
                                  0AC4  ;--
                                  0AC4
                                  0AC4
                                  0AC4  NO_LOCKIDS:
                                  0AC4           ; No lock ids.  Deallocate LKB and try to extend table.
                                  0AC4
               50    52      D0   0AC4           MOVL    R2,R0                   ; Address of LKB
          00000000'GF        16   0AC7           JSB     G^EXE$DEANONPAGED       ; Deallocate it
          00000000'GF        16   0ACD           JSB     G^LCK$EXTEND_IDTBL      ; Extend table
               01    50      E8   0AD3           BLBS    R0,BUILD_LKB            ; Success, repeat routine
                                  0AD6  BUILD_LKB_ERR:
                             05   0AD6           RSB
                                  0AD7
                                  0AD7  BUILD_LKB:
          51    0060 8F      3C   0AD7           MOVZWL  #LKB$K_LENGTH,R1        ; Size of LKB
          00000000'GF        16   0ADC           JSB     G^EXE$ALONONPAGED       ; Allocate it
               F1    50      E9   0AE2           BLBC    R0,BUILD_LKB_ERR                ; Unable to allocate it
 08 A2    00350060 8F        D0   0AE5           MOVL    #<DYN$C_LKB@T6>!-       ; Store size and type
                                  0AED                   LKB$K_LENGTH,LKB$W_SIZE(R2)
               0C A2         D4   0AED           CLRL    LKB$L_PID(R2)           ; Clear PID
    50    00000000'GF        D0   0AF0           MOVL    G^LCK$GL_NXTID,R0       ; Get next lock id
```

```
              CB   13  0AF7            BEQL    NO_LOCKIDS               ; No more - return error
51   00000000'GF   D0  0AF9            MOVL    G^LCK$GL_IDTBL,R1        ; Get address of lock id table.  *** May
                       0B00                                             ; combine with next instr. if no loading
        30 A2 50   B0  0B00            MOVW    R0,LKB$L_LKID(R2)        ; Store lockid index
         51 6140   DE  0B04            MOVAL   (R1)[R0],R1              ; Get address of lockid table entry
   00000000'GF 61  3C  0B08            MOVZWL  (R1),G^LCK$GL_NXTID      ; Update ptr to next free id
     32 A2 02 A1   B0  0B0F            MOVW    2(R1),LKB$L_LKID+2(R2)   ; Store lockid sequence number
           61 52   D0  0B14            MOVL    R2,(R1)                  ; Store LKB address in table entry
        50 A2 58   D0  0B17            MOVL    R8,LKB$L_RSB(R2)         ; Make LKB point to RSB
        48 A2 56   D0  0B1B            MOVL    R6,LKB$L_PARENT(R2)      ; Point to parent LKB
              03   13  0B1F            BEQL    10$                      ; No parent
        4C A6      B6  0B21            INCW    LKB$W_REFCNT(R6)         ; Incr. parent's reference count
                       0B24
                       0B24    10$:    ; Build portion of LKB that comes from input message
                       0B24
                       0B24            ASSUME  LKMSG$W_FLAGS  EQ  LKMSG$L_PRCLKID+4
                       0B24            ASSUME  LKMSG$B_RQMODE  EQ  LKMSG$W_FLAGS+2
                       0B24            ASSUME  LKMSG$B_GRMODE  EQ  LKMSG$B_RQMODE+1
                       0B24            ASSUME  LKMSG$L_BLKASTFLG  EQ  LKMSG$B_GRMODE+1
                       0B24
           56 52   D0  0B24            MOVL    R2,R6                    ; Use R6 for LKB from now on
        53 14 A9   9E  0B27            MOVAB   LKMSG$L_PRCLKID(R9),R3   ; Point into message
        54 A6 83   D0  0B2B            MOVL    (R3)+,LKB$L_REMLKID(R6)  ; Store process lock id
        28 A6 83   B0  0B2F            MOVW    (R3)+,LKB$W_FLAGS(R6)    ; Store flags
        34 A6 83   B0  0B33            MOVW    (R3)+,LKB$B_RQMODE(R6)   ; Store lock modes
        20 A6 83   D0  0B37            MOVL    (R3)+,LKB$L_BLKASTADR(R6) ; Store blocking AST flag
                       0B3B
                       0B3B            ; Store EPID of owner of lock unless LCK$M_CVTSYS is set in
                       0B3B            ; which case, store 0.  Also store deadlock priority if not system
                       0B3B            ; owned.
                       0B3B
              50   D4  0B3B            CLRL    R0                       ; Assume system owned
              06   E0  0B3D            BBS     #LCK$V_CVTSYS,-          ; Branch if it is system owned
        09 28 A6       0B3F            LKB$W_FLAGS(R6),20$
     50 10 A9      D0  0B42            MOVL    LKMSG$L_EPIDNEW(R9),R0   ; Get EPID
           4C A9   D0  0B46            MOVL    LKMSG$L_DLCKPRI_NEW(R9),-
           24 A6       0B49            LKB$L_DLCKPRI(R6)        ; Store deadlock priority
     14 A6    50   D0  0B4B    20$:    MOVL    R0,LKB$L_EPID(R6)        ; Store EPID or 0
                       0B4F
                       0B4F            ; Finish remainder of LKB
                       0B4F
              10   B0  0B4F            MOVW    #LKB$M_MSTCPY,-          ; Store master copy flag
           2A A6       0B51            LKB$W_STATUS(R6)
              08   E1  0B53            BBC     #LCK$V_PROTECT,-         ; Branch if this is not a protected lock
        06 28 A6       0B55            LKB$W_FLAGS(R6),30$
         0200 8F   A8  0B58            BISW    #LKB$M_PROTECT,-         ; Set PROTECT status bit
           2A A6       0B5C            LKB$W_STATUS(R6)
        4C A6      B4  0B5E    30$:    CLRW    LKB$W_REFCNT(R6)         ; Clear ref. count
        50    00'  D0  0B61            MOVL    S^#SS$_NORMAL,R0         ; Store success status
                   05  0B64            RSB
```

```
                         0B65             .SBTTL  LCK$DEALLOC_LKB - Deallocate a LKB
                         0B65
                         0B65     ;++
                         0B65     ; FUNCTIONAL DESCRIPTION:
                         0B65     ;
                         0B65     ;       This routine is used to deallocate a LKB and perform
                         0B65     ;       additional cleanup such as deallocating the lock id. and
                         0B65     ;       adjusting the parent's reference count.
                         0B65     ;
                         0B65     ; CALLING SEQUENCE:
                         0B65     ;
                         0B65     ;       BSBW    LCK$DEALLOC_LKB
                         0B65     ;
                         0B65     ; INPUT PARAMETERS:
                         0B65     ;
                         0B65     ;       R6      Address of LKB
                         0B65     ;
                         0B65     ; OUTPUT PARAMETERS:
                         0B65     ;
                         0B65     ;       None
                         0B65     ;
                         0B65     ; SIDE EFFECTS:
                         0B65     ;
                         0B65     ;       R0 - R3 destroyed
                         0B65     ;--
                         0B65
                         0B65     LCK$DEALLOC_LKB::
                         0B65             ; Remove lock from timeout queue, if it's on it.
                         0B65
              06    E5   0B65             BBCC    #LKB$V_TIMOUTQ,-              ; Is lock on timeout queue?
        03 2A A6         0B67                     LKB$W_STATUS(R6),5$
           50   66  0F   0B6A             REMQUE  LKB$L_ASTQFL(R6),R0           ; Yes, remove it
                         0B6D
                         0B6D     5$:     ; Deallocate lock id.
                         0B6D
        50   30 A6  3C   0B6D             MOVZWL  LKB$L_LKID(R6),R0             ; Get lock id index
   51    00000000'GF D0  0B71             MOVL    G^LCK$GL_IDTBL,R1             ; *** Combine with next instr.
           51   6140 DE  0B78             MOVAL   (R1)[R0],R1                   ; Point to table entry
   61    00000000'GF B0  0B7C             MOVW    G^LCK$GL_NXTID,(R1)           ; Store next id in this id's slot
02 A1    32 A6  01  A1   0B83             ADDW3   #1,LKB$L_LKID+2(R6),2(R1)     ; Incr. and store sequence number
              04    1C   0B89             BVC     20$                           ; Didn't overflow to a system address
        02 A1  01   B0   0B8B             MOVW    #1,2(R1)                      ; Overflowed - restart seq. number at 1
00000000'GF    50   D0   0B8F     20$:    MOVL    R0,G^LCK$GL_NXTID             ; This id becomes the next one
                         0B96
                         0B96             ; Decrement parent LKB's reference count
                         0B96
        50   48 A6  D0   0B96             MOVL    LKB$L_PARENT(R6),R0           ; Get address of parent
              05    13   0B9A             BEQL    30$                           ; No parent
           4C A0    B7   0B9C             DECW    LKB$W_REFCNT(R0)              ; Decr. parent's reference count
              0A    19   0B9F             BLSS    80$                           ; Went negative
                         0BA1
                         0BA1     30$:    ; Deallocate LKB
                         0BA1
        50   56     D0   0BA1             MOVL    R6,R0
   00000000'GF      16   0BA4             JSB     G^EXE$DEANONPAGED             ; Deallocate LKB
                    05   0BAA             RSB
                         0BAB
```

```
0BAB  2380 80$:    BUG_CHECK       LKBREFNEG,FATAL
```

```
                        0BAF            .SBTTL  LCK$SND_GRANTED - Send a lock granted message
                        0BAF
                        0BAF    ;++
                        0BAF    ; FUNCTIONAL DESCRIPTION:
                        0BAF    ;
                        0BAF    ;       This routine sends a lock granted message to the process system.
                        0BAF    ;       We are the master system for this lock.
                        0BAF    ;
                        0BAF    ; CALLING SEQUENCE:
                        0BAF    ;
                        0BAF    ;       JSB     LCK$SND_GRANTED
                        0BAF    ;       IPL must be at IPL$_SYNCH
                        0BAF    ;
                        0BAF    ; INPUT PARAMETERS:
                        0BAF    ;
                        0BAF    ;       R6      Address of LKB
                        0BAF    ;       R8      Address of RSB
                        0BAF    ;
                        0BAF    ; OUTPUT PARAMETERS:
                        0BAF    ;
                        0BAF    ;       None
                        0BAF    ;
                        0BAF    ; SIDE EFFECTS:
                        0BAF    ;
                        0BAF    ;       R0 - R5 are not preserved.
                        0BAF    ;--
                        0BAF
                        0BAF    LCK$SND_GRANTED::
          06      E5    0BAF            BBCC    #LKB$V_TIMOUTQ,-                ; Remove lock from timeout queue
    03 2A A6            0BB1                    LKB$W_STATUS(R6),10$            ; if it was on it
    50    66      0F    0BB4            REMQUE  LKB$L_ASTQFL(R6),R0
          02      AA    0BB7    10$:    BICW    #LKB$M_DBLKAST,-                ; Clear deliver blocking AST bit
       2A A6            0BB9                    LKB$W_STATUS(R6)                ; that was set spuriously
 53    58 A6      D0    0BBB            MOVL    LKB$L_CSID(R6),R3               ; Get CSID of remote system
       F43E'      30    0BBF            BSBW    CNX$ALLOC_CDRP                  ; Alloc. a CDRP and convert CSID to CSB
       25 50      E9    0BC2            BLBC    R0,80$                          ; Unable to allocate one or CSID error
                        0BC5
                        0BC5            ; Put lock ids. into CDRP and then send message.
                        0BC5            ; Note:  It might seem preferable to put the LKB address into
                        0BC5            ; the CDRP rather than the lock ids.  The problem with this approach
                        0BC5            ; is that the LKB could be deallocated while the thread to
                        0BC5            ; build and send the message was waiting for resources.
                        0BC5            ; So instead, we use the lock ids. and re-validate the LKB.
                        0BC5            ; If we find that the LKB has been deallocated, we no longer
                        0BC5            ; need to send a message.  However, there is no way to stop
                        0BC5            ; sending a message once we are inside the message build routine.
                        0BC5            ; So instead, we send the message and depend on a lockid mismatch
                        0BC5            ; at the remote end.
                        0BC5
       30 A6      D0    0BC5            MOVL    LKB$L_LKID(R6),-                ; Store master lockid
       2C A5            0BC8                    CDRP$L_VAL1(R5)
       54 A6      D0    0BCA            MOVL    LKB$L_REMLKID(R6),-             ; Store process lockid
       30 A5            0BCD                    CDRP$L_VAL2(R5)
 50    64 A3      D0    0BCF            MOVL    CSB$L_CLUB(R3),R0               ; Get address of CLUB
    00AC C0       B0    0BD3            MOVW    CLUB$W_MEMSEQ(R0),-             ; Store membership seq. num.
       34 A5            0BD7                    CDRP$L_VAL3(R5)
                        0BD9
```

```
                       0BD9          ; Store address of message build routine and send message
                       0BD9
        0E'AF     9E   0BD9          MOVAB   B^BLD_GRANTEDMSG,-          ; Store address of message
        4C A5          0BDC                  CDRP$L_MSGBLD(R5)           ; build routine
        F41F'     30   0BDE          BSBW    CNX$SEND_MSG_CSB            ; Send the message
     50   55      D0   0BE1          MOVL    R5,R0                       ; Address of CDRP
  00000000'GF     17   0BE4          JMP     G^EXE$DEANONPAGED           ; Deallocate it
                       0BEA
     54   56      D0   0BEA  80$:    MOVL    R6,R4                       ; Move LKB address
                       0BED
                       0BED  INSERT_ON_TIMOUTQ:
                       0BED          ; CSID conversion error or no CDRPs.  Queue LKB to timeout queue
                       0BED          ; if insufficient memory.  LKB address is in R4.
                       0BED
  0000'8F 50      B1   0BED          CMPW    R0,#SS$_INSFMEM             ; Is it insufficient memory?
          16      12   0BF2          BNEQ    90$                         ; No
     0040 8F      A8   0BF4          BISW    #LKB$M_TIMOUTQ,-            ; Set timeout bit
        2A A4          0BF8                  LKB$W_STATUS(R4)
  00000000'GF     D0   0BFA          MOVL    G^EXE$GL_ABSTIM,-           ; Store immediate timeout time
        18 A4          0C00                  LKB$L_DUETIME(R4)
           64     0E   0C02          INSQUE  LKB$L_ASTQFL(R4),-          ; Insert at head of timeout queue
  00000000'GF          0C04                  G^LCK$GL_TIMOUTQ
                  05   0C09          RSB
                       0C0A
                       0C0A  90$:    BUG_CHECK       LOCKMGRERR,FATAL
                       0C0E
                       0C0E          ; Action routine to build the lock granted message.  Inputs are:
                       0C0E          ;       R2      Address of message buffer
                       0C0E          ;       R5      Address of CDRP
                       0C0E
                       0C0E  BLD_GRANTEDMSG:
           54     DD   0C0E          PUSHL   R4                          ; Save registers
           56     DD   0C10          PUSHL   R6
                       0C12          ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
08 A2  0202 8F    B0   0C12          MOVW    #LKMSG$K_GRANTED@8-         ; Store function and facility codes
                       0C18                  !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
        34 A5     B0   0C18          MOVW    CDRP$L_VAL3(R5),-           ; Store membership seq. num.
        0C A2          0C1B                  LKMSG$W_MEMSEQ(R2)
     51 2C A5     D0   0C1D          MOVL    CDRP$L_VAL1(R5),R1          ; Get master lockid
     50 30 A5     D0   0C21          MOVL    CDRP$L_VAL2(R5),R0          ; Get process lockid
        2C A5     7D   0C25          MOVQ    CDRP$L_VAL1(R5),-           ; Store lockids
        10 A2          0C28                  LKMSG$L_MSTLKID(R2)
         03AA     30   0C2A          BSBW    VERIFYREMLKID
        22 50     E9   0C2D          BLBC    R0,60$                      ; Branch if LKB went away
     50 50 A6     D0   0C30          MOVL    LKB$L_RSB(R6),R0            ; Get RSB address
                       0C34
                       0C34          ; Note we are only sending the low byte of the RSB status
                       0C34          ; as the high byte overlaps the LKMSG$B_GRMODE field.
                       0C34          ; We can get away with this because we only need to send
                       0C34          ; the RSB$V_VALINVLD bit.  However, this should be fixed.
                       0C34
                       0C34          ASSUME  LKMSG$B_GRMODE  EQ  LKMSG$W_RSBSTATUS+1
                       0C34          ASSUME  RSB$V_VALINVLD  LE  7
                       0C34
        2A A6     B0   0C34          MOVW    LKB$W_STATUS(R6),-          ; Store LKB status
        18 A2          0C37                  LKMSG$W_LKBSTATUS(R2)
        0E A0     90   0C39          MOVB    RSB$W_STATUS(R0),-          ; Store low byte of RSB status
```

```
      1A A2        0C3C  2496                   LKMSG$W_RSBSTATUS(R2)
      35 A6   90   0C3E  2497          MOVB     LKB$B_GRMODE(R6),-            ; Store granted mode
      1B A2        0C41  2498                   LKMSG$B_GRMODE(R2)
      28 A0   7D   0C43  2499          MOVQ     RSB$Q_VALBLK(R0),-            ; Store value block
      20 A2        0C46  2500                   LKMSG$Q_VALBLK(R2)
      30 A0   7D   0C48  2501          MOVQ     RSB$Q_VALBLK+8(R0),-
      28 A2        0C4B  2502                   LKMSG$Q_VALBLK+8(R2)
      3C A0   D0   0C4D  2503          MOVL     RSB$L_VALSEQNUM(R0),-         ; Store value block sequence number
      30 A2        0C50  2504                   LKMSG$L_VALSEQNUM(R2)
         56 8ED0   0C52  2505 60$:     POPL     R6                            ; Restore registers
         54 8ED0   0C55  2506          POPL     R4
              05   0C58  2507          RSB
```

```
                     0C59          .SBTTL  LCK$RCV_GRANTED - Receive lock granted message
                     0C59
                     0C59  ;++
                     0C59  ; FUNCTIONAL DESCRIPTION:
                     0C59  ;
                     0C59  ;       This routine handles the receiving of a lock granted message
                     0C59  ;       sent by the master system.  This system is the process system for
                     0C59  ;       this lock.
                     0C59  ;
                     0C59  ; CALLING SEQUENCE:
                     0C59  ;
                     0C59  ;       JSB     LCK$RCV_GRANTED (called by SCS received message routine)
                     0C59  ;       IPL must be at IPL$_SYNCH
                     0C59  ;
                     0C59  ; INPUT PARAMETERS:
                     0C59  ;
                     0C59  ;       R2      Address of message buffer
                     0C59  ;       R3      Address of CSB
                     0C59  ;
                     0C59  ; OUTPUT PARAMETERS:
                     0C59  ;
                     0C59  ;       None
                     0C59  ;
                     0C59  ; SIDE EFFECTS:
                     0C59  ;
                     0C59  ;       R0 - R5 Destroyed
                     0C59  ;--
                     0C59
                     0C59  LCK$RCV_GRANTED::
      0FCC 8F   BB   0C59          PUSHR   #^M<R2,R3,R6,R7,R8,R9,R10,R11>
                     0C5D          ASSUME  LKMSG$L_PRCLKID  EQ  LKMSG$L_MSTLKID+4
 50     10 A2   7D   0C5D          MOVQ    LKMSG$L_MSTLKID(R2),R0   ; Get process and master lockids
        0373    30   0C61          BSBW    VERIFYREMLKID            ; Convert to LKB address
        63 50   E9   0C64          BLBC    R0,40$                   ; Invalid lockid - ignore message
 58     50 A6   D0   0C67          MOVL    LKB$L_RSB(R6),R8         ; Get RSB address
                     0C6B
                     0C6B          ; Store value block unless the one we already have is later.
                     0C6B          ; Note we are only sending the low byte of the RSB status
                     0C6B          ; as the high byte overlaps the LKMSG$B_GRMODE field.
                     0C6B          ; We can get away with this because we only need to send
                     0C6B          ; the RSB$V_VALINVLD bit.  However, this should be fixed.
                     0C6B
                     0C6B          ASSUME  RSB$V_VALINVLD  LE  7
                     0C6B
        3C A8   C3   0C6B          SUBL3   RSB$L_VALSEQNUM(R8),-    ; Compare sequence numbers
 50     30 A2        0C6E                  LKMSG$L_VALSEQNUM(R2),R0
           1C   15   0C71          BLEQ    10$                      ; We already have a newer one
        20 A2   7D   0C73          MOVQ    LKMSG$Q_VALBLK(R2),-     ; Store value block
        28 A8        0C76                  RSB$Q_VALBLK(R8)
        28 A2   7D   0C78          MOVQ    LKMSG$Q_VALBLK+8(R2),-
        30 A8        0C7B                  RSB$Q_VALBLK+8(R8)
        30 A2   D0   0C7D          MOVL    LKMSG$L_VALSEQNUM(R2),-  ; Store sequence number
        3C A8        0C80                  RSB$L_VALSEQNUM(R8)
           02   AA   0C82          BICW    #RSB$M_VALINVLD,-        ; Clear value block invalid flag
        0E A8        0C84                  RSB$W_STATUS(R8)
           01   E1   0C86          BBC     #RSB$V_VALINVLD,-        ; and conditionally set it
     04 1A A2        0C88                  LKMSG$Q_RSBSTATUS(R2),10$
```

```
      02  A8  0C8B           BISW     #RSB$M_VALINVLD,-
   0E A8      0C8D                    RSB$W_STATUS(R8)
              0C8F
              0C8F 10$:      ; There are a number of race conditions that must be handled
              0C8F           ; correctly when this message is received.  These are:
              0C8F           ;          1)  We receive this message before the process has
              0C8F           ;              been scheduled.  The LKB state is RSPQUEUED.  In this
              0C8F           ;              case change the state to RSPGRANTD.
              0C8F           ;          2)  The lock was dequeued before this message arrived.
              0C8F           ;              In this case we will get an invalid lock id above
              0C8F           ;              and will ignore this message.
              0C8F           ;          3)  The lock was a conversion and it was canceled before
              0C8F           ;              this message arrived.  The LKB state is GRANTED.  Ignore
              0C8F           ;              this message unless the BLKASTQED bit is set.  In this
              0C8F           ;              case handle this message as if it were a blocking AST
              0C8F           ;              message.
              0C8F           ;          4)  Case # 3 but another conversion has already been issued.
              0C8F           ;              Lock state is SCSWAIT, RETRY, RSPQUEUED, or RSPGRANTD.
              0C8F           ;              This condition is also determined by the granted
              0C8F           ;              lock mode in the message not being equal to the requested
              0C8F           ;              mode in the LKB.  Ignore this message.
              0C8F
              0C8F           DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
              0C8F                    <-
              0C8F                    <GRANTED,20$>,-           ; Ignore unless BLKASTQED set
              0C8F                    <CONVERT,50$>,-           ; Handle normally
              0C8F                    <WAITING,50$>,-           ; Handle normally
              0C8F                    <RETRY,80$>,-             ; Ignore message
              0C8F                    <SCSWAIT,80$>,-           ; Ignore message
              0C8F                    <RSPQUEUED,30$>,-         ; Change state to RSPGRANTD
              0C8F                    <RSPGRANTD,80$>,-         ; Ignore message
              0C8F                    >
              0CA5           BUG_CHECK       LOCKMGRERR,FATAL; Invalid state
              0CA9
              0CA9 20$:      ; The lock state is granted.  This should only occur if the lock
              0CA9           ; request was canceled.  If the BLKASTQED bit is set then treat this
              0CA9           ; message as a blocking AST message.
              0CA9
      03  E1  0CA9           BBC      #LKB$V_BLKASTQED,-        ; Branch if should not queue blocking
52 18 A2      0CAB                    LKMSG$W_LKBSTATUS(R2),80$ ; AST
 0FCC 8F  BA  0CAE           POPR     #^M<R2,R3,R6,R7,R8,R9,R10,R11> ; Restore registers
    00B8  31  0CB2           BRW      LCK$RCV_BLKING            ; Use common code
              0CB5
              0CB5 30$:      ; The lock state is RSPQUEUED.  Change to RSPGRANTD and store
              0CB5           ; BLKASTQED bit.
              0CB5
   1B A2  91  0CB5           CMPB     LKMSG$B_GRMODE(R2),-      ; Is granted mode (in message) equal
   34 A6      0CB8                    LKB$B_RQMODE(R6)          ; to requested mode (in lock)?
      44  12  0CBA           BNEQ     80$                       ; No, ignore message
   FA 8F  90  0CBC           MOVB     #LKB$K_RSPGRANTD,-        ; Yes, store new state
   36 A6      0CBF                    LKB$B_STATE(R6)
      03  E1  0CC1           BBC      #LKB$V_BLKASTQED,-        ; Branch if should not queue blocking
3A 18 A2      0CC3                    LKMSG$W_LKBSTATUS(R2),80$ ; AST
      08  A8  0CC6           BISW     #LKB$M_BLKASTQED,-        ; Set BLKASTQED bit
   2A A6      0CC8                    LKB$W_STATUS(R6)
      34  11  0CCA 40$:      BRB      80$
              0CCC
```

```
                     0CCC  2623 50$:    ; The lock state is CONVERT or WAITING.  Now grant the lock.
                     0CCC  2624
       1B A2    91   0CCC  2625         CMPB    LKMSG$B_GRMODE(R2),-        ; Is granted mode (in message) equal
       34 A6         0CCF  2626                 LKB$B_RQMODE(R6)            ; to requested mode (in lock)?
          2D    12   0CD1  2627         BNEQ    80$                         ; No, ignore message
       34 A6    90   0CD3  2628         MOVB    LKB$B_RQMODE(R6),-          ; Yes, make granted mode equal
       35 A6         0CD6  2629                 LKB$B_GRMODE(R6)            ; requested mode
          00'   3C   0CD8  2630         MOVZWL  S^#SS$_NORMAL,-             ; Store status
       2C A6         0CDA  2631                 LKB$L_LKST1(R6)
  50   38 A6    0F   0CDC  2632         REMQUE  LKB$L_SQFL(R6),R0           ; Remove from wait queue
       20 A6    D5   0CE0  2633         TSTL    LKB$L_BLKASTADR(R6)         ; Blocking AST specified?
          15    13   0CE3  2634         BEQL    60$                         ; No
       42 A8    B6   0CE5  2635         INCW    RSB$W_BLKASTCNT(R8)         ; Yes, incr. blocking AST count
          03    E1   0CE8  2636         BBC     #LKB$V_BLKASTQED,-          ; Branch if should not queue blocking
    0D 18 A2         0CEA  2637                 LKMSG$W_LKBSTATUS(R2),60$ ; AST
  2A A6    0A   A8   0CED  2638         BISW    #LKB$M_BLKASTQED!-          ; Set blocking AST queued and
                     0CF1  2639                 LKB$M_DBLKAST,LKB$W_STATUS(R6) ; deliver blocking AST status
          10    88   0CF1  2640         BISB    #LKB$M_PKAST,-              ; Set piggyback kernel AST bit
       0B A6         0CF3  2641                 LKB$B_RMOD(R6)
       20 A6    D0   0CF5  2642         MOVL    LKB$L_BLKASTADR(R6),-       ; Store address of blocking AST routine
       10 A6         0CF8  2643                 LKB$L_AST(R6)
 00000000'GF    16   0CFA  2644 60$:    JSB     G^LCK$GRANT_REM             ; Finish granting it using common code
                     0D00  2645
                     0D00  2646 80$:    ; Deallocate message buffer and exit
                     0D00  2647
     0FCC 8F    BA   0D00  2648         POPR    #^M<R2,R3,R6,R7,R8,R9,R10,R11>
       F2F9'    31   0D04  2649         BRW     CNX$DEALL_MSG_BUF_CSB       ; Deallocate message buffer and return
```

```
                         0D07          .SBTTL  LCK$SND_BLKING  - Send a blocking message
                         0D07
                         0D07  ;++
                         0D07  ; FUNCTIONAL DESCRIPTION:
                         0D07  ;
                         0D07  ;       This routine sends a blocking message to the process system.
                         0D07  ;       We are the master system for this lock.
                         0D07  ;
                         0D07  ; CALLING SEQUENCE:
                         0D07  ;
                         0D07  ;       JSB     LCK$SND_BLKING
                         0D07  ;       IPL must be at IPL$_SYNCH
                         0D07  ;
                         0D07  ; INPUT PARAMETERS:
                         0D07  ;
                         0D07  ;       R5      Address of LKB blocking another lock
                         0D07  ;       R6      Address of LKB being blocked
                         0D07  ;
                         0D07  ; OUTPUT PARAMETERS:
                         0D07  ;
                         0D07  ;       None
                         0D07  ;
                         0D07  ; SIDE EFFECTS:
                         0D07  ;
                         0D07  ;       R0 - R5 are not preserved
                         0D07  ;--
                         0D07
                         0D07          .ENABL  LSB
                         0D07
                         0D07  LCK$SND_BLKING::
           08     A8     0D07          BISW    #LKB$M_BLKASTQED,-      ; Set blocking AST queued bit
        2A A5            0D09                  LKB$W_STATUS(R5)
                         0D0B
                         0D0B          ; If the blocking LKB (in R5) is on the same system as the
                         0D0B          ; LKB being blocked (in R6) then we don't send a message as
                         0D0B          ; this case is handled locally on the other system.
                         0D0B          ; Also note that the comment in LCK$SND_GRANTED about the LKB
                         0D0B          ; being deallocated out from under us applies here also.  However,
                         0D0B          ; in this case, copying the info. is not as bad since much less
                         0D0B          ; is copied.
                         0D0B
  53    58 A5     D0     0D0B          MOVL    LKB$L_CSID(R5),R3       ; Get CSID
           04     E1     0D0F          BBC     #LKB$V_MSTCPY,-         ; Branch if blockee is local
     06 2A A6            0D11                  LKB$W_STATUS(R6),10$
  58 A6    53     D1     0D14          CMPL    R3,LKB$L_CSID(R6)       ; Does CSID match other lock?
           3C     13     0D18          BEQL    30$                     ; Yes
     54    55     D0     0D1A  10$:    MOVL    R5,R4                   ; R4 will point to LKB
                         0D1D
                         0D1D  SEND_BLOCKING:
                         0D1D          ; Allocate a CDRP and copy lockids into it.  R3 contains CSID
                         0D1D          ; and R4 points to LKB.
                         0D1D
        F2E0'     30     0D1D          BSBW    CNX$ALLOC_CDRP          ; Alloc. CDRP and convert CSID to CSB
        03 50     E8     0D20          BLBS    R0,20$
         FEC7     31     0D23          BRW     INSERT_ON_TIMOUTQ       ; None available or CSID error
        30 A4     D0     0D26  20$:    MOVL    LKB$L_[KID(R4),-        ; Copy master lockid
        2C A5            0D29                  CDRP$[_VAL1(R5)
```

```
          54 A4    D0 0D2B              MOVL    LKB$L_REMLKID(R4),-     ; Copy process lockid
          30 A5       0D2E                      CDRP$L_VAL2(R5)
     50   64 A3    D0 0D30              MOVL    CSB$L_CLUB(R3),R0       ; Get address of CLUB
        00AC C0    B0 0D34              MOVW    CLUB$W_MEMSEQ(R0),-     ; Store membership seq. num.
          34 A5       0D38                      CDRP$L_VAL3(R5)
          35 A4    90 0D3A              MOVB    LKB$B_GRMODE(R4),-      ; Copy granted mode of lock
          36 A5       0D3D                      CDRP$L_VAL3+2(R5)
                      0D3F
                      0D3F              ; Store address of message build routine and send message
                      0D3F
            00000002  0D3F              .IF NE  CA$_MEASURE
       00000000'GF D6 0D3F              INCL    G^PMS$GL_BLK_OUT
                      0D45              .ENDC
                      0D45
          57'AF    9E 0D45              MOVAB   B^BLD_BLKINGMSG,-
          4C A5       0D48                      CDRP$L_MSGBLD(R5)
           F2B3'   30 0D4A              BSBW    CNX$SEND_MSG_CSB        ; Send the message
     50   55       D0 0D4D              MOVL    R5,R0                   ; Address of CDRP
       00000000'GF 17 0D50              JMP     G^EXE$DEANONPAGED       ; Deallocate it
                      0D56
                   05 0D56   30$:       RSB
                      0D57
                      0D57              ; Action routine to build the blocking message.  Inputs are:
                      0D57              ;               R2      Address of message buffer
                      0D57              ;               R5      Address of CDRP
                      0D57
                      0D57   BLD_BLKINGMSG:
                      0D57              ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
 08 A2    0502 8F  B0 0D57              MOVW    #LKMSG$K_BLKING@8-      ; Store function and facility codes
                      0D5D                      !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
          2C A5    7D 0D5D              MOVQ    CDRP$L_VAL1(R5),-       ; Store master and process lockids
          10 A2       0D60                      LKMSG$L_MSTLKID(R2)
          34 A5    B0 0D62              MOVW    CDRP$L_VAL3(R5),-       ; Store membership seq. num.
          0C A2       0D65                      LKMSG$W_MEMSEQ(R2)
          36 A5    90 0D67              MOVB    CDRP$L_VAL3+2(R5),-     ; Copy granted mode of lock
          1B A2       0D6A                      LKMSG$B_GRMODE(R2)
                   05 0D6C              RSB
                      0D6D
                      0D6D              .DSABL  LSB
```

```
                         0D6D         .SBTTL  LCK$RCV_BLKING - Receive a blocking message
                         0D6D
                         0D6D  ;++
                         0D6D  ; FUNCTIONAL DESCRIPTION:
                         0D6D  ;
                         0D6D  ;      This routine handles receiving a blocking AST message from
                         0D6D  ;      the master system.  This system is the process system.
                         0D6D  ;
                         0D6D  ; CALLING SEQUENCE:
                         0D6D  ;
                         0D6D  ;      JSB     LCK$RCV_BLKING (called by SCS received message routine)
                         0D6D  ;      IPL must be at IPL$_SYNCH
                         0D6D  ;
                         0D6D  ; INPUT PARAMETERS:
                         0D6D  ;
                         0D6D  ;      R2      Address of message buffer
                         0D6D  ;      R3      Address of CSB
                         0D6D  ;
                         0D6D  ; OUPUT PARAMETERS:
                         0D6D  ;
                         0D6D  ;      None
                         0D6D  ;
                         0D6D  ; SIDE EFFECTS:
                         0D6D  ;
                         0D6D  ;      R0 - R5 destroyed
                         0D6D  ;
                         0D6D  ;--
                         0D6D  LCK$RCV_BLKING::
                         0D6D
            00000002     0D6D         .IF NE  CA$_MEASURE
    00000000'GF     D6   0D6D         INCL    G^PMS$GL_BLK_IN
                         0D73         .ENDC
                         0D73
       004C 8F      BB   0D73         PUSHR   #^M<R2,R3,R6>
                         0D77         ASSUME  LKMSG$L_PRCLKID  EQ  LKMSG$L_MSTLKID+4
    50    10 A2     7D   0D77         MOVQ    LKMSG$L_MSTLKID(R2),R0  ; Get master and process lockids
          0259      30   0D7B         BSBW    VERIFYREMLKID           ; Verify lockid and convert to LKB addr.
         4A 50      E9   0D7E         BLBC    R0,80$                  ; Lock is gone - ignore message
                         0D81
                         0D81         ; Queue a blocking AST unless one has been queued before
                         0D81         ; (in which case we shouldn't have received the message; but
                         0D81         ; there are race conditions that will cause this to happen).
                         0D81         ; we will ignore that detail).  Also, state must be GRANTED
                         0D81         ; at the same lock mode to get a blocking AST.  It may not be
                         0D81         ; if we are in the middle of a conversion, or have canceled a
                         0D81         ; conversion, for example.  Also, we have to handle the case
                         0D81         ; that this message came in before the process has been awakened to see
                         0D81         ; that the request was granted.
                         0D81
                         0D81         DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
                         0D81                 <-
                         0D81                 <GRANTED,50$>,-         ; Handle normally
                         0D81                 <CONVERT,80$>,-         ; Ignore message
                         0D81                 <WAITING,80$>,-         ; Ignore message
                         0D81                 <RETRY,80$>,-           ; Ignore message
                         0D81                 <SCSWAIT,80$>,-         ; Ignore message
                         0D81                 <RSPQUEUED,80$>,-       ; Ignore message
```

```
                         0D81                     <RSPNOTQED,30$>,-       ; Set BLKASTQED bit
                         0D81                     <RSPGRANTD,40$>,-       ; Set BLKASTQED bit
                         0D81                     >
                         0D97         BUG_CHECK        LOCKMGRERR,FATAL; Invalid state
                         0D9B
                         0D9B  30$:   ; Lock state is RSPNOTQED.
                         0D9B
        35 A6      91    0D9B         CMPB    LKB$B_GRMODE(R6),-          ; Verify granted modes are the same
        1B A2            0D9E                 LKMSG$B_GRMODE(R2)
           29      12    0DA0         BNEQ    80$                         ; They're not - ignore message
           07      11    0DA2         BRB     45$
                         0DA4
                         0DA4  40$:   ; Lock state is RSPGRANTD.
                         0DA4
        34 A6      91    0DA4         CMPB    LKB$B_RQMODE(R6),-          ; Verify requested mode (in lock)
        1B A2            0DA7                 LKMSG$B_GRMODE(R2)          ; is the same as granted mode (in msg)
           20      12    0DA9         BNEQ    80$                         ; They're not - ignore message
           08      A8    0DAB  45$:   BISW    #LKB$M_BLKASTQED,-          ; Set BLKASTQED bit
        2A A6            0DAD                 LKB$W_STATUS(R6)
           1A      11    0DAF         BRB     80$
                         0DB1
                         0DB1  50$:   ; Lock state is GRANTED.
                         0DB1
        35 A6      91    0DB1         CMPB    LKB$B_GRMODE(R6),-          ; Verify granted modes are the same
        1B A2            0DB4                 LKMSG$B_GRMODE(R2)
           13      12    0DB6         BNEQ    80$                         ; They're not - ignore message
           03      E0    0DB8         BBS     #LKB$V_BLKASTQED,-          ; Branch if a blocking ast has been
     0E 2A A6            0DBA                 LKB$W_STATUS(R6),80$        ; queued
        20 A6      D5    0DBD         TSTL    LKB$L_BLKASTADR(R6)         ; Verify there is a blocking AST routine
           09      13    0DC0         BEQL    80$
     55    56      D0    0DC2         MOVL    R6,R5                       ; Move LKB address for subroutine call
  00000000'GF      16    0DC5         JSB     G^LCK$QUEUE_BLKAST          ; Queue a blocking AST
                         0DCB
                         0DCB  80$:   ; Deallocate message buffer and exit
                         0DCB
      004C 8F      BA    0DCB         POPR    #^M<R2,R3,R6>
        F22E'      31    0DCF         BRW     CNX$DEALL_MSG_BUF_CSB       ; Deallocate message buffer and return
```

```
                0DD2          .SBTTL  LCK$SND_DEQ - Send dequeue message
                0DD2
                0DD2  ;++
                0DD2  ; FUNCTIONAL DESCRIPTION:
                0DD2  ;
                0DD2  ;       This routine sends a dequeue lock message to a remote system.
                0DD2  ;       This system is the process system and the remote system is the
                0DD2  ;       master system.
                0DD2  ;
                0DD2  ; CALLING SEQUENCE:
                0DD2  ;
                0DD2  ;       JSB     LCK$SND_DEQGR   (Lock is in granted state)
                0DD2  ;       JSB     LCK$SND_DEQCV   (Lock is in conversion wait state)
                0DD2  ;       JSB     LCK$SND_DEQWT   (Lock is in waiting state)
                0DD2  ;       IPL must be at IPL$_SYNCH
                0DD2  ;
                0DD2  ; INPUT PARAMETERS:
                0DD2  ;
                0DD2  ;       R3      Address of CSB (not CSID!)
                0DD2  ;       R4      Dequeue flags
                0DD2  ;       R5      Address of CDRP
                0DD2  ;       R6      Address of LKB
                0DD2  ;       R7      Final completion status to store in LKB$L_LKST1 or 0
                0DD2  ;               which indicates a default status should be used
                0DD2  ;       R8      Address of RSB
                0DD2  ;       R9      Address of value block (LCK$SND_DEQGR entry only)
                0DD2  ;
                0DD2  ; OUTPUT PARAMETERS:
                0DD2  ;
                0DD2  ;       None
                0DD2  ;
                0DD2  ; SIDE EFFECTS:
                0DD2  ;
                0DD2  ;       R0 - R5 destroyed
                0DD2  ;--
                0DD2
                0DD2          .ENABL  LSB
                0DD2
                0DD2  LCK$SND_DEQWT::
                0DD2          ; Lock is waiting
                0DD2
54    06    AA  0DD2          BICW    #LCK$M_CANCEL-          ; Clear cancel
                0DD5                  !LCK$M_INVVALBLK,R4     ; and invalidate valblk flags
      59    D4  0DD5          CLRL    R9                      ; Indicate no value block
      1B    11  0DD7          BRB     40$
                0DD9
                0DD9  LCK$SND_DEQCV::
                0DD9          ; The lock is on the conversion queue.
                0DD9
54    04    AA  0DD9          BICW    #LCK$M_INVVALBLK,R4     ; Clear invalidate valblk flag
35 A6       90  0DDC          MOVB    LKB$B_GRMODE(R6),-      ; Store granted mode in case
37 A5           0DDF                  CDRP$L_VAL3+3(R5)       ; this is a CANCEL function
      59    D4  0DE1          CLRL    R9                      ; Indicate no value block
      0F    11  0DE3          BRB     40$
                0DE5
                0DE5  LCK$SND_DEQGR::
                0DE5          ; Lock is granted
```

```
                    0DE5
           59   D5  0DE5          TSTL    R9                      ; Is there a value block?
           0B   13  0DE7          BEQL    40$                     ; No value block
    38 A5  89   7D  0DE9          MOVQ    (R9)+,CDRP$L_VAL4(R5)   ; Copy value block into CDRP
    40 A5  69   7D  0DED          MOVQ    (R9),CDRP$L_VAL6(R5)
       59  01   D0  0DF1          MOVL    #1,R9                   ; Set low byte non-zero
                    0DF4
                    0DF4  40$:    ; Put additional information for message into CDRP.
                    0DF4
             00000002  0DF4       .IF NE  CA$_MEASURE
       00000000'GF  D6  0DF4      INCL    G^PMS$GL_DEQ_OUT
       00000000'GF  D7  0DFA      DECL    G^PMS$GL_DEQ_LOC        ; Counter was incorrectly incremented
                    0E00          .ENDC
                    0E00
    36 A5  59   90  0E00          MOVB    R9,CDRP$L_VAL3+2(R5)    ; Store value block flag
       54 A6    D0  0E04          MOVL    LKB$L_REMLKID(R6),-     ; Store master lockid
       2C A5        0E07                  CDRP$L_VAL1(R5)
       30 A6    D0  0E09          MOVL    LKB$L_LKID(R6),-        ; Store process lockid
       30 A5        0E0C                  CDRP$L_VAL2(R5)
    34 A5  54   B0  0E0E          MOVW    R4,CDRP$L_VAL3(R5)      ; Store flags
                    0E12
                    0E12          ; Remove LKB from RSB and deallocate RSB, if necessary (don't
                    0E12          ; do this if we are only cancelling the lock).
                    0E12
    10 54  01   E0  0E12          BBS     #LCK$V_CANCEL,R4,50$    ; Branch if cancelling lock
           55   DD  0E16          PUSHL   R5                      ; Save address of CDRP
           53   DD  0E18          PUSHL   R3                      ; Save address of CSB
 50    38 A6    0F  0E1A          REMQUE  LKB$L_SQFL(R6),R0       ; Remove lock from RSB
       00000000'GF  16  0E1E      JSB     G^LCK$CHECK_RSB         ; Check RSB and delete if necessary
           28   BA  0E24          POPR    #^M<R3,R5>              ; Restore registers
                    0E26
                    0E26  50$:    ; Send the message
                    0E26
 50    64 A3    D0  0E26          MOVL    CSB$L_CLUB(R3),R0       ; Get address of CLUB
     00AC C0    B0  0E2A          MOVW    CLUB$W_MEMSEQ(R0),-     ; Store membership seq. num.
       48 A5        0E2E                  CDRP$L_VAL8(R5)
       41'AF    9E  0E30          MOVAB   B^BLD_DEQMSG,-          ; Address of message build routine
       4C A5        0E33                  CDRP$L_MSGBLD(R5)
       F1C8'    30  0E35          BSBW    CNX$SEND_MSG_CSB
                    0E38
                    0E38          ; Deallocate the CDRP.  This is done on success or failure
                    0E38          ; since the lock is gone from this system (or CANCELed back to it's
                    0E38          ; old mode) as far as failover is concerned.
                    0E38
 50    55       D0  0E38          MOVL    R5,R0                   ; Address of CDRP
       00000000'GF  17  0E3B      JMP     G^EXE$DEANONPAGED
                    0E41
                    0E41          .DSABL  LSB
                    0E41
                    0E41  BLD_DEQMSG:
                    0E41          ; Action routine to build the actual dequeue lock message
                    0E41          ; Inputs are:
                    0E41          ;       R2      Address of message buffer
                    0E41          ;       R5      Address of CDRP
                    0E41          ; All registers except R0 and R1 must be preserved
                    0E41
                    0E41          ASSUME  LKMSG$L_PRCLKID  EQ  LKMSG$L_MSTLKID+4
```

```
                              0E41  2957          ASSUME  LKMSG$B_GRMODE  EQ  LKMSG$W_FLAGS+3
                              0E41  2958          ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
                              0E41  2959
   08 A2    0302 8F     B0    0E41  2960          MOVW    #LKMSG$K_DEQ@8-                    ; Store function and facility codes
                              0E47  2961                  !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
              48 A5     B0    0E47  2962          MOVW    CDRP$L_VAL8(R5),-                  ; Store membership seq. num.
              0C A2           0E4A  2963                  LKMSG$W_MEMSEQ(R2)
              2C A5     7D    0E4C  2964          MOVQ    CDRP$L_VAL1(R5),-                  ; Store process and master lockid
              10 A2           0E4F  2965                  LKMSG$L_MSTLKID(R2)
              34 A5     D0    0E51  2966          MOVL    CDRP$L_VAL3(R5),-                  ; Store flags and granted mode
              18 A2           0E54  2967                  LKMSG$W_FLAGS(R2)
              36 A5     9A    0E56  2968          MOVZBL  CDRP$L_VAL3+2(R5),-                ; Store value block flag
              1C A2           0E59  2969                  LKMSG$L_VALBLKFLG(R2)
                 0A     13    0E5B  2970          BEQL    30$                                ; No value block
              38 A5     7D    0E5D  2971          MOVQ    CDRP$L_VAL4(R5),-                  ; Store value block
              20 A2           0E60  2972                  LKMSG$Q_VALBLK(R2)
              40 A5     7D    0E62  2973          MOVQ    CDRP$L_VAL6(R5),-
              28 A2           0E65  2974                  LKMSG$Q_VALBLK+8(R2)
                        05    0E67  2975 30$:     RSB
```

```
                         0E68          .SBTTL  LCK$RCV_DEQ - Receive a dequeue lock message
                         0E68
                         0E68  ;++
                         0E68  ; FUNCTIONAL DESCRIPTION:
                         0E68  ;
                         0E68  ;       This routine dequeues a lock on our system (the master) upon
                         0E68  ;       receiving a message from the process system.
                         0E68  ;
                         0E68  ; CALLING SEQUENCE:
                         0E68  ;
                         0E68  ;       JSB     LCK$RCV_DEQ (called by the SCS received message routine)
                         0E68  ;       IPL must be at IPL$_SYNCH
                         0E68  ;
                         0E68  ; INPUT PARAMETERS:
                         0E68  ;
                         0E68  ;       R2      Address of message buffer
                         0E68  ;       R3      Address of CSB
                         0E68  ;
                         0E68  ; OUTPUT PARAMETERS:
                         0E68  ;
                         0E68  ;       None
                         0E68  ;
                         0E68  ; SIDE EFFECTS:
                         0E68  ;
                         0E68  ;       R0 - R5 destroyed
                         0E68  ;--
                         0E68
                         0E68  LCK$RCV_DEQ::
             00000002    0E68          .IF NE  CA$_MEASURE
    00000000'GF     D6   0E68          INCL    G^PMS$GL_DEQ_IN
                         0E6E          .ENDC
                         0E6E
       0FFC 8F      BB   0E6E          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
  51    10 A2       D0   0E72          MOVL    LKMSG$L_MSTLKID(R2),R1   ; Get master lockid
  50    14 A2       D0   0E76          MOVL    LKMSG$L_PRCLKID(R2),R0   ; Get process lockid (for verify)
          015A      30   0E7A          BSBW    VERIFYREMLKID            ; Convert to LKB address
         62 50      E9   0E7D          BLBC    R0,80$                   ; Invalid lock id
            06      E5   0E80          BBCC    #LKB$V_TIMOUTQ,-         ; Is lock on timeout queue?
      03 2A A6           0E82                  LKB$W_STATUS(R6),5$
       50   66      0F   0E85          REMQUE  LKB$L_ASTQFL(R6),R0      ; Yes, remove it
  59    1C A2       D0   0E88  5$:     MOVL    LKMSG$L_VALBLKFLG(R2),R9; Is there a value block?
            04      13   0E8C          BEQL    10$                      ; No
  59    20 A2       9E   0E8E          MOVAB   LKMSG$Q_VALBLK(R2),R9    ; Yes, set address of value block
  54    18 A2       3C   0E92  10$:    MOVZWL  LKMSG$W_FLAGS(R2),R4     ; Get user flags
    00000000'GF     16   0E96          JSB     G^LCK$DEQLOCK            ; Dequeue the lock
         07 50      E9   0E9C          BLBC    R0,60$                   ; Error
       0FFC 8F      BA   0E9F  20$:    POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
         F15A'      31   0EA3          BRW     CNX$DEALL_MSG_BUF_CSB    ; Deallocate message buffer and return
                         0EA6
                         0EA6  60$:    ; The dequeue failed.  The only failure allowed is SS$_CANCELGRANT
                         0EA6          ; which can occur if the process copy was dequeued while in
                         0EA6          ; the conversion queue while at the same time the master copied
                         0EA6          ; was granted.  In this case we have to regrant the master copy
                         0EA6          ; at the old mode to make it consistent with the process copy.
                         0EA6
  0000'8F     50    B1   0EA6          CMPW    R0,#SS$_CANCELGRANT      ; Is it this race condition?
              39    12   0EAB          BNEQ    90$                      ; No - must be SS$_SUBLOCKS
```

```
       58  50 A6   D0  0EAD         MOVL    LKB$L_RSB(R6),R8            ; Get address of RSB
       50  38 A6   0F  0EB1         REMQUE  LKB$L_SQFL(R6),R0           ; Remove lock from granted queue
           20 A6   D5  0EB5         TSTL    LKB$L_BLKASTADR(R6)         ; Is there an old blocking AST flag?
              03   13  0EB8         BEQL    65$                         ; No
           42 A8   B7  0EBA         DECW    RSB$W_BLKASTCNT(R8)         ; Yes, adjust count
           1C A2   D0  0EBD 65$:    MOVL    LKMSG$L_BLKASTFLG(R2),-     ; Store new blocking AST flag
           20 A6       0EC0                 LKB$L_BLKASTADR(R6)
       51  1B A2   9A  0EC2         MOVZBL  LKMSG$B_GRMODE(R2),R1       ; Get old granted mode
     00000000'GF   16  0EC6         JSB     G^LCK$COMP_GGMODE           ; Compute a new group grant mode in R5
       0C A8   55  90  0ECC         MOVB    R5,RSB$B_GGMODE(R8)         ; Store it
       0D A8   55  90  0ED0         MOVB    R5,RSB$B_CGMODE(R8)
     00000000'GF   16  0ED4         JSB     G^LCK$GRANT_LOCK            ; Grant the lock
     00000000'GF   16  0EDA         JSB     G^LCK$GRANTCVTS             ; Grant any waiters
              BD   11  0EE0         BRB     20$
                       0EE2
                       0EE2
                       0EE2 80$:    BUG_CHECK       INVLOCKID,FATAL     ; Invalid lock id
                       0EE6
                       0EE6 90$:    BUG_CHECK       LOCKMGRERR,FATAL;   Dequeue failed
```

```
                                      0EEA          .SBTTL  LCK$SND_RMVDIR - Send remove directory entry message
                                      0EEA
                                      0EEA  ;++
                                      0EEA  ; FUNCTIONAL DESCRIPTION:
                                      0EEA  ;
                                      0EEA  ;       This routine sends a message to the directory system specifying
                                      0EEA  ;       that a root resource directory entry be removed.  This routine
                                      0EEA  ;       is called when the last lock is dequeued from a root resource that
                                      0EEA  ;       is being managed by this system and this system is not the
                                      0EEA  ;       directory system for this resource (RSB$M_DIRENTRY is clear)
                                      0EEA  ;
                                      0EEA  ; CALLING SEQUENCE:
                                      0EEA  ;
                                      0EEA  ;       JSB     LCK$SND_RMVDIR
                                      0EEA  ;       IPL must be at IPL$_SYNCH
                                      0EEA  ;
                                      0EEA  ; INPUT PARAMETERS:
                                      0EEA  ;
                                      0EEA  ;       R8      Address of RSB
                                      0EEA  ;
                                      0EEA  ; OUTPUT PARAMETERS:
                                      0EEA  ;
                                      0EEA  ;       None
                                      0EEA  ;
                                      0EEA  ; SIDE EFFECTS:
                                      0EEA  ;
                                      0EEA  ;       R0 - R5 destroyed
                                      0EEA  ;--
                                      0EEA
                                      0EEA  LCK$SND_RMVDIR::
                                      0EEA          ; Compute CSID of directory system
                                      0EEA
        53  00000000'GF    D0         0EEA          MOVL    G^LCK$GL_DIRVEC,R3          ; Get address of directory vector
           51    44 A8     3C         0EF1          MOVZWL  RSB$W_HASHVAL(R8),R1        ; Get hash value
                    52     D4         0EF5          CLRL    R2                          ; Clear high order hash value
  51    50    51  F4 A3    7B         0EF7          EDIV    -12(R3),R1,R0,R1            ; Compute hash index (in R1)
           53      6341    D0         0EFD          MOVL    (R3)[R1],R3                 ; Get directory system
                    35     13         0F01          BEQL    70$                         ; Error
                                      0F03
                 F0FA'     30         0F03          BSBW    CNX$ALLOC_CDRP              ; Alloc. a CDRP and convert CSID to CSB
                 33 50     E9         0F06          BLBC    R0,80$                      ; No memory or CSID error
           2C A5    58     D0         0F09          MOVL    R8,CDRP$L_VAL1(R5)          ; Store RSB address in CDRP
           50    64 A3     D0         0F0D          MOVL    CSB$L_CLUB(R3),R0           ; Get address of CLUB
              00AC C0      B0         0F11          MOVW    CLUB$W_MEMSEQ(R0),-         ; Store membership seq. num.
                 34 A5                0F15                  CDRP$L_VAL3(R5)
                 42'AF     9E         0F17          MOVAB   B^BLD_RMVDIRMSG,-           ; Store address of message build routine
                 4C A5                0F1A                  CDRP$L_MSGBLD(R5)
                                      0F1C
              00000002                0F1C          .IF NE  CA$_MEASURE
           00000000'GF     D6         0F1C          INCL    G^PMS$GL_DIR_OUT
                                      0F22          .ENDC
                                      0F22
                 F0DB'     30         0F22          BSBW    CNX$SEND_MSG_CSB            ; Send the message
           50    2C A5     D0         0F25          MOVL    CDRP$L_VAL1(R5),R0          ; Get address of RSB
           00000000'GF     16         0F29          JSB     G^EXE$DEANONPAGED           ; Deallocate it
              50    55     D0         0F2F          MOVL    R5,R0                       ; Get address of CDRP
           00000000'GF     17         0F32          JMP     G^EXE$DEANONPAGED           ; Deallocate it
```

```
                    0F38
                    0F38  70$:    ; We are the directory system for this resource, yet RSB$M_DIRENTRY
                    0F38          ; is not set.  See comment under RELOOKUP for a description of
                    0F38          ; a race condition that is similar to what may have caused this.
                    0F38
                    0F38          BUG_CHECK       LOCKMGRERR,FATAL
                    0F3C
                    0F3C  80$:    ; No CDRP to allocate or CSID conversion error.  If insufficient
                    0F3C          ; memory, then place the RSB on the timeout queue.  Since we
                    0F3C          ; normally place LKBs on the timeout queue, we need to verify
                    0F3C          ; that the fields we use in the RSB are not needed.  Hence,
                    0F3C          ; the following ASSUME statements.
                    0F3C
                    0F3C          ASSUME  RSB$L_HSHCHN  EQ  LKB$L_ASTQFL
                    0F3C          ASSUME  RSB$L_CVTQFL  EQ  LKB$L_DUETIME
                    0F3C          ASSUME  RSB$Q_VALBLK+2  EQ  LKB$W_STATUS
                    0F3C
      54    58  D0  0F3C          MOVL    R8,R4                   ; Move RSB address
          FCAB  31  0F3F          BRW     INSERT_ON_TIMOUTQ
                    0F42
                    0F42          ; Action routine to build the actual remove directory
                    0F42          ; entry message.
                    0F42          ; Inputs are:
                    0F42          ;       R2      Address of message buffer
                    0F42          ;       R5      Address of CDRP
                    0F42          ; All registers except R0 and R1 must be preserved
                    0F42
                    0F42          ASSUME  LKMSG$B_RMOD  EQ  LKMSG$W_GROUP+2
                    0F42          ASSUME  LKMSG$B_RSNLEN  EQ  LKMSG$B_RMOD+1
                    0F42          ASSUME  LKMSG$T_RESNAM  EQ  LKMSG$B_RSNLEN+1
                    0F42          ASSUME  RSB$B_RMOD  EQ  RSB$W_GROUP+2
                    0F42          ASSUME  RSB$B_RSNLEN  EQ  RSB$B_RMOD+1
                    0F42          ASSUME  RSB$T_RESNAM  EQ  RSB$B_RSNLEN+1
                    0F42
                    0F42  BLD_RMVDIRMSG:
            3C  BB  0F42          PUSHR   #^M<R2,R3,R4,R5>
                    0F44          ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
 08 A2    0402 8F  B0  0F44       MOVW    #LKMSG$K_RMVDIR@8-      ; Store function and facility codes
                    0F4A                  !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
         34 A5  B0  0F4A          MOVW    CDRP$L_VAL3(R5),-       ; Store membership seq. num.
         0C A2      0F4D                  LKMSG$W_MEMSEQ(R2)
    51   2C A5  D0  0F4F          MOVL    CDRP$L_VAL1(R5),R1      ; Get address of RSB
         44 A1  B0  0F53          MOVW    RSB$W_HASHVAL(R1),-     ; Store hash value
         0E A2      0F56                  LKMSG$W_HASHVAL(R2)
    50   4F A1  9A  0F58          MOVZBL  RSB$B_RSNLEN(R1),R0     ; Get length of resource name
      50    04  C0  0F5C          ADDL    #4,R0                   ; Account for group, access mode, etc.
 4C A1    50    28  0F5F          MOVC3   R0,RSB$W_GROUP(R1),-    ; Move group no., resource name, etc.
         28 A2      0F63                  LKMSG$W_GROUP(R2)       ; inot message buffer
            3C  BA  0F65          POPR    #^M<R2,R3,R4,R5>
            05      0F67          RSB
```

```
                       0F68          .SBTTL  LCK$RCV_RMVDIR - Receive remove directory entry message
                       0F68
                       0F68    ;++
                       0F68    ; FUNCTIONAL DESCRIPTION:
                       0F68    ;
                       0F68    ;       This routine handles receiving a remove directory entry message.
                       0F68    ;       This system must be the directory system for this resource.
                       0F68    ;       The specified root resource is located in the resource hash table
                       0F68    ;       and if all of its queues are empty, then the resource is deallocated.
                       0F68    ;       If the wait queue has locks on it, then these must represent
                       0F68    ;       requests from this system being sent to the system that formerly
                       0F68    ;       managed this resource (the system that sent us this message).  In
                       0F68    ;       this case the resource is not deallocated.  Instead, the RSB$L_CSID
                       0F68    ;       field is cleared and we master the resource.  Clearing the field
                       0F68    ;       must be done now instead of when our lock messages are returned
                       0F68    ;       with the response "do locally".  Otherwise there is a race if the
                       0F68    ;       remote system follows the RMVDIR message with a directory lookup
                       0F68    ;       lock request.  We will then erroneously tell the other system to
                       0F68    ;       do it locally while the other system is telling us the same thing.
                       0F68    ;       Another race condition occurs if this message is delayed on the
                       0F68    ;       sending side dur to insufficient pool.  In that case, it is
                       0F68    ;       possible for us to receive a do locally lock response before
                       0F68    ;       receiving this message.  That is why we tolerate the RSB$L_CSID
                       0F68    ;       field being zero if we it doesn't match the remote system's CSID.
                       0F68    ;
                       0F68    ; CALLING SEQUENCE:
                       0F68    ;
                       0F68    ;       JSB     LCK$RCV_RMVDIR (called by the SCS received message routine)
                       0F68    ;       IPL must be at IPL$_SYNCH
                       0F68    ;
                       0F68    ; INPUT PARAMETERS:
                       0F68    ;
                       0F68    ;       R2      Address of message buffer
                       0F68    ;       R3      Address of CSB
                       0F68    ;
                       0F68    ; OUTPUT PARAMETERS:
                       0F68    ;
                       0F68    ;       None
                       0F68    ;
                       0F68    ; SIDE EFFECTS:
                       0F68    ;
                       0F68    ;       R0 - R5 destroyed
                       0F68    ;--
                       0F68
                       0F68    LCK$RCV_RMVDIR::
             00000002  0F68          .IF NE  CA$_MEASURE
      00000000'GF  D6  0F68          INCL    G^PMS$GL_DIR_IN
                       0F6E          .ENDC
                       0F6E
         0D0C 8F   BB  0F6E          PUSHR   #^M<R2,R3,R8,R10,R11>
      54    24 A2  9E  0F72          MOVAB   LKMSG$L_PARMSTLKID(R2),R4 ; Set start of resource description
               64  D4  0F76          CLRL    (R4)                      ; Clear parent lock id
      5A    2B A2  9A  0F78          MOVZBL  LKMSG$B_RSNLEN(R2),R10    ; Get length of resource name
         5A    08  C0  0F7C          ADDL    #8,R10                    ; Account for extra fields to match
      51    0E A2  3C  0F7F          MOVZWL  LKMSG$W_HASHVAL(R2),R1    ; Get hash value
      00000000'GF  16  0F83          JSB     G^LCK$SRCH_HSHTBL         ; Search hash table for resource
            47 50  E8  0F89          BLBS    R0,80$                    ; Resource not found
```

```
                          0F8C
                          0F8C        ; Resource was been located, R5 points to RSB.  Verify it is a
                          0F8C        ; directory entry mastered by other system.  Then verify all queues
                          0F8C        ; are empty.  If granted or conversion queues are not empty then
                          0F8C        ; we are in serious trouble.   It's okay if wait queue is not empty,
                          0F8C        ; as this represents the race condition described above.
                          0F8C
                          0F8C        ASSUME  RSB$L_CVTQFL  EQ  RSB$L_GRQFL+8
                          0F8C        ASSUME  RSB$L_WTQFL   EQ  RSB$L_CVTQFL+8
                          0F8C        ASSUME  RSB$M_DIRENTRY  EQ  1
                          0F8C
        58    55     D0   0F8C        MOVL    R5,R8                         ; Move address of RSB
     40 0E    A8     E9   0F8F        BLBC    RSB$W_STATUS(R8),80$          ; Verify directory entry bit is set
   53   04    AE     D0   0F93        MOVL    4(SP),R3                      ; Get address of CSB
        4C    A3     D1   0F97        CMPL    CSB$L_CSID(R3),-              ; Verify sending system is
        38    A8          0F9A                RSB$L_CSID(R8)                ; mastering resource
              07     13   0F9C        BEQL    20$                           ; It is
        38    A8     D5   0F9E        TSTL    RSB$L_CSID(R8)                ; It's not, but it's ok if this system
              29     13   0FA1        BEQL    40$                           ; is mastering the resource
              2E     12   0FA3        BNEQ    80$                           ; Error
   50   10    A8     DE   0FA5  20$:  MOVAL   RSB$L_GRQFL(R8),R0            ; Get address of granted queue
        50    60     D1   0FA9        CMPL    (R0),R0                       ; Is granted queue empty?
              25     12   0FAC        BNEQ    80$                           ; No
        50    08     C0   0FAE        ADDL    #8,R0                         ; Yes, get address of conversion queue
        50    60     D1   0FB1        CMPL    (R0),R0                       ; Is conversion queue empty?
              1D     12   0FB4        BNEQ    80$                           ; No
        50    08     C0   0FB6        ADDL    #8,R0                         ; Yes, get address of wait queue
        50    60     D1   0FB9        CMPL    (R0),R0                       ; Is wait queue empty?
              05     13   0FBC        BEQL    30$                           ; Yes
        38    A8     D4   0FBE        CLRL    RSB$L_CSID(R8)                ; No, we master resource
              09     11   0FC1        BRB     40$
                          0FC3
                          0FC3  30$:  ; All queues are empty and checks verified.  Deallocate RSB
                          0FC3
        38    A8     D4   0FC3        CLRL    RSB$L_CSID(R8)                ; Do this so that LCK$DEALLOC_RSB
                          0FC6                                              ; really deallocates it
         00000000'GF     16   0FC6        JSB     G^LCK$DEALLOC_RSB             ; Deallocate it
                          0FCC
         0D0C 8F     BA   0FCC  40$:  POPR    #^M<R2,R3,R8,R10,R11>
            F02D'    31   0FD0        BRW     CNX$DEALL_MSG_BUF_CSB         ; Deallocate message buffer and return
                          0FD3
                          0FD3
                          0FD3  80$:  ; One of the following bugs:
                          0FD3        ;       Resource not found (R5 = 0)
                          0FD3        ;       Not marked as a directory entry (RSB$M_DIRENTRY not set)
                          0FD3        ;       Not mastered by sending system (RSB$L_CSID not equal CSB$L_CSID
                          0FD3        ;       and also not zero)
                          0FD3        ;       Conversion or granted queue not empty (check queues)
                          0FD3
                          0FD3        BUG_CHECK       LOCKMGRERR,FATAL
```

```
                              0FD7          .SBTTL  VERIFYREMLKID - Verify remote lock id
                              0FD7
                              0FD7  ;++
                              0FD7  ; FUNCTIONAL DESCRIPTION:
                              0FD7  ;
                              0FD7  ;       This routine verifies a lock id sent by another system
                              0FD7  ;       and converts it into a LKB address.
                              0FD7  ;
                              0FD7  ; CALLING SEQUENCE:
                              0FD7  ;
                              0FD7  ;       BSBW    VERIFYREMLKID
                              0FD7  ;       IPL must be at IPL$_SYNCH
                              0FD7  ;
                              0FD7  ; INPUT PARAMETERS:
                              0FD7  ;
                              0FD7  ;       R0      Lock id on remote system
                              0FD7  ;       R1      Lock id on this system
                              0FD7  ;
                              0FD7  ; OUTPUT PARAMETERS:
                              0FD7  ;
                              0FD7  ;       R0      Completion code
                              0FD7  ;       R6      Address of LKB
                              0FD7  ;
                              0FD7  ; COMPLETION CODES:
                              0FD7  ;
                              0FD7  ;       SS$_NORMAL      Lock id was valid can converted to LKB address
                              0FD7  ;       SS$_IVLOCKID    Invalid lock id
                              0FD7  ;
                              0FD7  ; SIDE EFFECTS:
                              0FD7  ;
                              0FD7  ;       R4 is destroyed
                              0FD7  ;
                              0FD7  ; NOTE:
                              0FD7  ;
                              0FD7  ;       This routine does two consistency checks.  The first is that
                              0FD7  ;       it verifies the lock id is valid via the sequence number check.
                              0FD7  ;       If the lock id fails this check, then an error is returned to
                              0FD7  ;       the caller as this is allowed in some cases and is fatal in others.
                              0FD7  ;       However, if the lock id passes this check, then another check
                              0FD7  ;       is made that compares the remote lock id as sent by the remote
                              0FD7  ;       system with the remote lock id stored here in the LKB.  If this
                              0FD7  ;       check fails then it is immediately fatal as the first check should
                              0FD7  ;       catch all races that cause lock ids to not match across systems.
                              0FD7  ;       Also note that this second check is not perfect in that it should
                              0FD7  ;       also check CSB addresses.  This is considered unnecessary as the
                              0FD7  ;       additional protection that check offers is small.
                              0FD7  ;--
                              0FD7
                              0FD7  VERIFYREMLKID:
              56     51  3C   0FD7          MOVZWL  R1,R6                   ; Put lockid index in R6
    00000000'GF      56  D1   0FDA          CMPL    R6,G^LCK$GL_MAXID       ; Is the lock id too big?
                     1D  1A   0FE1          BGTRU   40$                     ; Yes
54  00000000'GF          D0   0FE3          MOVL    G^LCK$GL_IDTBL,R4       ; *** May combine with next instr.
         56        6446  D0   0FEA          MOVL    (R4)[R6],R6             ; Get LKB address
                     10  18   0FEE          BGEQ    40$                     ; Unallocated id
          30 A6      51  D1   0FF0          CMPL    R1,LKB$L_LKID(R6)       ; Check sequence number
                     0A  12   0FF4          BNEQ    40$                     ; Not valid
```

```
     54 A6    50  D1  0FF6        CMPL    R0,LKB$L_REMLKID(R6)    ; Check remote lock id
              0A  12  0FFA        BNEQ    50$                     ; Doesn't match
        50   00'  3C  0FFC        MOVZWL  S^#SS$_NORMAL,R0        ; Success
                  05  0FFF        RSB
                      1000
   50   0000'8F   3C  1000  40$:  MOVZWL  #SS$_IVLOCKID,R0        ; Invalid lock id
                  05  1005        RSB
                      1006
                      1006  50$:  BUG_CHECK       INVLOCKID,FATAL; Invalid remote lockid
                      100A
                      100A
                      100A
                      100A
                      100A        .END
```

```
$$BASE                        = FFFFFFFA
$$DISPL                       = 00000002
$$GENSW                       = 00000001
$$HIGH                        = 00000001
$$LIMIT                       = 00000007
$$LOW                         = FFFFFFFA
$$MNSW                        = 00000001
$$MXSW                        = 00000001
BLDLKB_ERROR                    00000A03 R      02
BLDLKB_ERROR2                   00000A16 R      02
BLD_BLKINGMSG                   00000D57 R      02
BLD_CVTCOM                      000002FA R      02
BLD_CVTMSG                      00000289 R      02
BLD_CVTMSGR                     000002D0 R      02
BLD_CVTRSP                      00000488 R      02
BLD_DEQMSG                      00000E41 R      02
BLD_GRANTEDMSG                  00000C0E R      02
BLD_LOCKCOM                     00000702 R      02
BLD_LOCKMSG                     000006FC R      02
BLD_LOCKRSP                     00000A6B R      02
BLD_RMVDIRMSG                   00000F42 R      02
BLD_RSPMSG                      00000A5A R      02
BUG$_INSFLOCKID                 ********    X   02
BUG$_INSFPOOL                   ********    X   02
BUG$_INVLOCKID                  ********    X   02
BUG$_LKBREFNEG                  ********    X   02
BUG$_LOCKMGRERR                 ********    X   02
BUILD_LKB                       00000AD7 R      02
BUILD_LKB_ERR                   00000AD6 R      02
CA$_MEASURE                   = 00000002
CDRP$L_MSGBLD                 = 0000004C
CDRP$L_VAL1                   = 0000002C
CDRP$L_VAL2                   = 00000030
CDRP$L_VAL3                   = 00000034
CDRP$L_VAL4                   = 00000038
CDRP$L_VAL6                   = 00000040
CDRP$L_VAL8                   = 00000048
CDRP$L_VAL9                   = 0000005C
CGRNTD_FORK                     0000024D R      02
CGRNTD_PROC                     00000160 R      02
CHECK_RSB                       000005B5 R      02
CLSMSG$B_FACILITY             = 00000008
CLSMSG$B_FUNC                 = 00000009
CLSMSG$K_FAC_LCK              = 00000002
CLSMSG$L_RSPID                = 00000004
CLSMSG$M_RESPMSG              = 00000080
CLU$GL_CLUB                     ********    X   02
CLUB$W_MEMSEQ                 = 000000AC
CNOTQD_FORK                     0000024D R      02
CNOTQD_PROC                     000001A3 R      02
CNX$ALLOC_CDRP                  ********    X   02
CNX$ALLOC_WARMCDRP              ********    X   02
CNX$DEALL_MSG_BUF_CSB           ********    X   02
CNX$DEALL_WARMCDRP_CSB          ********    X   02
CNX$INIT_CDRP                   ********    X   02
CNX$RCV_REJECT                  ********    X   02
CNX$SEND_MSG_CSB                ********    X   02
CNX$SEND_MSG_RESP               ********    X   02
CSB$L_CLUB                    = 00000064
CSB$L_CSID                    = 0000004C
CSID_ERROR                      0000054D R      02
CVTQED_FORK                     00000281 R      02
CVTQED_PROC                     00000193 R      02
CVTRETRY_PROC                   00000149 R      02
CVT_INVLKID                     00000306 R      02
CVT_RETRY                       00000238 R      02
CVT_SCS_WAIT                    000000F0 R      02
CVT_WAIT                        000000F3 R      02
DEALL_WARMCDRP                  000006DD R      02
DIR_ENTER                       000008DF R      02
DIR_LOOKUP                      000008AC R      02
DOLOCL_FORK                     0000068C R      02
DOLOCL_PROC                     00000568 R      02
DYN$C_LKB                     = 00000035
DYN$C_RSB                     = 00000036
EVT$_AST                        ********    X   02
EXE$ALONONPAGED                 ********    X   02
EXE$DEANONPAGED                 ********    X   02
EXE$GL_ABSTIM                   ********    X   02
EXISTING_RESOURCE               00000816 R      02
GRANTD_FORK                     000006A5 R      02
GRANTD_PROC                     00000572 R      02
INSERT_ON_TIMOUTQ               00000BED R      02
IPL$_ASTDEL                   = 00000002
IPL$_SCS                      = 00000008
IPL$_SYNCH                    = 00000008
LCK$BLD_REBLDLOCK               00000761 RG     02
LCK$CHECK_DIRENTRY              ********    X   02
LCK$CHECK_RSB                   ********    X   02
LCK$COMPAT_TBL                  ********    X   02
LCK$COMP_GGMODE                 ********    X   02
LCK$CVTNOTQED                   ********    X   02
LCK$CVTRSP_TBL                  00000000 RG     02
LCK$CVT_GRANTED                 ********    X   02
LCK$DEALLOC_LKB                 00000B65 RG     02
LCK$DEALLOC_RSB                 ********    X   02
LCK$DEQLOCK                     ********    X   02
LCK$DISPATCH                    00000006 RG     02
LCK$EXTEND_IDTBL                ********    X   02
LCK$GB_REBLD_STATE              ********    X   02
LCK$GB_STALLREQS                ********    X   02
LCK$GL_DIRVEC                   ********    X   02
LCK$GL_IDTBL                    ********    X   02
LCK$GL_MAXID                    ********    X   02
LCK$GL_NXTID                    ********    X   02
LCK$GL_TIMOUTQ                  ********    X   02
LCK$GRANTCVTS                   ********    X   02
LCK$GRANTWTRS                   ********    X   02
LCK$GRANT_LOCK                  ********    X   02
LCK$GRANT_LOCK_ALT              ********    X   02
LCK$GRANT_REM                   ********    X   02
LCK$K_PWMODE                  = 00000004
LCK$LOCAL_CVT                   ********    X   02
LCK$LOCAL_LOCK                  ********    X   02
```

```
LCK$M_CANCEL                = 00000002                 LKB$L_ASTQFL           = 00000000
LCK$M_INVVALBLK             = 00000004                 LKB$L_BLKASTADR        = 00000020
LCK$NOT_QUEUED                ********   X      02     LKB$L_CSID             = 00000058
LCK$QUEUECVT                  ********   X      02     LKB$L_DLCKPRI          = 00000024
LCK$QUEUED_EXIT               ********   X      02     LKB$L_DUETIME          = 00000018
LCK$QUEUEWAIT                 ********   X      02     LKB$L_EPID             = 00000014
LCK$QUEUE_BLKAST              ********   X      02     LKB$L_LKID             = 00000030
LCK$QUEUE_REM                 ********   X      02     LKB$L_LKST1            = 0000002C
LCK$RCV_BLKING                00000D6D RG       02     LKB$L_OLDBLKAST        = 0000005C
LCK$RCV_CVTMSG                0000030A RG       02     LKB$L_PARENT           = 00000048
LCK$RCV_CVTREQ                0000030C RG       02     LKB$L_PID              = 0000000C
LCK$RCV_DEQ                   00000E68 RG       02     LKB$L_REMLKID          = 00000054
LCK$RCV_DLCKFND               ********   X      02     LKB$L_RSB              = 00000050
LCK$RCV_GRANTED               00000C59 RG       02     LKB$L_SQFL             = 00000038
LCK$RCV_LOCKREQ               000007B6 RG       02     LKB$M_ASYNC            = 00000004
LCK$RCV_REDO_SRCH             ********   X      02     LKB$M_BLKASTQED        = 00000008
LCK$RCV_RMVDIR                00000F68 RG       02     LKB$M_CVTTOSYS         = 00000100
LCK$RCV_SRCHDLCK              ********   X      02     LKB$M_DBLKAST          = 00000002
LCK$RCV_TIMESTAMP_RQST        ********   X      02     LKB$M_DCPLAST          = 00000001
LCK$REBLD_LOCK                ********   X      02     LKB$M_MSTCPY           = 00000010
LCK$SND_BLKING                00000D07 RG       02     LKB$M_PKAST            = 00000010
LCK$SND_CVTREQ                00000091 RG       02     LKB$M_PROTECT          = 00000200
LCK$SND_DEQCV                 00000DD9 RG       02     LKB$M_TIMOUTQ          = 00000040
LCK$SND_DEQGR                 00000DE5 RG       02     LKB$M_WASSYSOWN        = 00000080
LCK$SND_DEQWT                 00000DD2 RG       02     LKB$V_ASYNC            = 00000002
LCK$SND_GRANTED               00000BAF RG       02     LKB$V_BLKASTQED        = 00000003
LCK$SND_LOCKREQ               000004E2 RG       02     LKB$V_DBLKAST          = 00000001
LCK$SND_RMVDIR                00000EEA RG       02     LKB$V_MSTCPY           = 00000004
LCK$SRCH_HSHTBL               ********   X      02     LKB$V_PROTECT          = 00000009
LCK$SYNC_EXIT                 ********   X      02     LKB$V_TIMOUTQ          = 00000006
LCK$V_CANCEL                = 00000001                 LKB$W_FLAGS            = 00000028
LCK$V_CVTSYS                = 00000006                 LKB$W_REFCNT           = 0000004C
LCK$V_NOQUEUE               = 00000002                 LKB$W_RQSEQNM          = 00000010
LCK$V_PROTECT               = 00000008                 LKB$W_SIZE             = 00000008
LCK$V_RECOVER               = 00000007                 LKB$W_STATUS           = 0000002A
LCK$V_VALBLK                = 00000000                 LKMSG$B_GRMODE         = 0000001B
LCKRETRY_PROC                 00000562 R        02     LKMSG$B_LCKSTATE       = 0000006A
LCK_RETRY                     00000680 R        02     LKMSG$B_LSTATUS        = 00000068
LCK_SCS_WAIT                  00000505 R        02     LKMSG$B_RMOD           = 0000002A
LCK_SENDERR                   00000625 R        02     LKMSG$B_RQMODE         = 0000001A
LCK_WAIT                      00000508 R        02     LKMSG$B_RSNLEN         = 0000002B
LKB$B_GRMODE                = 00000035                 LKMSG$B_RSTATUS        = 00000069
LKB$B_RMOD                  = 0000000B                 LKMSG$B_STATE          = 0000001E
LKB$B_RQMODE                = 00000034                 LKMSG$K_BLKING         = 00000005
LKB$B_STATE                 = 00000036                 LKMSG$K_CVTLCKM        = 00000006
LKB$K_CONVERT               = 00000000                 LKMSG$K_CVTLCKR        = 00000007
LKB$K_GRANTED               = 00000001                 LKMSG$K_DEQ            = 00000003
LKB$K_LENGTH                = 00000060                 LKMSG$K_DLCKFND        = 0000000B
LKB$K_RETRY                 = FFFFFFFE                 LKMSG$K_GRANTED        = 00000002
LKB$K_RSPDOLOCL             = FFFFFFF9                 LKMSG$K_NEWLOCK        = 00000001
LKB$K_RSPGRANTD             = FFFFFFFA                 LKMSG$K_REBLDLOCK      = 00000008
LKB$K_RSPNOTQED             = FFFFFFFC                 LKMSG$K_REDO_SRCH      = 0000000C
LKB$K_RSPQUEUED             = FFFFFFFB                 LKMSG$K_RMVDIR         = 00000004
LKB$K_RSPRESEND             = FFFFFFF8                 LKMSG$K_SRCHDLCK       = 0000000A
LKB$K_SCSWAIT               = FFFFFFFD                 LKMSG$K_TSRQST         = 00000009
LKB$K_WAITING               = FFFFFFFF                 LKMSG$L_BLKASTFLG      = 0000001C
LKB$L_AST                   = 00000010                 LKMSG$L_CSID           = 00000018
```

```
LKMSG$L_DLCKPRI_CVT     = 00000034
LKMSG$L_DLCKPRI_NEW     = 0000004C
LKMSG$L_EPIDCVT         = 00000030
LKMSG$L_EPIDNEW         = 00000010
LKMSG$L_MSTLKID         = 00000010
LKMSG$L_PARMSTLKID      = 00000024
LKMSG$L_PARPRCLKID      = 00000020
LKMSG$L_PRCLKID         = 00000014
LKMSG$L_VALBLKFLG       = 0000001C
LKMSG$L_VALSEQALT       = 00000064
LKMSG$L_VALSEQNUM       = 00000030
LKMSG$Q_VALBLK          = 00000020
LKMSG$Q_VALBLKALT       = 00000054
LKMSG$T_RESNAM          = 0000002C
LKMSG$W_FLAGS           = 00000018
LKMSG$W_GROUP           = 00000028
LKMSG$W_HASHVAL         = 0000000E
LKMSG$W_LKBSTATUS       = 00000018
LKMSG$W_MEMSEQ          = 0000000C
LKMSG$W_RQSEQALT        = 00000050
LKMSG$W_RQSEQNM         = 00000030
LKMSG$W_RSBSTATUS       = 0000001A
LKMSG$W_STATUS          = 0000001C
LOCK_GRANTED              000009DF R     02
LOCK_QUEUED               000009E7 R     02
NEW_RESOURCE              00000950 R     02
NOTQED_FORK               00000697 R     02
NOTQED_PROC               000005B1 R     02
NOTQUEUED_EXIT            00000922 R     02
NO_LOCKIDS                00000AC4 R     02
PCB$L_DLCKPRI           = 0000010C
PCB$L_EPID              = 00000064
PCB$L_PID               = 00000060
PMS$GL_BLK_IN             ********    X  02
PMS$GL_BLK_OUT            ********    X  02
PMS$GL_DEQ_IN             ********    X  02
PMS$GL_DEQ_LOC            ********    X  02
PMS$GL_DEQ_OUT            ********    X  02
PMS$GL_DIR_IN             ********    X  02
PMS$GL_DIR_OUT            ********    X  02
PMS$GL_ENQCVT_IN          ********    X  02
PMS$GL_ENQCVT_OUT         ********    X  02
PMS$GL_ENQNEW_IN          ********    X  02
PMS$GL_ENQNEW_OUT         ********    X  02
PMS$GL_ENQNOTQD           ********    X  02
PR$_IPL                   ********    X  02
PSL$V_IPL               = 00000010
QUEUED_EXIT               00000928 R     02
REBLD_LOCK                000009FE R     02
RELOOKUP                  000008D2 R     02
RESEND                    000004E7 R     02
RESEND_BRANCH             0000054B R     02
RESEND_FORK               00000634 R     02
RESPOND_DOLOCL            0000090C R     02
RESPOND_NOTQED            00000913 R     02
RESPOND_RESEND            000008B7 R     02
RSB$B_CGMODE            = 0000000D
RSB$B_DEPTH             = 0000000B
RSB$B_GGMODE            = 0000000C
RSB$B_RMOD              = 0000004E
RSB$B_RSNLEN            = 0000004F
RSB$B_TYPE              = 0000000A
RSB$K_LENGTH            = 00000050
RSB$L_CSID              = 00000038
RSB$L_CVTQFL            = 00000018
RSB$L_GRQFL             = 00000010
RSB$L_HSHCHN            = 00000000
RSB$L_HSHCHNBK          = 00000004
RSB$L_PARENT            = 00000048
RSB$L_VALSEQNUM         = 0000003C
RSB$L_WTQBL             = 00000024
RSB$L_WTQFL             = 00000020
RSB$M_DIRENTRY          = 00000001
RSB$M_VALINVLD          = 00000002
RSB$Q_VALBLK            = 00000028
RSB$T_RESNAM            = 00000050
RSB$V_VALINVLD          = 00000001
RSB$W_BLKASTCNT         = 00000042
RSB$W_GROUP             = 0000004C
RSB$W_HASHVAL           = 00000044
RSB$W_REFCNT            = 00000040
RSB$W_RQSEQNM           = 00000046
RSB$W_SIZE              = 00000008
RSB$W_STATUS            = 0000000E
RSN$_CLUSTRAN           = 0000000E
RSN$_NPDYNMEM           = 00000003
RSN$_SCS                = 0000000D
SCH$GL_CURPCB             ********    X  02
SCH$GL_PCBVEC             ********    X  02
SCH$RSE                   ********    X  02
SCH$RWAIT                 ********    X  02
SENDCVTMSG                000001B5 R     02
SENDCVTREQ                000001E0 R     02
SENDLOCKREQ               000005C5 R     02
SEND_BLOCKING             00000D1D R     02
SEND_CVTRSP               00000452 R     02
SS$_CANCELGRANT           ********    X  02
SS$_INSFMEM               ********    X  02
SS$_IVLOCKID              ********    X  02
SS$_NOLOCKID              ********    X  02
SS$_NORMAL                ********    X  02
SS$_NOSUCHNODE            ********    X  02
SS$_NOTQUEUED             ********    X  02
VERIFYREMLKID             00000FD7 R     02
WAITFORMEM                00000551 R     02
WAITNG_FORK               0000069E R     02
WAITNG_PROC               000005A5 R     02
WAKE_PROCESS              000006E3 R     02
```

```
                                      +----------------+
                                      ! 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
$$$020                      0000100A  ( 4106.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE

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

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                   29     00:00:00.05    00:00:01.15
Command processing              108     00:00:00.46    00:00:02.88
Pass 1                          564     00:00:20.25    00:01:19.02
Symbol table sort                 0     00:00:01.56    00:00:04.47
Pass 2                          414     00:00:05.88    00:00:25.41
Symbol table output               1     00:00:00.17    00:00:00.49
Psect synopsis output             0     00:00:00.01    00:00:00.01
Cross-reference output            0     00:00:00.00    00:00:00.00
Assembler run totals           1118     00:00:28.39    00:01:53.43

The working set limit was 2400 pages.
159344 bytes (312 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1471 non-local and 144 local symbols.
3340 source lines were read in Pass 1, producing 33 object records in Pass 2.
31 pages of virtual memory were used to define 29 macros.

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

Macro library name                            Macros defined
------------------                            --------------
_$255$DUA28:[SYSLOA.OBJ]CLUSTER.MLB;1                1
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                      16
_$255$DUA28:[SYSLIB]STARLET.MLB;2                    6
TOTALS (all libraries)                              23

1478 GETS were required to define 23 macros.

There were no errors, warnings or information messages.

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

0394 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CSPOPCOM
LIS

CSPWAIT
LIS

CSPRCPCAC
LIS

CSPCJFRES
LIS

CSPQUORUM
LIS

DISTRLKI
LIS

CSPMOUNT
LIS

CSPVECTOR
LIS

CSPCLIENT
LIS

DSTRDLCK
LIS

DSTRLOCK
LIS

0395 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

ERRSUBUV1
LIS

ERRSUB790
LIS

ERRSUB780
LIS

ERRSUB750
LIS

FALMAN
LIS

ERRSUB730
LIS