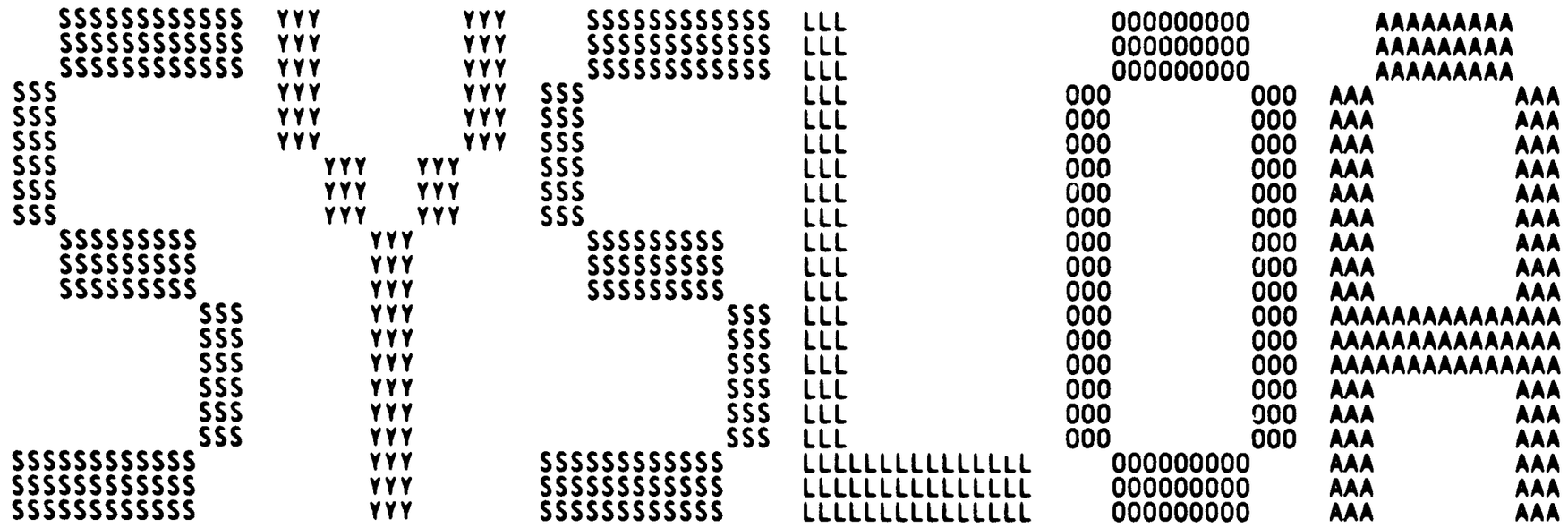
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000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

FILEID**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     1          .TITLE  DSTRLOCK - Distributed Lock Manager Loadable Code
0000     2          .IDENT  'V04-001'
0000     3
0000     4  ;
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27  ;
0000    28
0000    29  ;++
0000    30  ;
0000    31  ; FACILITY:     Executive, system services and fork level code
0000    32  ;
0000    33  ; ABSTRACT:
0000    34  ;       This module contains routines used to implement distributed
0000    35  ;       lock manager functions.
0000    36  ;
0000    37  ; ENVIRONMENT:  Kernel mode, fork level, loadable code
0000    38  ;
0000    39  ;--
0000    40  ;
0000    41  ; AUTHOR: Steve Beckhardt,       CREATION DATE: 11-Oct-1982
0000    42  ;
0000    43  ; MODIFIED BY:
0000    44  ;
0000    45  ;       V04-001 SRB0144         Steve Beckhardt                 6-Sep-1984
0000    46  ;               Fixed race condition in conversion code that could
0000    47  ;               ultimately clobber value blocks.  Also fixed bug in
0000    48  ;               CVTRETRY_PROC code for locks in RETRY state that are
0000    49  ;               ultimately not queued.
0000    50  ;
0000    51  ;       V03-023 SRB0140         Steve Beckhardt                 2-Aug-1984
0000    52  ;               Change SRB0136 was incomplete.  Fixed one more place
0000    53  ;               to bugcheck this system rather than returning an error.
0000    54  ;
0000    55  ;       V03-022 SRB0136         Steve Beckhardt                 9-Jul-1984
0000    56  ;               Changed handling of insufficient resources during failover
0000    57  ;               to bugcheck the system with insufficient resources.
```

```
0000     58 ;
0000     59 ;       V03-021 SRB0134         Steve Beckhardt                 26-Jun-1984
0000     60 ;               Simplified message build routine for rebuild lock message
0000     61 ;               to not handle transient SCS states (RSPxxx).  This is
0000     62 ;               a result of a corresponding change in the rebuild code
0000     63 ;               to put all those locks into RETRY state.
0000     64 ;
0000     65 ;       V03-020 SRB0132         Steve Beckhardt                 25-May-1984
0000     66 ;               Fixed bug where cleanup path from building an LKB error
0000     67 ;               didn't decrement parent's sub-RSB reference count.
0000     68 ;
0000     69 ;       V03-019 SRB0120         Steve Beckhardt                 20-Apr-1984
0000     70 ;               Fixed race conditions in LCK$RCV_RMVDIR code.
0000     71 ;               Fixed bug where locks in SCS states were rebuilt incorrectly.
0000     72 ;
0000     73 ;       V03-018 SRB0119         Steve Beckhardt                 2-Apr-1984
0000     74 ;               Added support for waiting for pool.  Fixed several bugs.
0000     75 ;
0000     76 ;       V03-017 SRB0117         Steve Beckhardt                 19-Mar-1984
0000     77 ;               Added support for distributed deadlock detection.
0000     78 ;
0000     79 ;       V03-016 SRB0115         Steve Beckhardt                 23-Feb-1984
0000     80 ;               Added support for maintaining deadlock priority field in
0000     81 ;               master-copy locks.  Fixed bug in conversion code.
0000     82 ;
0000     83 ;       V03-015 SRB0110         Steve Beckhardt                 27-Jan-1984
0000     84 ;               Added checking of MEMSEQ of incoming REBLDLOCK messages.
0000     85 ;               Added checking of rebuild state in message dispatching.
0000     86 ;
0000     87 ;       V03-014 SRB0108         Steve Beckhardt                 11-Jan-1984
0000     88 ;               Added support for new failin/out design with hashed root
0000     89 ;               directory.  Added support for sending HASHVAL in lock
0000     90 ;               messages.  Added support for maintaining EPID in locks.
0000     91 ;               Changed PMS counters.
0000     92 ;
0000     93 ;       V03-013 SRB0106         Steve Beckhardt                 6-Dec-1983
0000     94 ;               Changed LKB$L_REFCNT,RSB$L_REFCNT, and RSB$L_BLKASTCNT
0000     95 ;               to word fields.
0000     96 ;
0000     97 ;       V03-012 SRB0104         Steve Beckhardt                 17-Oct-1983
0000     98 ;               Fixed bug in LCK$SND_BLKING.
0000     99 ;
0000    100 ;       V03-011 SRB0100         Steve Beckhardt                 18-Jul-1983
0000    101 ;               Fixed PMS counters.  Fixed handling of receiving a lock
0000    102 ;               granted message to handle canceled locks with blocking ASTs.
0000    103 ;
0000    104 ;       V03-010 SRB0097         Steve Beckhardt                 23-Jun-1983
0000    105 ;               Added support for RECOVER bit.  Made several changes
0000    106 ;               to support multi-node failover.  Added support for
0000    107 ;               converting new locks to be system owned.
0000    108 ;
0000    109 ;       V03-009 SRB0093         Steve Beckhardt                 31-May-1983
0000    110 ;               Simplified use of LKB state codes.
0000    111 ;
0000    112 ;       V03-008 SRB0090         Steve Beckhardt                 18-May-1983
0000    113 ;               Added support for extending the lock id. table.  Added
0000    114 ;               support for multi-node failover.  Added PMS counters.
```

```
0000   115 ;                Fixed bug involving system owned locks.  Added support
0000   116 ;                for stalling requests during failover.
0000   117 ;
0000   118 ;        V03-007 SRB0083         Steve Beckhardt                 20-Apr-1983
0000   119 ;                Finished adding support for INVVALBLK and CANCEL flags.
0000   120 ;                Rewrote conversion code to send messages without RSPIDs
0000   121 ;                in some cases.
0000   122 ;
0000   123 ;        V03-006 SRB0077         Steve Beckhardt                 14-Apr-1983
0000   124 ;                Removed use of LKB$M_PRCCPY bit.
0000   125 ;
0000   126 ;        V03-005 SRB0073         Steve Beckhardt                 29-Mar-1983
0000   127 ;                Changed references to CSB to CSID in LKB and RSB.
0000   128 ;                Started adding support for $DEQ flags INVVALBLK and CANCEL.
0000   129 ;
0000   130 ;        V03-004 SRB0069         Steve Beckhardt                 7-Mar-1983
0000   131 ;                Changed HALT instructions to BUG_CHECKs.  Changed handling
0000   132 ;                of value blocks on conversions to return value block on
0000   133 ;                conversions to same lock mode.
0000   134 ;
0000   135 ;        V03-003 SRB0063         Steve Beckhardt                 21-Jan-1983
0000   136 ;                Made a number of bug fixes.
0000   137 ;
0000   138 ;        V03-002 SRB0062         Steve Beckhardt                 11-Jan-1983
0000   139 ;                Changed BSBW to JSB.
0000   140 ;
0000   141 ;
```

```
                        .SBTTL  DECLARATIONS
                ;
                ; INCLUDE FILES:
                ;

                ;
                ; MACROS:
                ;

                        $ACBDEF                         ; ACB offsets
                        $CADEF                          ; Conditional assembly switches
                        $CDRPDEF                        ; CDRP offsets
                        $CLSMSGDEF                      ; Cluster message offsets
                        $CLUBDEF                        ; CLUB offsets
                        $CSBDEF                         ; CSB offsets
                        $DYNDEF                         ; Data structure names
                        $FKBDEF                         ; Fork block offsets
                        $IPLDEF                         ; IPL definitions
                        $LCKDEF                         ; LCK definitions
                        $LKBDEF                         ; LKB offsets
                        $PCBDEF                         ; PCB offsets
                        $PRIDEF                         ; Priority definitions
                        $PSLDEF                         ; PSL definitions
                        $RSBDEF                         ; RSB offsets
                        $RSNDEF                         ; Resource numbers

                ;
                ; EQUATED SYMBOLS:
                ;

                ;
                ; OWN STORAGE:
                ;

                        .PSECT  $$$020

                ;*****************************************************************************
                ;
                ; NOTE: The following assumption is in effect for this entire module.
                ;
                ;*****************************************************************************

                        ASSUME  IPL$_SYNCH  EQ  IPL$_SCS
```

```
                        ;
                        ;                       CONVERSION RESPONSE TABLE
                        ;
                        ; This table indicates which conversions require a response message .
                        ;
                        ;                                       TO
                        ;
                        ;                    |   NL   |   CR   |   CW   |   PR   |   PW   |   EX
                        ;                    | NL/SW  |  R/SW  |  W/SW  |  R/SR  |  W/SR  |  W/NL
                        ;          ----------+--------+--------+--------+--------+--------+--------
                        ;          NL NL/SW  |  yes   |  yes   |  yes   |  yes   |  yes   |  yes
                        ;          ----------+--------+--------+--------+--------+--------+--------
                        ;          CR  R/SW  |  no    |  yes   |  yes   |  yes   |  yes   |  yes
                        ;          ----------+--------+--------+--------+--------+--------+--------
                        ;          CW  W/SW  |  no    |  no    |  no    |  yes   |  yes   |  yes
                        ; FROM     ----------+--------+--------+--------+--------+--------+--------
                        ;          PR  R/SR  |  no    |  no    |  yes   |  no    |  yes   |  yes
                        ;          ----------+--------+--------+--------+--------+--------+--------
                        ;          PW  W/SR  |  no    |  no    |  no    |  no    |  no    |  yes
                        ;          ----------+--------+--------+--------+--------+--------+--------
                        ;          EX  W/NL  |  no    |  no    |  no    |  no    |  no    |  no
                        ;
                        ;
                        LCK$CVTRSP_TBL::
                        ;
3F 0000 213                     .BYTE   ^B 111111
3E 0001 214                     .BYTE   ^B 111110
38 0002 215                     .BYTE   ^B 111000
34 0003 216                     .BYTE   ^B 110100
20 0004 217                     .BYTE   ^B 100000
00 0005 218                     .BYTE   ^B 000000
```

```
                                0006   220          .SBTTL  LCK$DISPATCH - Dispatch incoming lock message
                                0006   221
                                0006   222 ;++
                                0006   223 ; FUNCTIONAL DESCRIPTION:
                                0006   224 ;
                                0006   225 ;       This routine dispatches incoming lock manager messages tc
                                0006   226 ;       the appropriate routine.  It also verifies that incoming messages
                                0006   227 ;       are stamped with a membership sequence number that is the same as
                                0006   228 ;       the current one on this system.  If not, the messages are either
                                0006   229 ;       returned with a LKB$K_RETRY response (RSPID messages) or are ignored
                                0006   230 ;       (non-RSPID messages).
                                0006   231 ;
                                0006   232 ; CALLING SEQUENCE:
                                0006   233 ;
                                0006   234 ;       JSB     LCK$DISPATCH    (called from connection manager received
                                0006   235 ;                                message routine)
                                0006   236 ;       IPL is at IPL$_SYNCH
                                0006   237 ;
                                0006   238 ; INPUT PARAMETERS:
                                0006   239 ;
                                0006   240 ;       R2      Address of message
                                0006   241 ;       R3      Address of CSB
                                0006   242 ;       R4      Address of PDT
                                0006   243 ;       R5      Address of CDRP (if this message needs a response)
                                0006   244 ;
                                0006   245 ; OUTPUT PARAMETERS:
                                0006   246 ;
                                0006   247 ;       None
                                0006   248 ;
                                0006   249 ; SIDE EFFECTS:
                                0006   250 ;
                                0006   251 ;       R0 - R5 destroyed
                                0006   252 ;--
                                0006   253
                                0006   254 LCK$DISPATCH::
                                0006   255
      50    64 A3    D0        0006   256          MOVL    CSB$L_CLUB(R3),R0       ; Get address of CLUB
         00AC C0    A3        000A   257          SUBW3   CLUB$W_MEMSEQ(R0),-     ; Compare sequence numbers
      50    0C A2              000E   258                  LKMSG$W_MEMSEQ(R2),R0
            58   19            0011   259          BLSS    50$                     ; Message is old, ignore or retry
            2A   14            0013   260          BGTR    20$                     ; MEMSEQ must be in error!
50    00000000'GF    9A        0015   261          MOVZBL  G^LCK$GB_REBLD_STATE,R0 ; Are we in any phase of failover?
            25   12            001C   262          BNEQ    30$                     ; Yes
                               001E   263          DISPATCH        CLSMSG$B_FUNC(R2),TYPE=B,PREFIX=LKMSG$K_,-
                               001E   264                  <-
                               001E   265                  <NEWLOCK,LCK$RCV_LOCKREQ>,-     ; New lock request
                               001E   266                  <GRANTED,LCK$RCV_GRANTED>,-     ; Lock granted message
                               001E   267                  <DEQ,LCK$RCV_DEQS>,-            ; Dequeue request
                               001E   268                  <RMVDIR,LCK$RCV_RMVDIR>,-       ; Remove directory entry
                               001E   269                  <BLKING,LCK$RCV_BLKING>,-       ; Lock blocking message
                               001E   270                  <CVTLCKM,LCK$RCV_CVTMSG>,-      ; Convert lock message
                               001E   271                  <CVTLCKR,LCK$RCV_CVTREQ>,-      ; Convert lock req. (needs RSP)
                               001E   272                  <TSRQST,LCK$RCV_TIMESTAMP_RQST>,- ; Time stamp request
                               001E   273                  <SRCHDLCK,LCK$RCV_SRCHDLCK>,-   ; Search for deadlock
                               001E   274                  <DLCKFND,LCK$RCV_DLCKFND>,-     ; Deadlock found
                               001E   275                  <REDO_SRCH,LCK$RCV_REDO_SRCH>,- ; Redo deadlock search
                               001E   276                  >
```

```
                        003B   277
                        003B   278          BUG_CHECK       LOCKMGRERR,FATAL        ; Illegal function code
                        003F   279
                        003F   280 20$:      BUG_CHECK       LOCKMGRERR,FATAL        ; MEMSEQ error
                        0043   281
                        0043   282 30$:      ; We are in the middle of a membership state change.  Dispatch
                        0043   283          ; on phase of state change.
                        0043   284
                        0043   285          DISPATCH        R0,TYPE=B,-
                        0043   286                  <-
                        0043   287                  <2,35$>,-
                        0043   288                  <3,45$>,-
                        0043   289                  >
                        004B   290
                        004B   291          BUG_CHECK       LOCKMGRERR,FATAL        ; Unrecognized state or
                        004F   292                                                  ; no messages allowed in
                        004F   293                                                  ; this state
                        004F   294
                        004F   295 35$:      DISPATCH        CLSMSG$B_FUNC(R2),TYPE=B,PREFIX=LKMSG$K_,-
                        004F   296                  <-
                        004F   297                  <REBLDLOCK,LCK$RCV_LOCKREQ>,-   ; Rebuild lock (failover)
                        004F   298                  >
                        0056   299
                        0056   300          BUG_CHECK       LOCKMGRERR,FATAL        ; Illegal function code
                        005A   301
                        005A   302 45$:      DISPATCH        CLSMSG$B_FUNC(R2),TYPE=B,PREFIX=LKMSG$K_,-
                        005A   303                  <-
                        005A   304                  <GRANTED,LCK$RCV_GRANTED>,-     ; Lock granted message
                        005A   305                  <BLKING,LCK$RCV_BLKING>,-       ; Lock blocking message
                        005A   306                  >
                        0067   307
                        0067   308          BUG_CHECK       LOCKMGRERR,FATAL        ; Illegal function code
                        006B   309
                        006B   310 50$:      ; Message should either be ignored or returned with LKB$K_RETRY
                        006B   311          ; response (the latter if a RSPID is provided).
                        006B   312
            04 A2   D5  006B   313          TSTL    CLSMSG$L_RSPID(R2)      ; Is there a RSPID?
              03    12  006E   314          BNEQ    70$                     ; Yes
               FF8D'  31  0070   315          BRW     CNX$DEALL_MSG_BUF_CSB   ; No, deallocate message bfr and return
                        0073   316
                        0073   317 70$:      ; Send back LKB$K_RETRY response.
                        0073   318
      53    4C A3   D0  0073   319          MOVL    CSB$L_CSID(R3),R3       ; Get CSID
               FF86'  30  0077   320          BSBW    CNX$INIT_CDRP           ; Initialize CDRP
               FE 8F  90  007A   321          MOVB    #LKB$K_RETRY,-          ; Store retry response
               32 A5      007D   322                  CDRP$L_VAL2+2(R5)
          0A5A'CF     9E  007F   323          MOVAB   W^BLD_RSPMSG,-          ; Store address of message build routine
               4C A5      0083   324                  CDRP$L_MSGBLD(R5)
               FF78'  30  0085   325          BSBW    CNX$SEND_MSG_RESP       ; Send message response
           50    55   D0  0088   326          MOVL    R5,R0                   ; Address of CDRP
00000000'GF     17      008B   327          JMP     G^EXE$DEANONPAGED       ; Deallocate it and return
```

```
                          0091   329          .SBTTL  LCK$SND_CVTREQ - Send a conversion request to remote system
                          0091   330
                          0091   331 ;++
                          0091   332 ; FUNCTIONAL DESCRIPTION:
                          0091   333 ;
                          0091   334 ;       This routine handles conversion requests that must be forwarded
                          0091   335 ;       to the master system for this lock.  This system is the process
                          0091   336 ;       system.
                          0091   337 ;
                          0091   338 ; CALLING SEQUENCE:
                          0091   339 ;
                          0091   340 ;       JMP     LCK$SND_CVTREQ (This routine does not return to its
                          0091   341 ;                               caller.  Rather, it jumps to an
                          0091   342 ;                               appropriate destination.)
                          0091   343 ;       IPL must be at IPL$_SYNCH
                          0091   344 ;
                          0091   345 ; INPUT PARAMETERS:
                          0091   346 ;
                          0091   347 ;       R1      Requested lock mode
                          0091   348 ;       R3      CSID of destination system
                          0091   349 ;       R6      Address of LKB
                          0091   350 ;       R7      Old status from LKB
                          0091   351 ;       R8      Address of RSB
                          0091   352 ;       R9      Flags
                          0091   353 ;       R11     Current granted mode
                          0091   354 ;
                          0091   355 ; OUTPUT PARAMETERS:
                          0091   356 ;
                          0091   357 ;       R0      Completion code (on some exit paths)
                          0091   358 ;
                          0091   359 ; SIDE EFFECTS:
                          0091   360 ;
                          0091   361 ;       The process will go into MWAIT until the response from the remote
                          0091   362 ;       system arrives.
                          0091   363 ;       R1 - R5 destroyed
                          0091   364 ;--
                          0091   365
                          0091   366          .ENABL  LSB
                          0091   367
                          0091   368 LCK$SND_CVTREQ::
                          0091   369
                 00000002 0091   370          .IF NE  CA$_MEASURE
       00000000'GF   D6   0091   371          INCL    G^PMS$GL_ENQCVT_OUT
                          0097   372          .ENDC
                          0097   373
    54 00000000'GF   D0   0097   374          MOVL    G^SCH$GL_CURPCB,R4      ; Get our PCB address
         34 A6       51   90   009E   375          MOVB    R1,LKB$B_RQMODE(R6)     ; Store requested mode
33 FF58 CF4B         51   E0   00A2   376          BBS     R1,LCK$CVTRSP_TBL[R11],20$ ; Branch if response needed
                          00A9   377
                          00A9   378          ; No response needed.  Allocate CDRP, store value block (if
                          00A9   379          ; specified), and send message.
                          00A9   380
                          00A9   381          ASSUME  LCK$V_VALBLK  EQ  0
                          00A9   382
                FF54'     30   00A9   383          BSBW    CNX$ALLOC_CDRP          ; Alloc. a CDRP and convert CSID to CSB
              50 50       E9   00AC   384          BLBC    R0,25$                  ; No CDRPs or CSID error
              0A 59       E9   00AF   385          BLBC    R9,10$                  ; No value block
```

```
                28 A8    7D  00B2   386          MOVQ    RSB$Q_VALBLK(R8),-      ; Copy value block
                38 A5        00B5   387                  CDRP$[_VAL4(R5)
                30 A8    7D  00B7   388          MOVQ    RSB$Q_VALBLK+8(R8),-
                40 A5        00BA   389                  CDRP$[_VAL6(R5)
                60 A4    D0  00BC   390 10$:     MOVL    PCB$L_PID(R4),-         ; Store PID in CDRP
                48 A5        00BF   391                  CDRP$[_VAL8(R5)
                34 A6    B0  00C1   392          MOVW    LKB$B_RQMODE(R6),-      ; Store req. and granted mode in CDRP
                5C A5        00C4   393                  CDRP$[_VAL9(R5)
                   50    D4  00C6   394          CLRL    R0                      ; Initialize blocking AST flag
                20 A6    D5  00C8   395          TSTL    LKB$L_BLKASTADR(R6)     ; Test for a blocking AST address
                   02    13  00CB   396          BEQL    15$                     ; No blocking AST
                   50    D6  00CD   397          INCL    R0                      ; Set blocking AST flag
             5E A5 50    90  00CF   398 15$:     MOVB    R0,CDRP$L_VAL9+2(R5)    ; Store blocking AST flag
                50 56    D0  00D3   399          MOVL    R6,R0                   ; Move address of LKB
                   00DC  30  00D6   400          BSBW    SENDCVTMSG              ; Send message
                   0084  31  00D9   401          BRW     CGRNTD_PROC             ; Join granted code
                         00DC   402
                         00DC   403 20$:     ; Send a message and wait for response.
                         00DC   404
                   FF21' 30  00DC   405          BSBW    CNX$ALLOC_WARMCDRP      ; Alloc. a CDRP with RSPID and cvt CSID
                52 50    E9  00DF   406 25$:     BLBC    R0,60$                  ; No CDRPs or CSID error
             50 56       D0  00E2   407          MOVL    R6,R0                   ; Move address of LKB
          48 A5 54       D0  00E5   408          MOVL    R4,CDRP$L_VAL8(R5)      ; Store PCB address in CDRP
                   00F4  30  00E9   409          BSBW    SENDCVTREQ              ; Send convert request
          54    48 A5    D0  00EC   410          MOVL    CDRP$L_VAL8(R5),R4      ; Restore address of PCB
                         00F0   411
                         00F0   412 CVT_SCS_WAIT:
             50    0D    D0  00F0   413          MOVL    #RSN$_SCS,R0            ; Go into MWAIT for resource RSN$_SCS
                         00F3   414
                         00F3   415 CVT_WAIT:
                         00F3   416          ; Put the process into MWAIT until the response arrives or the
                         00F3   417          ; resource becomes available.  Resource number is in R0.
                         00F3   418          ; PCB address is in R4.  If this is a system owned lock,
                         00F3   419          ; then the process's PID is temporarily stored in
                         00F3   420          ; the LKB.  This serves two purposes.  First, it allows the process
                         00F3   421          ; to be made executable when the response arrives and secondly, it
                         00F3   422          ; prevents any other processes from manipulating this lock
                         00F3   423          ; (VERIFYLOCKID will fail) while we wait for the response.
                         00F3   424          ; Note that the only two conversions that require a response message
                         00F3   425          ; yet will allow a system lock are a CR to CR conversion or a NL to
                         00F3   426          ; NL conversion and that these are always synchronously grantable.
                         00F3   427          ; This also occurs if we go into MWAIT for non-paged pool.
                         00F3   428
                60 A4    D0  00F3   429          MOVL    PCB$L_PID(R4),-         ; Make sure PID is stored in case lock
                0C A6        00F6   430                  LKB$L_PID(R6)           ; is system owned
                   10    78  00F8   431          ASHL    #PSL$V_IPL,-            ; Create a PSL on stack with IPL
             7E    02        00FA   432                  #IPL$_ASTDEL,-(SP)      ; set to ASTDEL
00000000'GF      16    FC  00FC   433          JSB     G^SCH$RWAIT             ; Wait
                         0102   434
                         0102   435          ; Upon reawakening the LKB$B_STATE field tells us what action
                         0102   436          ; to perform.  But we first must determine if we are stalling requests.
                         0102   437
                         0102   438          SETIPL  #IPL$_SYNCH             ; Raise IPL
00000000'GF      95    0105   439          TSTB    G^LCK$GB_STALLREQS      ; Are we stalling?
                   1B    12  010B   440          BNEQ    40$                     ; Yes
          03 59    06  E1  010D   441 30$:     BBC     #LCK$V_CVTSYS,R9,33$    ; Branch if not system owned
                0C A6    D4  0111   442          CLRL    LKB$L_PID(R6)           ; Clear PID of system owned lock
```

```
                     0114    443
                     0114    444 33$:    DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
                     0114    445                 <-
                     0114    446                 <RSPGRANTD,CGRNTD_PROC>,-       ; Granted
                     0114    447                 <RSPQUEUED,CVTQED_PROC>,-       ; Queued
                     0114    448                 <RSPNOTQED,CNOTQD_PROC>,-       ; Not queued
                     0114    449                 <SCSWAIT,CVT_SCS_WAIT>,-        ; Spurious wakeup
                     0114    450                 <RETRY,CVTRETRY_PROC>,-         ; Retry
                     0114    451                 >
                     0124    452
                     0124    453                 BUG_CHECK       LOCKMGRERR,FATAL        ; Invalid lock state
                     0128    454
                     0128    455 40$:    ; We are stalling some or all lock requests.  See which ones.
                     0128    456
           05   19  0128    457                 BLSS    50$                     ; All requests are being stalled
           09   E1  012A    458                 BBC     #LKB$V_PROTECT,-        ; Only protected locks are being stalled
      DE 2A A6        012C    459                         LKB$W_STATUS(R6),30$    ; Branch if this is not a protected lock
       50  0E   D0  012F    460 50$:    MOVL    #RSN$_CLUSTRAN,R0               ; Resource to wait for
           BF   11  0132    461                 BRB     CVT_WAIT                ; Go into MWAIT for RSN$_CLUSTRAN
                0134    462
                0134    463 60$:    ; Either no memory or CSID to CSB conversion failed with
                0134    464         ; SS$_NOSUCHNODE.  Note that if we wait, system owned locks
                0134    465         ; must have a non-zero PID.
                0134    466
  0000'8F   50  B1  0134    467                 CMPW    R0,#SS$_INSFMEM         ; Is it insuff. memory?
           0A   12  0139    468                 BNEQ    70$                     ; No
         FE 8F  90  013B    469                 MOVB    #LKB$K_RETRY,-          ; Store lock state
        36 A6       013E    470                         LKB$B_STATE(R6)
       50   03  D0  0140    471                 MOVL    #RSN$_NPDYNMEM,R0       ; Resource to wait for
           AE   11  0143    472                 BRB     CVT_WAIT                ; Go into MWAIT for RSN$_NPDYNMEM
                0145    473
                0145    474 70$:    BUG_CHECK       LOCKMGRERR,FATAL; CSID conversion error
                0149    475
                0149    476         .DSABL  LSB
                0149    477
                0149    478 ;****************************************************************************
                0149    479 ;
                0149    480 ;       Process level routines for handling remote conversion requests
                0149    481 ;
                0149    482 ;****************************************************************************
                0149    483
                0149    484 CVTRETRY_PROC:
                0149    485         ; Retry this conversion due to the destination being removed
                0149    486         ; from the cluster or we were unable to allocate a CDRP.
                0149    487
      51   34 A6  9A  0149    488         MOVZBL  LKB$B_RQMODE(R6),R1     ; Get requested lock mode
      53   38 A8  D0  014D    489         MOVL    RSB$L_CSID(R8),R3       ; Get CSID of destination system
           0A   12  0151    490         BNEQ    10$                     ; Resend to destination
           01   90  0153    491         MOVB    #LKB$K_GRANTED,-        ; Store correct lock state
        36 A6       0155    492                 LKB$B_STATE(R6)
  00000000'GF   17  0157    493         JMP     G^LCK$LOCAL_CVT         ; Do it locally
         FF31   31  015D    494 10$:    BRW     LCK$SND_CVTREQ
                0160    495
                0160    496 CGRNTD_PROC:
                0160    497         ; Conversion request was granted
                0160    498
           03   E1  0160    499         BBC     #LKB$V_BLKASTQED,-      ; Branch if a blocking AST
```

```
                 0162   500                    LKB$W_STATUS(R6),20$    ; should not be queued
      02   A8    0165   501          BISW      #LKB$M_DBLKAST,-        ; Set deliver blocking AST status
   2A A6           0167   502                    LKB$W_STATUS(R6)
      10   88    0169   503          BISB      #LKB$M_PKAST,-          ; Set piggyback kernel AST bit
   0B A6           016B   504                    LKB$B_RMOD(R6)
   20 A6   D0    016D   505          MOVL      LKB$L_BLKASTADR(R6),-   ; Store address of blocking AST routine
   10 A6           0170   506                    LKB$L_AST(R6)
50 38 A6   0F    0172   507 20$:     REMQUE    LKB$L_SQFL(R6),R0       ; Remove lock from RSB's granted queue
   34 A6   90    0176   508          MOVB      LKB$B_RQMODE(R6),-      ; Set granted mode
   35 A6           0179   509                    LKB$B_GRMODE(R6)
      00'  D0    017B   510          MOVL      S^#SS$_NORMAL,-         ; Set status for lock status block
   2C A6           017D   511                    LKB$L_LKST1(R6)
   20 A6   D5    017F   512          TSTL      LKB$L_BLKASTADR(R6)     ; Blocking AST specified?
      03   13    0182   513          BEQL      40$                     ; No
   42 A8   B6    0184   514          INCW      RSB$W_BLKASTCNT(R8)     ; Yes, adjust count
00000000'GF   16    0187   515 40$:     JSB       G^LCK$GRANT_REM         ; Finish granting the lock
00000000'GF   17    018D   516          JMP       G^LCK$CVT_GRANTED       ; Return value block and exit
                 0193   517
                 0193   518 CVTQED_PROC:
                 0193   519          ; Conversion was queued
                 0193   520
50 38 A6   0F    0193   521          REMQUE    LKB$L_SQFL(R6),R0       ; Remove lock from granted queue
00000000'GF   16    0197   522          JSB       G^LCK$QUEUECVT          ; Queue it to conversion queue
00000000'GF   17    019D   523          JMP       G^LCK$QUEUED_EXIT
                 01A3   524
                 01A3   525 CNOTQD_PROC:
                 01A3   526          ; Conversion was not queued.  However, we have to handle the case
                 01A3   527          ; where we receive a blocking AST message before we get here.
                 01A3   528
      03   E1    01A3   529          BBC       #LKB$V_BLKASTQED,-      ; Branch if we didn't get a blocking AST
03 2A A6           01A5   530                    LKB$W_STATUS(R6),60$
57    02   A8    01A8   531          BISW      #LKB$M_DBLKAST,R7       ; Set deliver BLKAST bit in old status
      01   90    01AB   532 60$:     MOVB      #LKB$K_GRANTED,-        ; Store correct lock state
   36 A6           01AD   533                    LKB$B_STATE(R6)
00000000'GF   17    01AF   534          JMP       G^LCK$CVTNOTQED
```

```
                                        .SBTTL  SENDCVTREQ - Send convert request to remote system

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

```
                34 A5          01EC   593                  CDRP$L_VAL3(R5)
            02D0'CF    9E      01EE   594          MOVAB   W^BLD_CVTMSGR,-          ; Store address of message build
                4C A5          01F2   595                  CDRP$L_MSGBLD(R5)       ; routine
                FD 8F    90    01F4   596          MOVB    #LKB$K_SCSWAIT,-         ; Store lock state
                36 A0          01F7   597                  LKB$B_STATE(R0)
                 FE04'   30    01F9   598          BSBW    CNX$SEND_MSG_CSB         ; Send the message
                39 50    E9    01FC   599          BLBC    R0,CVT_RETRY             ; Dest. system is no longer in cluster
                               01FF   600
                               01FF   601          ; We are resumed here when the response message arrives.
                               01FF   602          ; Registers contain:
                               01FF   603          ;       R2      Address of message buffer
                               01FF   604          ;       R3      CSB address
                               01FF   605          ;       R4      Address of PDT
                               01FF   606          ;       R5      Address of CDRP
                               01FF   607          ; First validate membership sequence number.  I.e. ignore response
                               01FF   608          ; and retry request if we did a membership state change while
                               01FF   609          ; this message was in transit.  Otherwise store response code
                               01FF   610          ; in LKB and then dispatch on it.
                               01FF   611
          50    64 A3    D0    01FF   612          MOVL    CSB$L_CLUB(R3),R0        ; Get address of CLUB
             00AC C0    B1    0203   613          CMPW    CLUB$W_MEMSEQ(R0),-      ; Compare sequence numbers
                0C A2          0207   614                  LKMSG$W_MEMSEQ(R2)
                    05   13    0209   615          BEQL    20$                      ; They match
                FE 8F    90    020B   616          MOVB    #LKB$K_RETRY,-           ; No match - change state to RETRY
                1E A2          020E   617                  LKMSG$B_STATE(R2)
                               0210   618 20$:
          50    2C A5    D0    0210   619          MOVL    CDRP$L_VAL1(R5),R0       ; R0 = LKB address
                36 A0    91    0214   620          CMPB    LKB$B_STATE(R0),-        ; Make sure LKB is in SCSWAIT state
                FD 8F          0217   621                  #LKB$K_SCSWAIT
                    19   12    0219   622          BNEQ    30$                      ; It's not
                1E A2    90    021B   623          MOVB    LKMSG$B_STATE(R2),-      ; Store state
                36 A0          021E   624                  LKB$B_STATE(R0)
                               0220   625
                               0220   626          DISPATCH        LKB$B_STATE(R0),TYPE=B,PREFIX=LKB$K_,-
                               0220   627                  <-
                               0220   628                  <RETRY,DEALL_WARMCDRP>,-         ; Retry request
                               0220   629                  <RSPGRANTD,CGRNTD_FORK>,-        ; Granted
                               0220   630                  <RSPQUEUED,CVTQED_FORK>,-        ; Queued
                               0220   631                  <RSPNOTQED,CNOTQD_FORK>,-        ; Not queued
                               0220   632                  >
                               0230   633
                               0230   634          BUG_CHECK       LOCKMGRERR,FATAL         ; Invalid lock state
                               0234   635
                               0234   636 30$:     BUG_CHECK       LOCKMGRERR,FATAL         ; LKB not in SCSWAIT state
                               0238   637
                               0238   638 CVT_RETRY
                               0238   639          ; Remote system is being removed from cluster
                               0238   640
          54    2C A5    D0    0238   641          MOVL    CDRP$L_VAL1(R5),R4       ; Get LKB address
                FE 8F    90    023C   642          MOVB    #LKB$K_RETRY,-           ; Store retry state
                36 A4          023F   643                  LKB$B_STATE(R4)
             50    55    D0    0241   644          MOVL    R5,R0                    ; Address of CDRP
00000000'GF    16    0244   645          JSB     G^EXE$DEANONPAGED        ; Deallocate CDRP
                 0496   31    024A   646          BRW     WAKE_PROCESS             ; Wake process
                               024D   647
                               024D   648 CNOTQD_FORK:
                               024D   649 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$V_VALINVLD,-       ; and conditionally set it
  04 1A A2           026E   667                  LKMSG$Q_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$Q_LKBSTATUS(R2),50$
         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   686 ;*****************************************************************************
                     0289   687 ;
                     0289   688 ;               Build convert message action routines
                     0289   689 ;
                     0289   690 ;*****************************************************************************
                     0289   691
                     0289   692         ; Action routine to build the actual convert lock request message
                     0289   693         ; Inputs are:
                     0289   694         ;       R2      Address of message buffer
                     0289   695         ;       R5      Address of CDRP
                     0289   696         ; NOTE: CDRP$L_VAL contains a PID for BLD_CVTMSG and a PCB
                     0289   697         ;       address for BLD_CVTMSGR.  This is because the PCB address
                     0289   698         ;       must be revalidated for the former case as the process
                     0289   699         ;       can get deleted before the message build routine is called.
                     0289   700         ; All registers except R0 and R1 must be preserved
                     0289   701
                     0289   702 BLD_CVTMSG:
                     0289   703         ; This entry point is for conversions that don't require a response
                     0289   704
                     0289   705         ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
                     0289   706         ASSUME  LKMSG$L_PRCLKID  EQ  LKMSG$L_MSTLKID+4
                     0289   707         ASSUME  LKMSG$L_DLCKPRI_CVT  EQ  LKMSG$L_EPIDCVT+4
                     0289   708
08 A2    0602 8F     B0   0289   709         MOVW    #LKMSG$K_CVTLCKMSG-     ; Store function and facility codes
                     028F   710                 !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
              2C A5  7D   028F   711         MOVQ    CDRP$L_VAL(R5),-        ; Store master and process lockids
              10 A2       0292   712                 LKMSG$L_MSTLKID(R2)     ; in message
              34 A5  B0   0294   713         MOVW    CDRP$L_VAL3(R5),-       ; Store memseq in message
              0C A2       0297   714                 LKMSG$W_MEMSEQ(R2)
              36 A5  B0   0299   715         MOVW    CDRP$L_VAL3+2(R5),-     ; Store flags in message
              18 A2       029C   716                 LKMSG$W_FLAGS(R2)
                          029E   717         ASSUME  LCK$V_VALBLK  EQ  0
           0A 18 A2  E9   029E   718         BLBC    LKMSG$W_FLAGS(R2),10$   ; Branch if no value block specified
              38 A5  7D   02A2   719         MOVQ    CDRP$L_VAL4(R5),-       ; Store value block in message
              20 A2       02A5   720                 LKMSG$Q_VALBLK(R2)
              40 A5  7D   02A7   721         MOVQ    CDRP$L_VAL6(R5),-
              28 A2       02AA   722                 LKMSG$Q_VALBLK+8(R2)
              5C A5  9B   02AC   723 10$:    MOVZBW  CDRP$L_VAL9(R5),-       ; Store requested mode, and clear
              1A A2       02AF   724                 LKMSG$B_RQMODE(R2)      ; granted mode
              5E A5  9A   02B1   725         MOVZBL  CDRP$L_VAL9+2(R5),-     ; Store blocking AST flag
              1C A2       02B4   726                 LKMSG$L_BLKASTFLG(R2)
        51    48 A5  3C   02B6   727         MOVZWL  CDRP$L_VAL8(R5),R1      ; Get PID index
50    00000000'GF  D0   02BA   728         MOVL    G^SCH$GL_PCBVEC,R0      ; Get address of PCB vector
        50    6041  D0   02C1   729         MOVL    (R0)[R1],R0             ; Get address of PCB
              48 A5  D1   02C5   730         CMPL    CDRP$L_VAL8(R5),-       ; Validate PCB
              60 A0       02C8   731                 PCB$L_PID(R0)
                 2E  13   02CA   732         BEQL    BLD_CVTCOM              ; It's ok
              30 A2  7C   02CC   733         CLRQ    LKMSG$L_EPIDCVT(R2)     ; Process was deleted; clear
                 05  02CF   734         RSB                             ; EPID and deadlock priority fields
                     02D0   735
                     02D0   736 BLD_CVTMSGR:
                     02D0   737         ; This entry point is for conversions that require a response
                     02D0   738
                     02D0   739         ASSUME  LKMSG$L_MSTLKID  EQ  LKMSG$W_MEMSEQ+4
                     02D0   740         ASSUME  LKMSG$L_PRCLKID  EQ  LKMSG$L_MSTLKID+4
                     02D0   741         ASSUME  LKMSG$W_FLAGS  EQ  LKMSG$L_PRCLKID+4
                     02D0   742         ASSUME  LKMSG$B_RQMODE  EQ  LKMSG$Q_FLAGS+2
```

```
                                02D0    743          ASSUME  LKMSG$L_BLKASTFLG  EQ  LKMSG$B_RQMODE+2
                                02D0    744          ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
                                02D0    745
        08 A2   0702 8F   B0    02D0    746          MOVW    #LKMSG$K_CVTLCKR@8-     ; Store function and facility codes
                                02D6    747                  !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
             50   2C A5   D0    02D6    748          MOVL    CDRP$L_VAL1(R5),R0      ; Get LKB address
             51   0C A2   3E    02DA    749          MOVAW   LKMSG$W_MEMSEQ(R2),R1   ; Point into message buffer
             81   34 A5   3C    02DE    750          MOVZWL  CDRP$L_VAL3(R5),(R1)+   ; Store seq. number
             81   54 A0   D0    02E2    751          MOVL    LKB$L_REMLKID(R0),(R1)+ ; Store master lockid
             81   30 A0   D0    02E6    752          MOVL    LKB$L_LKID(R0),(R1)+    ; Store process lockid
             81   28 A0   B0    02EA    753          MOVW    LKB$W_FLAGS(R0),(R1)+   ; Store flags
             81   34 A0   9B    02EE    754          MOVZBW  LKB$B_RQMODE(R0),(R1)+  ; Store req. mode and clear granted mode
             81   20 A0   D0    02F2    755          MOVL    LKB$L_BLKASTADR(R0),(R1)+ ; Store blocking AST flag
             50   48 A5   D0    02F6    756          MOVL    CDRP$L_VAL8(R5),R0      ; Get PCB address
                                02FA    757
                                02FA    758 BLD_CVTCOM:
                  64 A0   D0    02FA    759          MOVL    PCB$L_EPID(R0),-        ; Store EPID
                  30 A2         02FD    760                  LKMSG$L_EPIDCVT(R2)
                 010C C0   D0   02FF    761          MOVL    PCB$L_DLCKPRI(R0),-     ; Store deadlock priority
                  34 A2         0303    762                  LKMSG$L_DLCKPRI_CVT(R2)
                        05      0305    763          RSB
```

```
                     0306   765          .SBTTL  LKC$RCV_CVTREQ - Receive convert request
                     0306   766
                     0306   767 ;++
                     0306   768 ; FUNCTIONAL DESCRIPTION:
                     0306   769 ;
                     0306   770 ;       This routine receives conversion requests from the remote system.
                     0306   771 ;       The remote system is the process system and we are the master
                     0306   772 ;       system for this lock.  The conversion request is performed and
                     0306   773 ;       a response message is returned if called at entry point
                     0306   774 ;       LCK$RCV_CVTREQ.
                     0306   775 ;
                     0306   776 ; CALLING SEQUENCE:
                     0306   777 ;
                     0306   778 ;       JSB     LCK$RCV_CVTREQ (called from SCS received message routine)
                     0306   779 ;       IPL must be at IPL$_SYNCH
                     0306   780 ;
                     0306   781 ; INPUT PARAMETERS:
                     0306   782 ;
                     0306   783 ;       R2      Address of message buffer
                     0306   784 ;       R3      Address of CSB
                     0306   785 ;       R5      Address of CDRP if a response is needed
                     0306   786 ;
                     0306   787 ; OUTPUT PARAMETERS:
                     0306   788 ;
                     0306   789 ;       None
                     0306   790 ;
                     0306   791 ; SIDE EFFECTS:
                     0306   792 ;
                     0306   793 ;       A message may be sent back as a response
                     0306   794 ;       R0 - R5 destroyed
                     0306   795 ;--
                     0306   796
                     0306   797          ASSUME  LKMSG$L_PRCLKID  EQ  LKMSG$L_MSTLKID+4
                     0306   798          ASSUME  LKMSG$W_FLAGS  EQ  LKMSG$L_PRCLKID+4
                     0306   799          ASSUME  LKMSG$B_RQMODE  EQ  LKMSG$W_FLAGS+2
                     0306   800          ASSUME  LKMSG$L_BLKASTFLG  EQ  LKMSG$B_RQMODE+2
                     0306   801          ASSUME  LKMSG$Q_VALBLK  EQ  LKMSG$L_BLKASTFLG+4
                     0306   802
                     0306   803 CVT_INVLKID:
                     0306   804          BUG_CHECK       LOCKMGRERR,FATAL
                     030A   805
                     030A   806 LCK$RCV_CVTMSG::
                55   D4   030A   807          CLRL    R5                      ; Indicate no CDRP
                     030C   808
                     030C   809 LCK$RCV_CVTREQ::
                     030C   810
            00000002  030C   811          .IF NE  CA$_MEASURE
00000000'GF     D6   030C   812          INCL    G^PMS$GL_ENQCVT_IN
                     0312   813          .ENDC
                     0312   814
          0FCC 8F    BB   0312   815          PUSHR   #^M<R2,R3,R6,R7,R8,R9,R10,R11>
               55    DD   0316   816          PUSHL   R5                      ; Save CDRP address or 0
               03    13   0318   817          BEQL    5$                      ; No CDRP
           FCE3'     30   031A   818          BSBW    CNX$INIT_CDRP           ; Initialize CDRP
57    52     10    C1   031D   819 5$:      ADDL3   #LKMSG$L_MSTLKID,R2,R7  ; Point R7 into message buffer
      51     87    D0   0321   820          MOVL    (R7)+,R1                ; Get master lockid (id on this system)
      50     87    D0   0324   821          MOVL    (R7)+,R0                ; Get process lockid (for verify)
```

```
                 0CAD   30  0327   822          BSBW    VERIFYREMLKID           ; Convert to LKB address
               D9 50    E9  032A   823          BLBC    R0,CVT_INVLKID          ; Invalid id
                                   032D   824
                                   032D   825          ; Have LKB address in R6, pointer into message in R7.  Set
                                   032D   826          ; up other registers as needed.
                                   032D   827
            58   50 A6   D0  032D   828          MOVL    LKB$L_RSB(R6),R8        ; Get RSB address
               59   87   3C  0331   829          MOVZWL  (R7)+,R9                ; Get flags
            28 A6   59   B0  0334   830          MOVW    R9,LKB$W_FLAGS(R6)      ; Store them
                 2A A6   DD  0338   831          PUSHL   LKB$W_STATUS(R6)        ; Save old status bits
                         AA  033B   832          BICW    #LKB$M_DCPLAST-         ; Clear relevant status bits
                                    033C   833                  !LKB$M_DBLKAST-    ; The following bits retain their
                                    033C   834                  !LKB$M_ASYNC-      ; old state:
                                    033C   835                  !LKB$M_BLKASTQED-  ;       MSTCPY
                                    033C   836                  !LKB$M_TIMOUTQ-    ;       NOQUOTA (should not be set)
                                    033C   837                  !LKB$M_WASSYSOWN-  ;       PROTECT
                                    033C   838                  !LKB$M_CVTTOSYS,-
      2A A6   01CF   8F             033C   839                  LKB$W_STATUS(R6)
                 51   87   9A  0341   840          MOVZBL  (R7)+,R1                ; Get requested mode, skip granted
                      57   D6  0344   841          INCL    R7                      ; mode field
               5B   35 A6   9A  0346   842          MOVZBL  LKB$B_GRMODE(R6),R11    ; Get granted mode
                                   034A   843
                                   034A   844          ; Store EPID of owner of lock unless LCK$M_CVTSYS is set in
                                   034A   845          ; which case, store 0.  R7 points to BLKASTFLG field of message.
                                   034A   846          ; Also store deadlock priority in LKB if not system owned.
                                   034A   847
                      50   D4  034A   848          CLRL    R0                      ; Assume system owned
            09 59      06   E0  034C   849          BBS     #LCK$V_CVTSYS,R9,8$     ; Branch if it is system owned
            50   14 A7   D0  0350   850          MOVL    LKMSG$L_EPIDCVT-LKMSG$L_BLKASTFLG(R7),R0  ; Get EPID
                 18 A7   D0  0354   851          MOVL    LKMSG$L_DLCKPRI_CVT-LKMSG$L_BLKASTFLG(R7),-
                 24 A6       0357   852                  LKB$L_DLCKPRI(R6)       ; Store deadlock priority
            14 A6   50   D0  0359   853 8$:      MOVL    R0,LKB$L_EPID(R6)       ; Store EPID or 0
                                   035D   854
                                   035D   855          ; Decrement blocking AST count if a blocking AST was specified
                                   035D   856          ; and then store new blocking AST flag.
                                   035D   857          ; NOTE: the store of the new flag MUST be done after the test
                                   035D   858          ; of the old flag and decrement of the count.
                                   035D   859
                 20 A6   D0  035D   860          MOVL    LKB$L_BLKASTADR(R6),-   ; Save old blocking AST flag
                 5C A6       0360   861                  LKB$L_OLDBLKAST(R6)
                      03   13  0362   862          BEQL    10$                     ; No
                   42 A8   B7  0364   863          DECW    RSB$W_BLKASTCNT(R8)     ; Yes, adjust count
            20 A6   87   D0  0367   864 10$:      MOVL    (R7)+,LKB$L_BLKASTADR(R6) ; Store new flag
                                   036B   865
                                   036B   866          ; If we are converting down (or same) from PW or EX then copy
                                   036B   867          ; the caller's value block (if specified) into the RSB.
                                   036B   868          ; R7 points to value block in message.
                                   036B   869
                                   036B   870          ASSUME  LCK$V_VALBLK  EQ  0
                                   036B   871
                 19 59   E9  036B   872          BLBC    R9,20$                  ; Branch if no value block specified
               04   5B   91  036E   873          CMPB    R11,#LCK$K_PWMODE       ; Is grmode mode PW or EX?
                      14   1F  0371   874          BLSSU   20$                     ; No
               5B   51   91  0373   875          CMPB    R1,R11                  ; Is rqmode greater than grmode?
                      0F   1A  0376   876          BGTRU   20$                     ; Yes
            28 A8   87   7D  0378   877          MOVQ    (R7)+,RSB$Q_VALBLK(R8)  ; Copy caller's value block to RSB
            30 A8   67   7D  037C   878          MOVQ    (R7),RSB$Q_VALBLK+8(R8)
```

```
                3C A8   D6  0380   879          INCL    RSB$L_VALSEQNUM(R8)     ; Increment value block seq. number
                   02   AA  0383   880          BICW    #RSB$M_VALINVLD,-       ; Clear invalid bit
                0E A8           0385   881                  RSB$W_STATUS(R8)
                                0387   882
                                0387   883 20$:    ; Remove this lock from the granted queue.  If it was the only one and
                                0387   884          ; if the conversion queue is also empty, then the conversion request
                                0387   885          ; can be granted immediately.  This path is special cased because it
                                0387   886          ; is the normal case.
                                0387   887
          50    38 A6   0F    0387   888          REMQUE  LKB$L_SQFL(R6),R0       ; Remove lock from granted queue
                   1F   12    038B   889          BNEQ    40$                     ; Not the only one
          5A    18 A8   DE    038D   890          MOVAL   RSB$L_CVTQFL(R8),R10    ; It's the only granted lock
             5A    6A   D1    0391   891          CMPL    (R10),R10               ; Is conversion queue empty?
                   16   12    0394   892          BNEQ    40$                     ; It's not - must check the longer way
00000000'GF        16   16    0396   893          JSB     G^LCK$GRANT_LOCK_ALT    ; It is - grant lock
             5A    08   C0    039C   894          ADDL    #8,R10                  ; Point to wait queue
             5A    6A   D1    039F   895          CMPL    (R10),R10               ; Is wait queue empty?
                   46   13    03A2   896          BEQL    60$                     ; Yes, exit
00000000'GF        16   16    03A4   897          JSB     G^LCK$GRANTWTRS         ; Try granting waiting locks
                   3E   11    03AA   898          BRB     60$
                                03AC   899
                                03AC   900 40$:    ; There was at least one other holder of the resource so we have
                                03AC   901          ; to check for compatibility the longer way.  The granted mode
                                03AC   902          ; of this lock is compared with the conversion grant mode.  If,
                                03AC   903          ; other than the head of the conversion queue, there are granted
                                03AC   904          ; locks with higher lock modes than this lock, then there
                                03AC   905          ; is no need to recompute the group grant mode or attempt to
                                03AC   906          ; grant waiting conversions.
                                03AC   907
          0D A8    5B   91    03AC   908          CMPB    R11,RSB$B_CGMODE(R8)    ; Is granted mode = conv. grant mode?
                   15   13    03B0   909          BEQL    45$                     ; Yes
             55    0C A8   9A    03B2   910          MOVZBL  RSB$B_GGMODE(R8),R5     ; No, get group grant mode
54 00000000'GF45    51   E1    03B6   911          BBC     R1,G^LCK$COMPAT_TBL[R5],80$ ; Branch if not compatible
          00000000'GF        16   03BF   912          JSB     G^LCK$GRANT_LOCK        ; Grant the lock
                   23   11    03C5   913          BRB     60$                     ; Exit with completion status in R0
                                03C7   914
          00000000'GF        16   03C7   915 45$:    JSB     G^LCK$COMP_GGMODE       ; Compute new group grant mode in R5
3D 00000000'GF45    51   E1    03CD   916          BBC     R1,G^LCK$COMPAT_TBL[R5],80$ ; Branch if not compatible
          0C A8    55   90    03D6   917          MOVB    R5,RSB$B_GGMODE(R8)     ; Store group grant mode in RSB
          0D A8    55   90    03DA   918          MOVB    R5,RSB$B_CGMODE(R8)     ; Also store conversion grant mode
          00000000'GF        16   03DE   919          JSB     G^LCK$GRANT_LOCK        ; Grant lock
          00000000'GF        16   03E4   920          JSB     G^LCK$GRANTCVTS         ; Try granting conversions and waiters
                                03EA   921
                                03EA   922 60$:    ; Set response state to be granted
                                03EA   923
             5E    04   C0    03EA   924          ADDL    #4,SP                   ; Remove old status from stack
                   02   AA  03ED   925          BICW    #LKB$M_DBLKAST,-        ; Got set spuriously by common code
                      2A A6        03EF   926                  LKB$W_STATUS(R6)
          50    FA 8F   90    03F1   927          MOVB    #LKB$K_RSPGRANTD,R0     ; Set state
                55 8ED0  03F5   928          POPL    R5                      ; Get response CDRP or 0
                   58   12    03F8   929          BNEQ    SEND_CVTRSP             ; Have a CDRP - send response
                                03FA   930
                                03FA   931 70$:    ; Since a response is not required, we have to deallocate the
                                03FA   932          ; message buffer.  However, it may be necessary to send a blocking
                                03FA   933          ; message.
                                03FA   934
                   0C   BA    03FA   935          POPR    #^M<R2,R3>              ; Get buffer and CSB addresses
```

```
                FC01'  30  03FC   936          BSBW    CNX$DEALL_MSG_BUF_CSB   ; Deallocate message buffer
                  03   E1  03FF   937          BBC     #LKB$V_BLKASTQED,-      ; Branch if no blocking AST queued
              0A 2A A6         0401   938                  LKB$W_STATUS(R6),75$
           53    58 A6   D0   0404   939          MOVL    LKB$L_CSID(R6),R3       ; Get CSID of other system
              54    56   D0   0408   940          MOVL    R6,R4                   ; Move LKB address
                 090F   30   040B   941          BSBW    SEND_BLOCKING           ; Send blocking message
              0FC0 8F   BA   040E   942 75$:     POPR    #^M<R6,R7,R8,R9,R10,R11>
                        05   0412   943          RSB
                             0413   944
                             0413   945 80$:     ; The conversion cannot be granted.  Queue the request
                             0413   946          ; unless the noqueue bit is set.  If the conversion queue is empty
                             0413   947          ; then R5 contains the new conversion grant mode.
                             0413   948
                    57 8ED0  0413   949          POPL    R7                      ; Restore old LKB status bits
           18 59      02   E0  0416   950          BBS     #LCK$V_NOQUEUE,R9,85$   ; Branch if noqueue is set
                             041A   951
                             041A   952          ; Queue the conversion
                             041A   953
           34 A6      51   90  041A   954          MOVB    R1,LKB$B_RQMODE(R6)     ; Store requested mode
                 46 A8   B0   041E   955          MOVW    RSB$W_RQSEQNM(R8),-     ; Store request sequence number
                 10 A6        0421   956                  LKB$W_RQSEQNM(R6)
                 46 A8   B6   0423   957          INCW    RSB$W_RQSEQNM(R8)       ; Increment sequence number
        00000000'GF  16   0426   958          JSB     G^LCK$QUEUECVT          ; Insert onto conversion queue, etc.
           50    FB 8F   90   042C   959          MOVB    #LKB$K_RSPQUEUED,R0     ; Set response state
                    1D   11   0430   960          BRB     98$                     ; Send response
                             0432   961
                             0432   962 85$:     ; The request is not to be queued.  Insert back onto th
                             0432   963          ; granted queue.
                             0432   964
                   00000002  0432   965          .IF NE  CA$_MEASURE
        00000000'GF      D6  0432   966          INCL    G^PMS$GL_ENQNOTQD
                             0438   967          .ENDC
                             0438   968
           2A A6      57   B0  0438   969          MOVW    R7,LKB$W_STATUS(R6)     ; Put back old status bits
                 38 A6   0E   043C   970          INSQUE  LKB$L_SQFL(R6),-        ; Put lock back on granted queue
                 10 A8        043F   971                  RSB$L_GRQFL(R8)
                 5C A6   D0   0441   972          MOVL    LKB$L_OLDBLKAST(R6),-   ; Restore old blocking AST flag
                 20 A6        0444   973                  LKB$L_BLKASTADR(R6)
                    03   13   0446   974          BEQL    95$                     ; No
                 42 A8   B6   0448   975          INCW    RSB$W_BLKASTCNT(R8)     ; Incr. blocking AST count
           50    FC 8F   90   044B   976 95$:     MOVB    #LKB$K_RSPNOTQLD,R0     ; Set response state
                    55 8ED0  044F   977 98$:     POPL    R5                      ; Restore CDRP address
                             0452   978
                             0452   979 SEND_CVTRSP:
                             0452   980          ; Send response message.  CDRP address is in R5.
                             0452   981
           2C A5      56   D0  0452   982          MOVL    R6,CDRP$L_VAL1(R5)      ; Store LKB address
           32 A5      50   90  0456   983          MOVB    R0,CDRP$L_VAL2+2(R5)    ; Store response state
                 88'AF   9E   045A   984          MOVAB   B^BLD_CVTRSP,-          ; Store address of message build
                 4C A5        045D   985                  CDRP$L_MSGBLD(R5)       ; routine
              0FCC 8F   BA   045F   986          POPR    #^M<R2,R3,R6,R7,R8,R9,R10,R11>
           51    64 A3   D0   0463   987          MOVL    CSB$L_CLUB(R3),R1       ; Get address of CLUB
              00AC C1   B0   0467   988          MOVW    CLUB$W_MEMSEQ(R1),-     ; Store membership seq. number
                 34 A5        046B   989                  CDRP$L_VAL3(R5)
           53    4C A3   D0   046D   990          MOVL    CSB$L_CSID(R3),R3       ; Get CSID out of CSB
                  FB8C'  30   0471   991          BSBW    CNX$SEND_MSG_RESP       ; Send message response
        0000'8F    50   B1   0474   992          CMPW    R0,#SS$_NOSUCHNODE      ; Is status SS$_NOSUCHNODE?
```

```
                    09   13  0479   993          BEQL    10$                     ; Yes, error
                 50 55   D0  047B   994          MOVL    R5,R0                   ; No, SS$_NODELEAVE is okay
           00000000'GF   17  047E   995          JMP     G^EXE$DEANONPAGED       ; Deallocate CDRP
                         0484   996
                         0484   997 10$:     BUG_CHECK       LOCKMGRERR,FATAL; CSID invalid
                         0488   998
                         0488   999 BLD_CVTRSP:
                         0488  1000          ; This routine builds a response message for conversion requests
                         0488  1001          ; Inputs:
                         0488  1002          ;       R2      Address of message buffer
                         0488  1003          ;       R3      Address of CSB
                         0488  1004          ;       R5      Address of CDRP
                         0488  1005          ; All registers except R0 and R1 must be preserved
                         0488  1006          ;
                         0488  1007          ; Note:  This message build routine uses pointers (to LKB and RSB)
                         0488  1008          ; that will not be valid if we started a state change.  Therefore,
                         0488  1009          ; it is necessary to validiate the membership sequence number
                         0488  1010          ; before using any pointers stored in the CDRP.
                         0488  1011
                         0488  1012          ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
                         0488  1013
        08 A2    0782 8F   B0  0488  1014          MOVW    #LKMSG$K_CVTLCKR@8-     ; Store function and facility codes
                               048E  1015                  !CLSMSG$M_RESPMSG-
                               048E  1016                  !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
              34 A5    B0      048E  1017          MOVW    CDRP$L_VA[3(R5),-       ; Store membership seq. num.
              0C A2            0491  1018                  LKMSG$Q_MEMSEQ(R2)
           51 64 A3    D0      0493  1019          MOVL    CSB$L_C[UB(R3),R1       ; Get address of CLUB
              00AC C1  31      0497  1020          CMPW    CLUB$Q_MEMSEQ(R1),-     ; Have we done a state change?
              34 A5            049B  1021                  CDRP$L_VAL3(R5)
                 3D    12      049D  1022          BNEQ    20$                     ; Yes
           50 2C A5    D)      049F  1023          MOVL    CDRP$L_VAL1(R5),R0      ; Get address of LKB
              30 A0    D)      04A3  1024          MOVL    LKB$L_[KID(R0),-        ; Store master lockid
              10 A2            04A6  1025                  LKMSG$L_MSTLKID(R2)
              54 A0    D(      04A8  1026          MOVL    LKB$L_REMLKID(R0),-     ; Store process lockid
              14 A2            04AB  1027                  LKMSG$L_PRCLKID(R2)
              32 A5    9C      04AD  1028          MOVB    CDRP$L_VAL2+2(R5),-     ; Store response state
              1E A2            04B0  1029                  LKMSG$B_STATE(R2)
              36 A0    9'      04B2  1030          CMPB    LKB$B_STATE(R0),-       ; Is lock state = GRANTED
                 01            04B5  1031                  #LKB$K_GRANTED
                 1E    1.      04B6  1032          BNEQ    10$                     ; No
              2A A0    B(      04B8  1033          MOVW    LKB$W_STATUS(R0),-      ; Yes, store LKB status
              18 A2            04BB  1034                  LKMSG$W_LKBSTATUS(R2)
           50 50 A0    D0      04BD  1035          MOVL    LKB$L_RSB(R0),R0        ; Get RSB address
              28 A0    7D      04C1  1036          MOVQ    RSB$Q_VALBLK(R0),-      ; Store value block
              20 A2            04C4  1037                  LKMSG$Q_VALBLK(R2)
              30 A0    7D      04C6  1038          MOVQ    RSB$Q_VALBLK+8(R0),-
              28 A2            04C9  1039                  LKMSG$Q_VALBLK+8(R2)
              3C A0    D0      04CB  1040          MOVL    RSB$L_VALSEQNUM(R0),-   ; Store value block seq. number
              30 A2            04CE  1041                  LKMSG$L_VALSEQNUM(R2)
              0E A0    B0      04D0  1042          MOVW    RSB$W_STATUS(R0),-      ; Store RSB status
              1A A2            04D3  1043                  LKMSG$W_RSBSTATUS(R2)
                       05      04D5  1044          RSB
                               04D6  1045
              10 A0    B0      04D6  1046 10$:     MOVW    LKB$W_RQSEQNM(R0),-     ; Store request sequence number
              30 A2            04D9  1047                  LKMSG$W_RQSEQNM(R2)     ; (not used if state = RSPCNOTQD)
                       05      04DB  1048          RSB
                               04DC  1049
```

```
                                 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  1058          .SBTTL  LCK$SND_LOCKREQ - Send lock request to remote system
                        04E2  1059
                        04E2  1060 ;++
                        04E2  1061 ; FUNCTIONAL DESCRIPTION:
                        04E2  1062 ;
                        04E2  1063 ;       This routine handles lock requests that must be forwarded
                        04E2  1064 ;       to the master system for this lock.  This system is the process
                        04E2  1065 ;       system.  In the case of root locks, the master system has not
                        04E2  1066 ;       been determined yet so this request effectively functions as
                        04E2  1067 ;       a directory lookup.
                        04E2  1068 ;
                        04E2  1069 ; CALLING SEQUENCE:
                        04E2  1070 ;
                        04E2  1071 ;       JMP     LCK$SND_LOCKREQ (This routine does not return to its
                        04E2  1072 ;                                caller.  Instead it jumps to an
                        04E2  1073 ;                                appropriate destination.)
                        04E2  1074 ;       IPL must be at IPL$_SYNCH
                        04E2  1075 ;
                        04E2  1076 ; INPUT PARAMETERS:
                        04E2  1077 ;
                        04E2  1078 ;       R3      CSID of destination system
                        04E2  1079 ;       R6      Address of LKB
                        04E2  1080 ;       R8      Address of RSB
                        04E2  1081 ;       R9      Flags
                        04E2  1082 ;
                        04E2  1083 ; OUTPUT PARAMETERS:
                        04E2  1084 ;
                        04E2  1085 ;       R0      Completion code (on some exit paths)
                        04E2  1086 ;
                        04E2  1087 ; SIDE EFFECTS:
                        04E2  1088 ;
                        04E2  1089 ;       The process will go into MWAIT state until the response from the
                        04E2  1090 ;       remote system arrives.
                        04E2  1091 ;
                        04E2  1092 ;       R0 - R5 destroyed
                        04E2  1093 ;--
                        04E2  1094
                        04E2  1095          .ENABL  LSB
                        04E2  1096
                        04E2  1097 LCK$SND_LOCKREQ::
                        04E2  1098
                        04E2  1099          ; The LKB is inserted on the RSB's wait queue until the response
                        04E2  1100          ; arrives.  Note:  Normally, the LKB state field should get stored
                        04E2  1101          ; adjacent to the INSQUE.  Here, it gets stored shortly; either
                        04E2  1102          ; in SENDLOCKREQ or in the code that waits for pool.
                        04E2  1103
              38 A6   0E  04E2  1104          INSQUE  LKB$L_SQFL(R6),-        ; Insert LKB onto tail of RSB's
              24 B8       04E5  1105                  @RSB$L_WTQBL(R8)        ; wait queue.
                          04E7  1106
                          04E7  1107 RESEND: ; Allocate a CDRP and send the message
                          04E7  1108
    54  00000000'GF   D0  04E7  1109          MOVL    G^SCH$GL_CURPCB,R4      ; Get our PCB address
            FB0F'   30  04EE  1110          BSBW    CNX$ALLOC_WARMCDRP      ; Alloc. a CDRP with RSPID and CVT CSID
          5D 50     E9  04F1  1111          BLBC    R0,WAITFORMEM           ; No CDRP or CSID error
       50   56     D0  04F4  1112          MOVL    R6,R0                   ; Move LKB address
       51   58     D0  04F7  1113          MOVL    R8,R1                   ; Move RSB address
     48 A5  54     D0  04FA  1114          MOVL    R4,CDRP$L_VAL8(R5)      ; Store PCB address in CDRP
```

```
                00C4   30  04FE  1115          BSBW    SENDLOCKREQ             ; Send the lock request
          54    48 A5  D0  0501  1116          MOVL    CDRP$L_VAL8(R5),R4      ; Restore PCB address
                       0505  1117
                       0505  1118 LCK_SCS_WAIT:
             50   0D   D0  0505  1119          MOVL    #RSN$_SCS,R0            ; Go into MWAIT for resource RSN$_SCS
                       0508  1120
                       0508  1121 LCK_WAIT:
                       0508  1122          ; Put the process into MWAIT until the response arrives or
                       0508  1123          ; the resource becomes available.  Resource number is in R0.
                       0508  1124          ; PCB address is in R4.
                       0508  1125
                 60 A4 D0  0508  1126          MOVL    PCB$L_PID(R4),-         ; Make sure PID is stored in case lock
                 0C A6     050B  1127                  LKB$L_PID(R6)           ; is system owned
                    10 78  050D  1128          ASHL    #PSL$V_IPL,-            ; Create a PSL on stack with IPL
              7E    02     050F  1129                  #IPL$_ASTDEL,-(SP)      ; set to IPL$_ASTDEL
         00000000'GF  16  0511  1130          JSB     G^SCH$RWAIT             ; Wait
                       0517  1131
                       0517  1132          ; Upon reawakening the LKB$B_STATE field tells us what action
                       0517  1133          ; to perform.  But first, we have to make sure that we're not
                       0517  1134          ; stalling lock requests.
                       0517  1135
                       0517  1136          SETIPL  #IPL$_SYNCH             ; Raise IPL
         00000000'GF   95  051A  1137          TSTB    G^LCK$GB_STALLREQS      ; Are we stalling?
                    1D 12  0520  1138          BNEQ    30$                     ; Yes
          03 59     06 E1  0522  1139 15$:     BBC     #LCK$V_CVTSYS,R9,20$    ; Branch if not system owned
                 0C A6 D4  0526  1140          CLRL    LKB$L_PID(R6)           ; Clear PID of system owned lock
                       0529  1141
                       0529  1142 20$:     DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
                       0529  1143                  <-
                       0529  1144                  <RSPDOLOCL,DOLOCL_PROC>,-       ; Do locally
                       0529  1145                  <RSPGRANTD,GRANTD_PROC>,-       ; Granted
                       0529  1146                  <RSPQUEUED,WAITNG_PROC>,-       ; Waiting
                       0529  1147                  <RSPNOTQED,NOTQED_PROC>,-       ; Queued
                       0529  1148                  <SCSWAIT,LCK_SCS_WAIT>,-        ; Spurious wakeup
                       0529  1149                  <RETRY,LCKRETRY_PROC>,-         ; Retry lock
                       0529  1150                  >
                       053B  1151
                       053B  1152          BUG_CHECK       LOCKMGRERR,FATAL        ; Invalid lock state
                       053F  1153
                       053F  1154 30$:     ; We are stalling some or all lock requests.  See which ones.
                       053F  1155
                    05 19  053F  1156          BLSS    40$                     ; All requests are being stalled
                    09 E1  0541  1157          BBC     #LKB$V_PROTECT,-        ; Only protected locks are being stalled
          DC 2A     A6     0543  1158                  LKB$W_STATUS(R6),15$    ; Branch if this is not a protected lock
             50     0E  D0  0546  1159 40$:     MOVL    #RSN$_CLUSTRAN,R0       ; Resource to wait for
                    BD 11  0549  1160          BRB     LCK_WAIT                ; Go into MWAIT for RSN$_CLUSTRAN
                       054B  1161
                       054B  1162 RESEND_BRANCH:
                    9A 11  054B  1163          BRB     RESEND                  ; *** TEMPORARY ***
                       054D  1164
                       054D  1165 CSID_ERROR:                                 ; CSID to CSB conversion error
                       054D  1166          BUG_CHECK       LOCKMGRERR,FATAL
                       0551  1167
                       0551  1168 WAITFORMEM:
                       0551  1169          ; CSID conversion error or insufficient pool to allocate
                       0551  1170          ; CDRP.  If not CSID error then wait for pool and try again.
                       0551  1171
```

```
                50   B1  0551  1172          CMPW    R0,#SS$_INSFMEM         ; Is it insuff. memory?
                F5   12  0556  1173          BNEQ    CSID_ERROR              ; No
              FE 8F  90  0558  1174          MOVB    #LKB$K_RETRY,-          ; Store lock state
              36 A6      055B  1175                  LKB$B_STATE(R6)
            50   03  D0  055D  1176          MOVL    #RSN$_NPDYNMEM,R0       ; Resource to wait for
                 A6  11  0560  1177          BRB     LCK_WAIT                ; Go into MWAIT for RSN$_NPDYNMEM
                     0562  1178
                     0562  1179          .DSABL  LSB
                     0562  1180
                     0562  1181 ;*****************************************************************************
                     0562  1182 ;
                     0562  1183 ;       Process level routines for handling remote lock requests
                     0562  1184 ;
                     0562  1185 ;*****************************************************************************
                     0562  1186
                     0562  1187 LCKRETRY_PROC:
         53    38 A8  D0  0562  1188          MOVL    RSB$L_CSID(R8),R3       ; Get CSID of destination system
                E3  12  0566  1189          BNEQ    RESEND_BRANCH           ; Try again
                     0568  1190
                     0568  1191          ; This system became the manager of the resource while we
                     0568  1192          ; were waiting.  Fall through to ...
                     0568  1193
                     0568  1194 DOLOCL_PROC:
                     0568  1195          ; Handle the request locally.
                     0568  1196
         50    38 A6  0F  0568  1197          REMQUE  LKB$L_SQFL(R6),R0       ; Remove from wait queue
   00000000'GF     17  056C  1198          JMP     G^LCK$LOCAL_LOCK
                     0572  1199
                     0572  1200 GRANTD_PROC:
                     0572  1201          ; Lock request was granted
                     0572  1202
                03   E1  0572  1203          BBC     #LKB$V_BLKASTQED,-      ; Branch if blocking AST should
             0D 2A A6        0574  1204                  LKB$W_STATUS(R6),10$    ; not be queued
                02   A8  0577  1205          BISW    #LKB$M_DBLKAST,-        ; Set deliver blocking AST status
                2A A6      0579  1206                  LKB$W_STATUS(R6)
                10   88  057B  1207          BISB    #LKB$M_PKAST,-          ; Set piggyback kernel AST bit
                0B A6      057D  1208                  LKB$B_RMOD(R6)
                20 A6  D0  057F  1209          MOVL    LKB$L_BLKASTADR(R6),-   ; Store address of blocking AST routine
                10 A6      0582  1210                  LKB$L_AST(R6)
         50     38 A6  0F  0584  1211 10$:     REMQUE  LKB$L_SQFL(R6),R0       ; Remove lock from RSB's wait queue
                34 A6  90  0588  1212          MOVB    LKB$B_RQMODE(R6),-      ; Set granted mode
                35 A6      058B  1213                  LKB$B_GRMODE(R6)
                00'    D0  058D  1214          MOVL    S^#SS$_NORMAL,-         ; Set status for lock status block
                2C A6      058F  1215                  LKB$L_LKST1(R6)
                20 A6  D5  0591  1216          TSTL    LKB$L_BLKASTADR(R6)     ; Blocking AST specified?
                   03  13  0594  1217          BEQL    15$                     ; No
                42 A8  B6  0596  1218          INCW    RSB$W_BLKASTCNT(R8)     ; Yes, adjust count
   00000000'GF     16  0599  1219 15$:     JSB     G^LCK$GRANT_REM         ; Finish granting the lock
   00000000'GF     17  059F  1220          JMP     G^LCK$SYNC_EXIT         ; Exit the service
                     05A5  1221
                     05A5  1222 WAITNG_PROC:
                     05A5  1223          ; Lock request had to wait.  PCB address is still in R4.
                     05A5  1224
   00000000'GF     16  05A5  1225          JSB     G^LCK$QUEUE_REM         ; Queue the lock request
   00000000'GF     17  05AB  1226          JMP     G^LCK$QUEUED_EXIT       ; Exit the service
                     05B1  1227
                     05B1  1228 NOTQED_PROC:
```

```
                     05B1  1229          ; Lock request was not queued due to an error.  Status code
                     05B1  1230          ; returned by remote system is in LKB.  Typical values are
                     05B1  1231          ;  SS$_NOTQUEUED or SS$_NOLOCKIDS.
                     05B1  1232
 5B   2C A6    3C    05B1  1233          MOVZWL  LKB$L_LKST1(R6),R11     ; Get status
                     05B5  1234
                     05B5  1235 CHECK_RSB:
                     05B5  1236          ; LKB is to be deallocated.  Check if RSB should be deallocated too.
                     05B5  1237          ; Status is in R11.
                     05B5  1238
 50   38 A6    0F    05B5  1239          REMQUE  LKB$L_SQFL(R6),R0       ; Remove lock from RSB's wait queue
00000000'GF   16    05B9  1240          JSB     G^LCK$CHECK_RSB         ; Deallocate RSB if necessary
00000000'GF   17    05BF  1241          JMP     G^LCK$NOT_QUEUED        ; Cleanup and exit service
                     05C5  1242
```

```
                     05C5  1244          .SBTTL  SENDLOCKREQ - Send a (new) lock request
                     05C5  1245
                     05C5  1246 ;++
                     05C5  1247 ; FUNCTIONAL DESCRIPTION:
                     05C5  1248 ;
                     05C5  1249 ;       This routine takes care of building and sending the new lock
                     05C5  1250 ;       request message to the remote system.  It also handles
                     05C5  1251 ;       fork level processing of the response.  We are the process system;
                     05C5  1252 ;       the remote system is the master system.
                     05C5  1253 ;
                     05C5  1254 ; CALLING SEQUENCE:
                     05C5  1255 ;
                     05C5  1256 ;       BSBW    SENDLOCKREQ
                     05C5  1257 ;       IPL must be at SCS fork IPL (8)
                     05C5  1258 ;       This routine operates as a fork process so it may return
                     05C5  1259 ;       to its caller before completing.
                     05C5  1260 ;
                     05C5  1261 ; INPUT PARAMETERS:
                     05C5  1262 ;
                     05C5  1263 ;       R0      LKB address
                     05C5  1264 ;       R1      RSB Address
                     05C5  1265 ;       R3      CSB address
                     05C5  1266 ;       R5      CDRP address
                     05C5  1267 ;
                     05C5  1268 ; IMPLICIT INPUTS:
                     05C5  1269 ;
                     05C5  1270 ;       CDRP$L_VAL8     Contains PCB address of process requesting lock
                     05C5  1271 ;
                     05C5  1272 ; OUTPUT PARAMETERS:
                     05C5  1273 ;
                     05C5  1274 ;       R0 - R4 destroyed
                     05C5  1275 ;
                     05C5  1276 ; SIDE EFFECTS:
                     05C5  1277 ;
                     05C5  1278 ;       NONE
                     05C5  1279 ;
                     05C5  1280 ;--
                     05C5  1281          .ENABL  LSB
                     05C5  1282
                     05C5  1283 SENDLOCKREQ:
                     05C5  1284          ; Put everything we need to build the message into the CDRP.
                     05C5  1285
2C A5    50    7D    05C5  1286          MOVQ    R0,CDRP$L_VAL1(R5)      ; Store LKB and RSB addresses
51    64 A3    D0    05C9  1287          MOVL    CSB$L_CLUB(R3),R1       ; Get address of CLUB
   00AC C1    B0    05CD  1288          MOVW    CLUB$Q_MEMSEQ(R1),-     ; Store membership seq. number
     34 A5          05D1  1289                  CDRP$L_VAL3(R5)
   06FC'CF    9E    05D3  1290          MOVAB   W^BLD_LOCKMSG,-         ; Store address of routine that will
     4C A5          05D7  1291                  CDRP$L_MSGBLD(R5)       ; actually build message
     FD 8F    90    05D9  1292          MOVB    #LKB$K_SCSWAIT,-        ; Set LKB state to SCSWAIT
     36 A0          05DC  1293                  LKB$B_STATE(R0)
                    05DE  1294
                    05DE  1295          ; Send the message
                    05DE  1296
      FA1F'    30   05DE  1297          BSBW    CNX$SEND_MSG_CSB
     41 50    E9    05E1  1298          BLBC    R0,LCK_SENDERR          ; Destination system is being
                    05E4  1299                                          ; removed from cluster
                    05E4  1300
```

```
                     05E4  1301          ; We are resumed here when the response message arrives.
                     05E4  1302          ; Registers contain:
                     05E4  1303          ;       R2      Address of message buffer
                     05E4  1304          ;       R3      Address of CSB
                     05E4  1305          ;       R4      Address of PDT
                     05E4  1306          ;       R5      Address of CDRP
                     05E4  1307          ; First validate membership sequence number.  I.e. ignore response
                     05E4  1308          ; and retry request if we did a membership state change while
                     05E4  1309          ; this message was in transit.  Otherwise store response code
                     05E4  1310          ; in LKB and then dispatch on it.
                     05E4  1311
50   64 A3    D0     05E4  1312          MOVL    CSB$L_CLUB(R3),R0       ; Get address of CLUB
  00AC C0    B1     05E8  1313          CMPW    CLUB$W_MEMSEQ(R0),-     ; Compare sequence numbers
    0C A2           05EC  1314                  LKMSG$W_MEMSEQ(R2)
       05    13     05EE  1315          BEQL    20$                     ; They match
    FE 8F    90     05F0  1316          MOVB    #LKB$K_RETRY,-          ; No match - change state to RETRY
    1E A2           05F3  1317                  LKMSG$B_STATE(R2)
50   2C A5    7D     05F5  1318 20$:     MOVQ    CDRP$L_VAL1(R5),R0      ; R0 = LKB address;  R1 = RSB address
    36 A0    91     05F9  1319          CMPB    LKB$B_STATE(R0),-       ; Make sure LKB is in SCSWAIT state
    FD 8F           05FC  1320                  #LKB$K_SCSWAIT
       1D    12     05FE  1321          BNEQ    30$                     ; It's not
    1E A2    90     0600  1322          MOVB    LKMSG$B_STATE(R2),-     ; Store state
    36 A0           0603  1323                  LKB$B_STATE(R0)
                    0605  1324
                    0605  1325          DISPATCH        LKB$B_STATE(R0),TYPE=B,PREFIX=LKB$K_,-
                    0605  1326                  <-
                    0605  1327                  <RETRY,DEALL_WARMCDRP>,-        ; Retry this request
                    0605  1328                  <RSPRESEND,RESEND_FORK>,-       ; Resend to specified node
                    0605  1329                  <RSPDOLOCL,DOLOCL_FORK>,-       ; Do locally
                    0605  1330                  <RSPGRANTD,GRANTD_FORK>,-       ; Granted
                    0605  1331                  <RSPQUEUED,WAITNG_FORK>,-       ; Waiting
                    0605  1332                  <RSPNOTQED,NOTQED_FORK>,-       ; Not queued
                    0605  1333                  >
                    0619  1334
                    0619  1335          BUG_CHECK       LOCKMGRERR,FATAL        ; Invalid lock state
                    061D  1336
                    061D  1337 30$:     BUG_CHECK       LOCKMGRERR,FATAL        ; LKB not in SCSWAIT state
                    0621  1338 35$:     BUG_CHECK       LOCKMGRERR,FATAL        ; Other system returned
                    0625  1339                                                  ; invalid CSID
                    0625  1340 LCK_SENDERR:
                    0625  1341          ; Remote system is being removed from cluster
                    0625  1342
 54   2C A5    D0     0625  1343          MOVL    CDRP$L_VAL1(R5),R4      ; Get LKB address
   50   55    D0     0629  1344          MOVL    R5,R0                   ; Address of CDRP
00000000'GF    16     062C  1345          JSB     G^EXE$DEANONPAGED       ; Deallocate CDRP
        4C    11     0632  1346          BRB     LCK_RETRY               ; Wake process and retry request
                    0634  1347
                    0634  1348 ;*****************************************************************************
                    0634  1349 ;
                    0634  1350 ;       Fork level routines for handling remote lock request responses
                    0634  1351 ;
                    0634  1352 ;*****************************************************************************
                    0634  1353
                    0634  1354 RESEND_FORK:
                    0634  1355          ; Resend request to another system (specified by CSID).
                    0634  1356          ; New destination CSID is in message.  But before we use it we
                    0634  1357          ; have to make sure the CSID address in the RSB didn't change out from
```

```
                     0634  1358          ; underneath us by a different thread.  If it did, we have to use the
                     0634  1359          ; stored CSID address because if we were to change the stored one,
                     0634  1360          ; we could confuse the other thread.   Once we determine
                     0634  1361          ; the correct CSID we deallocate the old
                     0634  1362          ; CDRP and message buffer and allocate a new ones.  The deallocate
                     0634  1363          ; and allocate are necessary since CRDPs and message buffers are
                     0634  1364          ; maintained on a per CSB basis.  Also, the CDRP will get
                     0634  1365          ; initialized when it's reallocated.  R3 still contains old
                     0634  1366          ; CSB address
                     0634  1367
          00000002   0634  1368          .IF NE  CA$_MEASURE
00000000'GF    D6    0634  1369          INCL    G^PMS$GL_DIR_OUT
                     063A  1370          .ENDC
                     063A  1371
 54    18 A2    D0   063A  1372          MOVL    LKMSG$L_CSID(R2),R4      ; Get specified CSID
       38 A1    D1   063E  1373          CMPL    RSB$L_CSID(R1),-         ; Has the CSID in the RSB changed?
       4C A3         0641  1374                  CSB$L_CSID(R3)
          06    13   0643  1375          BEQL    40$                      ; No
 54    38 A1    D0   0645  1376          MOVL    RSB$L_CSID(R1),R4        ; Yes, get new CSID
          3C    13   0649  1377          BEQL    50$                      ; It's now being done locally
 38 A1    54    D0   064B  1378 40$:     MOVL    R4,RSB$L_CSID(R1)        ; Store new CSID
       48 A5    DD   064F  1379          PUSHL   CDRP$L_VAL8(R5)          ; Save PCB address
    7E    50    7D   0652  1380          MOVQ    R0,-(SP)                 ; Save LKB and RSB addresses
        F9A8'   30   0655  1381          BSBW    CNX$DEALL_WARMCDRP_CSB   ; Deallocate old CDRP and message buffer
    53    54    D0   0658  1382          MOVL    R4,R3                    ; Move new CSID
00000000'GF    95   065B  1383          TSTB    G^LCK$GB_STALLREQS       ; Are we stalling lock requests?
          17    12   0661  1384          BNEQ    47$                      ; Yes, wake process
        F99A'   30   0663  1385          BSBW    CNX$ALLOC_WARMCDRP       ; Alloc. a CDRP with RSPID and cvt CSID
       0A 50    E9   0666  1386          BLBC    R0,45$                   ; Allocation or CSID convert failure
    50    8E    7D   0669  1387          MOVQ    (SP)+,R0                 ; Restore LKB and RSB addresses
       48 A5 8ED0   066C  1388          POPL    CDRP$L_VAL8(R5)          ; Restore PCB address into new CDRP
        FF52   31   0670  1389          BRW     SENDLOCKREQ              ; Resend it
                     0673  1390
                     0673  1391 45$:     ; Unable to allocate a CDRP or CSID returned by other system
                     0673  1392          ; was invalid.  If it's insufficient memory then let process
                     0673  1393          ; do the allocate so that it can wait, if necessary.  If CSID
                     0673  1394          ; returned by the other system is invalid, then bugcheck.
                     0673  1395
0000'8F    50    B1   0673  1396          CMPW    R0,#SS$_INSFMEM          ; Is it insuff. memory?
          A7    12   0678  1397          BNEQ    35$                      ; No, error
    54    8E    7D   067A  1398 47$:     MOVQ    (SP)+,R4                 ; Restore LKB and RSB addresses
    5E    04    C0   067D  1399          ADDL    #4,SP                    ; Throw away PCB address
                     0680  1400
                     0680  1401 LCK_RETRY:
                     0680  1402          ; Set lock state to RETRY and wake process.  LKB address is in R4.
                     0680  1403
       FE 8F    90   0680  1404          MOVB    #LKB$K_RETRY,-           ; Set RETRY state
       36 A4         0683  1405                  LKB$B_STATE(R4)
          5C    11   0685  1406          BRB     WAKE_PROCESS
                     0687  1407
       F9 8F    90   0687  1408 50$:     MOVB    #LKB$K_RSPDOLOCL,-       ; Change state to handle request
       36 A0         068A  1409                  LKB$B_STATE(R0)          ; locally and fall through to ...
                     068C  1410
                     068C  1411 DOLOCL_FORK:
                     068C  1412          ; Handle request locally
                     068C  1413
          00000002   068C  1414          .IF NE  CA$_MEASURE
```

```
00000000'GF   D6  068C  1415          INCL    G^PMS$GL_DIR_OUT
                  0692  1416          .ENDC
                  0692  1417
      38 A1   D4  0692  1418          CLRL    RSB$L_CSID(R1)          ; Indicate we will handle this resource
         46   11  0695  1419          BRB     DEALL_WARMCDRP          ; Deallocate CDRP etc. and wake process
                  0697  1420
                  0697  1421 NOTQED_FORK:
                  0697  1422          ; Lock request was not queued due to error or incompatibility
                  0697  1423
      1C A2   B0  0697  1424          MOVW    LKMSG$W_STATUS(R2),-    ; Move completion status to LKB
      20 A0       069A  1425                  LKB$L_LKST1(R0)
         39   11  069C  1426          BRB     70$                     ; Deallocate CDRP etc. and wake process
                  069E  1427
                  069E  1428 WAITNG_FORK:
                  069E  1429          ; Lock request has to wait
                  069E  1430
      30 A2   B0  069E  1431          MOVW    LKMSG$W_RQSEQNM(R2),-   ; Store request sequence number
      10 A0       06A1  1432                  LKB$W_RQSEQNM(R0)
         2D   11  06A3  1433          BRB     60$
                  06A5  1434
                  06A5  1435 GRANTD_FORK:
                  06A5  1436          ; Set BLKASTQED bit in LKB if appropriate.  Then store value
                  06A5  1437          ; block from message unless one in RSB is newer.
                  06A5  1438
         01   E1  06A5  1439          BBC     #LKB$V_DBLKAST,-        ; Branch if blocking AST shouldn't
   04 18 A2       06A7  1440                  LKMSG$Q_LKBSTATUS(R2),55$ ; be queued
         02   A8  06AA  1441          BISW    #LKB$M_DBLKAST,-        ; Set deliver blocking AST bit
      2A A0       06AC  1442                  LKB$W_STATUS(R0)
      3C A1   C3  06AE  1443 55$:     SUBL3   RSB$L_VALSEQNUM(R1),-   ; Compare sequence numbers
54    30 A2       06B1  1444                  LKMSG$L_VALSEQNUM(R2),R4
         1C   15  06B4  1445          BLEQ    60$                     ; We already have a newer one
      20 A2   7D  06B6  1446          MOVQ    LKMSG$Q_VALBLK(R2),-    ; Store value block
      28 A1       06B9  1447                  RSB$Q_VALBLK(R1)
      28 A2   7D  06BB  1448          MOVQ    LKMSG$Q_VALBLK+8(R2),-
      30 A1       06BE  1449                  RSB$Q_VALBLK+8(R1)
      30 A2   D0  06C0  1450          MOVL    LKMSG$L_VALSEQNUM(R2),- ; Store sequence number
      3C A1       06C3  1451                  RSB$L_VALSEQNUM(R1)
         02   AA  06C5  1452          BICW    #RSB$M_VALINVLD,-       ; Clear value block invalid flag
      0E A1       06C7  1453                  RSB$W_STATUS(R1)
         01   E1  06C9  1454          BBC     #RSB$V_VALINVLD,-       ; and conditionally set it
   04 1A A2       06CB  1455                  LKMSG$Q_RSBSTATUS(R2),60$
         02   A8  06CE  1456          BISW    #RSB$M_VALINVLD,-
      0E A1       06D0  1457                  RSB$W_STATUS(R1)
                  06D2  1458
      10 A2   D0  06D2  1459 60$:     MOVL    LKMSG$L_MSTLKID(R2),-   ; Store remote lock id
      54 A0       06D5  1460                  LKB$L_REMLKID(R0)
                  06D7  1461
         00000002 06D7  1462 70$:     .IF NE  CA$_MEASURE
000C0000'GF   D6  06D7  1463          INCL    G^PMS$GL_ENQNEW_OUT
                  06DD  1464          .ENDC
                  06DD  1465
                  06DD  1466 DEALL_WARMCDRP:
                  06DD  1467          ; Deallocate CDRP (R5), message buffer (R2), and RSPID (in CDRP).
                  06DD  1468          ; LKB address is in R0.  CSB address is in R3.
                  06DD  1469
   54   50   D0  06DD  1470          MOVL    R0,R4                   ; Move LKB address
      F91D'   30  06E0  1471          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
```

```
                                                                ;*****************************************************************************
                                                                ;
                                                                ;               Build message action routines for new lock requests
                                                                ;               and rebuild lock during failin/out
                                                                ;
                                                                ;*****************************************************************************

                                                        BLD_LOCKMSG:
                                                                ; Action routine to build the actual lock request message
                                                                ; Inputs are:
                                                                ;       R2      Address of message buffer
                                                                ;       R5      Address of CDRP
                                                                ; All registers except R0 and R1 must be preserved

                                                                ASSUME  LKMSG$W_HASHVAL  EQ  LKMSG$W_MEMSEQ+2
                                                                ASSUME  LKMSG$L_EPIDNEW  EQ  LKMSG$W_HASHVAL+2
                                                                ASSUME  LKMSG$L_PRCLKID  EQ  LKMSG$L_EPIDNEW+4
                                                                ASSUME  LKMSG$W_FLAGS  EQ  LKMSG$L_PRCLKID+4
                                                                ASSUME  LKMSG$B_RQMODE  EQ  LKMSG$W_FLAGS+2
                                                                ASSUME  LKMSG$B_GRMODE  EQ  LKMSG$B_RQMODE+1
                                                                ASSUME  LKMSG$L_BLKASTFLG  EQ  LKMSG$B_GRMODE+1
                                                                ASSUME  LKMSG$L_PARPRCLKID  EQ  LKMSG$L_BLKASTFLG+4
                                                                ASSUME  LKMSG$L_PARMSTLKID  EQ  LKMSG$L_PARPRCLKID+4
                                                                ASSUME  LKMSG$W_GROUP  EQ  LKMSG$L_PARMSTLKID+4
                                                                ASSUME  LKMSG$B_RMOD  EQ  LKMSG$W_GROUP+2
                                                                ASSUME  LKMSG$B_RSNLEN  EQ  LKMSG$B_RMOD+1
                                                                ASSUME  LKMSG$T_RESNAM  EQ  LKMSG$B_RSNLEN+1

                                                                ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
08 A2   0102 8F   B0                                            MOVW    #LKMSG$K_NEWLOCK@8-     ; Store function and facility codes
                                                                !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)

                                                        BLD_LOCKCOM:
          3C   BB                                               PUSHR   #^M<R2,R3,R4,R5>        ; Save registers
     50   2C   A5   7D                                         MOVQ    CDRP$L_VAL1(R5),R0      ; R0 = LKB address;  R1 = RSB address
     52   0C   C0                                              ADDL    #LKMSG$W_MEMSEQ,R2      ; Point inside message
     82   34   A5   B0                                         MOVW    CDRP$L_VAL3(R5),(R2)+   ; Store rebuild seq. number
     82   44   A1   B0                                         MOVW    RSB$W_HASHVAL(R1),(R2)+ ; Store resource hash value
     54   48   A5   D0                                         MOVL    CDRP$L_VAL8(R5),R4      ; Get PCB address
          3C   13                                              BEQL    20$                     ; Branch if system owned (rebuild only)
     82   64   A4   D0                                         MOVL    PCB$L_EPID(R4),(R2)+    ; Store EPID
     010C C4   D0                                              MOVL    PCB$L_DLCKPRI(R4),-     ; Store deadlock priority
          38   A2                                              LKMSG$L_DLCKPRI_NEW-LKMSG$L_PRCLKID(R2)
     82   30   A0   D0                                 5$:     MOVL    LKB$L_LKID(R0),(R2)+    ; Store process lock id
     82   28   A0   B0                                         MOVW    LKB$W_FLAGS(R0),(R2)+   ; Store input flags
     82   34   A0   B0                                         MOVW    LKB$B_RQMODE(R0),(R2)+  ; Store lock modes
          20   A0   D0                                         MOVL    LKB$L_BLKASTADR(R0),-   ; Set blocking AST indicator.  Note:
               82                                              (R2)+                   ; address is not used - just zero or not
               53   D4                                         CLRL    R3                      ; Assume no parent lock id
     54   48   A0   D0                                         MOVL    LKB$L_PARENT(R0),R4     ; Get address of parent LKB
               08   13                                         BEQL    10$                     ; No parent
     53   30   A4   D0                                         MOVL    LKB$L_LKID(R4),R3       ; Get parent's process lock id
     54   54   A4   D0                                         MOVL    LKB$L_REMLKID(R4),R4    ; and master lock id
     82        53   7D                                 10$:     MOVQ    R3,(R2)+                ; Store both parent lock ids
     50   4F   A1   9A                                         MOVZBL  RSB$B_RSNLEN(R1),R0     ; Get resource name length
          50   04   C0                                         ADDL    #4,R0                   ; Account for adjacent fields
62   4C   A1   50   28                                         MOVC3   R0,RSB$W_GROUP(R1),(R2) ; Move group number, access mode,
                                                                                               ; res. name length and resource name
```

```
                     3C   BA   0752  1546          POPR    #^M<R2,R3,R4,R5>        ; Restore registers
                          05   0754  1547          RSB
                               0755  1548
                               0755  1549 20$:     ; Have a zero for PCB address.  This should only occur doing rebuild
                               0755  1550          ; for system owned locks.  Verify CVTSYS bit is set.
                               0755  1551
                    52    04   C0   0755  1552          ADDL    #4,R2                   ; Move past EPID field
                          06   E0   0758  1553          BBS     #LCK$V_CVTSYS,-         ; Continue if CVTSYS is set
                   C6 28  A0        075A  1554                  LKB$W_FLAGS(R0),5$
                               075D  1555          BUG_CHECK       LOCKMGRERR,FATAL; CVTSYS not set and PCB address = 0
                               0761  1556
                               0761  1557 LCK$BLD_REBLDLOCK::
              08 A2    0802  8F   B0   0761  1558          MOVW    #LKMSG$K_REBLDLOCK@8-   ; Store function and facility codes
                               0767  1559                  !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
                          99   10   0767  1560          BSBB    BLD_LOCKCOM             ; Build normal portion of message
                    50    2C A5   7D   0769  1561          MOVQ    CDRP$L_VAL1(R5),R0      ; R0 = LKB address;  R1 = RSB address
                       10 A0   B0   076D  1562          MOVW    LKB$W_RQSEQNM(R0),-     ; Store request sequence number
                       50 A2        0770  1563                  LKMSG$W_RQSEQALT(R2)
                       28 A1   7D   0772  1564          MOVQ    RSB$Q_VALBLK(R1),-      ; Store value block
                       54 A2        0775  1565                  LKMSG$Q_VALBLKALT(R2)
                       30 A1   7D   0777  1566          MOVQ    RSB$Q_VALBLK+8(R1),-
                       5C A2        077A  1567                  LKMSG$Q_VALBLKALT+8(R2)
                       3C A1   D0   077C  1568          MOVL    RSB$L_VALSEQNUM(R1),-   ; Store value block sequence number
                       64 A2        077F  1569                  LKMSG$L_VALSEQALT(R2)
                               0781  1570
                               0781  1571          ASSUME  LKMSG$B_RSTATUS  EQ LKMSG$B_LSTATUS+1
                               0781  1572          ASSUME  LKB$V_ASYNC  LE  7
                               0781  1573          ASSUME  LKB$V_BLKASTQED  LE  7
                               0781  1574          ASSUME  RSB$V_VALINVLD  LE  7
                               0781  1575
                       68 A2   B4   0781  1576          CLRW    LKMSG$B_LSTATUS(R2)     ; Clear LKB and RSB status fields in msg
          68 A2    2A A0   F3 8F   8B   0784  1577          BICB3   #^C<LKB$M_BLKASTQED!-   ; Set specified LKB status bits
                               078B  1578                  LKB$M_ASYNC>,LKB$W_STATUS(R0),LKMSG$B_LSTATUS(R2)
                       FD 8F   8B   078B  1579          BICB3   #^C<RSB$M_VALINVLD>,-   ; Set specified RSB status bits
                 69 A2 0E A1        078E  1580                  RSB$W_STATUS(R1),LKMSG$B_RSTATUS(R2)
                    51   36 A0   90   0792  1581          MOVB    LKB$B_STATE(R0),R1      ; Get lock state
                               0796  1582
                               0796  1583          ; Handle special-cased lock states
                               0796  1584
                               0796  1585          DISPATCH        R1,TYPE=B,PREFIX=LKB$K_,-
                               0796  1586                  <-
                               0796  1587                  <GRANTED,50$>,-
                               0796  1588                  <CONVERT,50$>,-
                               0796  1589                  <WAITING,50$>,-
                               0796  1590                  <RETRY,20$>,-
                               0796  1591                  <SCSWAIT,20$>,-
                               0796  1592                  >
                               07A5  1593
                               07A5  1594          BUG_CHECK       LOCKMGRERR,FATAL        ; Illegal lock state
                               07A9  1595
                               07A9  1596 20$:     ; Lock state is RETRY or SCSWAIT.  This must be a
                               07A9  1597          ; conversion request as new lock requests in this state aren't even
                               07A9  1598          ; rebuilt.  Store the old blocking AST address in the message because
                               07A9  1599          ; that is what the master copy should hold at this point.  Then change
                               07A9  1600          ; the lock state to GRANTED.
                               07A9  1601
                       5C A0   D0   07A9  1602          MOVL    LKB$L_OLDBLKAST(R0),-   ; Store old blocking AST address
```

```
                                                K 2
                     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  1609          .SBTTL  LCK$RCV_LOCKREQ - Receive lock request
07B6  1610
07B6  1611 ;++
07B6  1612 ;  FUNCTIONAL DESCRIPTION
07B6  1613 ;
07B6  1614 ;       This routine receives new lock requests from the remote
07B6  1615 ;       system.  The other system is the process system and this system is
07B6  1616 ;       the master system.  Root lock requests are special-cased in that
07B6  1617 ;       the root directory logic is involved (see below).
07B6  1618 ;
07B6  1619 ; CALLING SEQUENCE:
07B6  1620 ;
07B6  1621 ;       JSB     LCK$RCV_LOCKREQ (called from SCS received message routine)
07B6  1622 ;       IPL must be at IPL$_SYNCH
07B6  1623 ;
07B6  1624 ; INPUT PARAMETERS:
07B6  1625 ;
07B6  1626 ;       R2      Address of message buffer
07B6  1627 ;       R3      Address of CSB
07B6  1628 ;       R5      Address of CDRP for response message
07B6  1629 ;
07B6  1630 ; OUTPUT PARAMETERS:
07B6  1631 ;
07B6  1632 ;       None
07B6  1633 ;
07B6  1634 ; SIDE EFFECTS:
07B6  1635 ;
07B6  1636 ;       A message may be sent back as a response
07B6  1637 ;
07B6  1638 ;
07B6  1639 ; DESCRIPTION OF DIRECTORY PROCESSING LOGIC:
07B6  1640 ;
07B6  1641 ;
07B6  1642 ;       If this is a sub-resource
07B6  1643 ;           then
07B6  1644 ;               Get address of parent resource;
07B6  1645 ;           else;
07B6  1646 ;       Search resource hash table for a matching resource;
07B6  1647 ;       If resource found
07B6  1648 ;           then
07B6  1649 ;               If resource is managed locally (CSB address = 0)
07B6  1650 ;                   then
07B6  1651 ;                       Handle lock request here;
07B6  1652 ;                       Send response (granted, waiting, blocking, or notqueued);
07B6  1653 ;                   else
07B6  1654 ;                       Bugcheck if this is a sub-resource
07B6  1655 ;                       If system managing the resource is the same
07B6  1656 ;                       system that sent us this request      [DIRECTORY LOOKUP]
07B6  1657 ;                           then
07B6  1658 ;                               Send response (do locally);
07B6  1659 ;                           else
07B6  1660 ;                               Send response (resend to specified
07B6  1661 ;                               system);
07B6  1662 ;           else
07B6  1663 ;               If this is a root resource (PARENT = 0)
07B6  1664 ;                   then
07B6  1665 ;                       If this is the directory node for this resource
```

```
                                                07B6  1666 ;                               then
                                                07B6  1667 ;                                   Create directory entry      [ENTER IN DIRECTORY]
                                                07B6  1668 ;                                   Send response (do locally);
                                                07B6  1669 ;                               else
                                                07B6  1670 ;                                   Send response (re-lookup in directory);
                                                07B6  1671 ;                                                           [DIRECTORY RACE]
                                                07B6  1672 ;                           else
                                                07B6  1673 ;                               Bugcheck if parent is not managed here (CSID not 0)
                                                07B6  1674 ;                               Handle lock request here
                                                07B6  1675 ;                               Send response (granted, blocking, waiting or
                                                07B6  1676 ;                               not queued);
                                                07B6  1677 ;
                                                07B6  1678 ;--
                                                07B6  1679
                                                07B6  1680 LCK$RCV_LOCKREQ::
                51    53    D0  07B6  1681          MOVL    R3,R1                           ; Move CSB address
                0FC6  8F    BB  07B9  1682          PUSHR   #^M<R1,R2,R6,R7,R8,R9,R10,R11>
                      55    DD  07BD  1683          PUSHL   R5                              ; Save CDRP address
                    4C  A3  DD  07BF  1684          PUSHL   CSB$L_CSID(R3)                  ; Convert CSB address to CSID and save
                59    52    D0  07C2  1685          MOVL    R2,R9                           ; Save input message address
                   F838'    30  07C5  1686          BSBW    CNX$INIT_CDRP                   ; Initialize CDRP
                                07C8  1687
                                07C8  1688          ; If this is not a root resource request then get address of parent
                                07C8  1689          ; RSB (needed for hash table search) and verify that the resource
                                07C8  1690          ; is managed on this system.
                                07C8  1691
                      57    D4  07C8  1692          CLRL    R7                              ; Assume no parent RSB
            56    24  A9    D0  07CA  1693          MOVL    LKMSG$L_PARMSTLKID(R9),R6 ; Get parent lock id
                      11    13  07CE  1694          BEQL    20$                             ; No parent - this is a root resource
                51    56    D0  07D0  1695          MOVL    R6,R1                           ; Move parent id
            50    20  A9    D0  07D3  1696          MOVL    LKMSG$L_PARPRCLKID(R9),R0 ; Get remote parent lock id for verify
                    07FD    30  07D7  1697          BSBW    VERIFYREMLKID                   ; Verify lockid and convert to LKB addr.
                      23    50  E9  07DA  1698          BLBC    R0,80$                          ; Invalid lock id
            57    50  A6    D0  07DD  1699          MOVL    LKB$L_RSB(R6),R7                ; Get parent RSB address
            24  A9    57    D0  07E1  1700 20$:     MOVL    R7,LKMSG$L_PARMSTLKID(R9) ; Store in message for convenience
                                07E5  1701
                                07E5  1702          ; Search hash table for a matching resource.  The input message
                                07E5  1703          ; is used as the template to match.
                                07E5  1704
                                07E5  1705          ASSUME  LKMSG$W_GROUP  EQ  LKMSG$L_PARMSTLKID+4
                                07E5  1706          ASSUME  LKMSG$B_RMOD  EQ  LKMSG$W_GROUP+2
                                07E5  1707          ASSUME  LKMSG$B_RSNLEN  EQ  LKMSG$B_RMOD+1
                                07E5  1708          ASSUME  LKMSG$T_RESNAM  EQ  LKMSG$B_RSNLEN+1
                                07E5  1709
            54    24  A9    9E  07E5  1710          MOVAB   LKMSG$L_PARMSTLKID(R9),R4 ; Point to start of fields to match
            5A    2B  A9    9A  07E9  1711          MOVZBL  LKMSG$B_RSNLEN(R9),R10  ; Get length of resource name
                  5A    08    C0  07ED  1712          ADDL    #8,R10                          ; Account for other fields to match
            51    0E  A9    3C  07F0  1713          MOVZWL  LKMSG$W_HASHVAL(R9),R1  ; Get resource hash value
00000000'GF       16        16  07F4  1714          JSB     G^LCK$SRCH_HSHTBL               ; Search hash table
                    19    50  E9  07FA  1715          BLBC    R0,EXISTING_RESOURCE
                    0150    31  07FD  1716          BRW     NEW_RESOURCE
                                0800  1717
                                0800  1718 80$:     ; Other system sent us an invalid lock id
                                0800  1719
                                0800  1720          BUG_CHECK       INVLOCKID,FATAL
```

```
                     0804  1722 ;*****************************************************************************
                     0804  1723 ;
                     0804  1724 ;               Existing resource
                     0804  1725 ;
                     0804  1726 ;*****************************************************************************
                     0804  1727
                     0804  1728         .ENABL  LSB
                     0804  1729
                     0804  1730 5$:      ; It looks like we should perform a directory lookup.  However,
                     0804  1731         ; if we are rebuilding a lock during failover, then additional
                     0804  1732         ; checking must be performed.
                     0804  1733
      09 A9    91    0804  1734         CMPB    CLSMSG$B_FUNC(R9),-     ; Is this a rebuild lock message
         08          0807  1735                 #LKMSG$K_REBLDLOCK      ; (only sent during failover)?
         09    12    0808  1736         BNEQ    10$                     ; No
   55    6E    D0    080A  1737         MOVL    (SP),R5                 ; Yes, get CSID of requesting system
      F7F0'    30    080D  1738         BSBW    LCK$CHECK_DIRENTRY      ; Check if it's really a directory entry
      0C 50    E9    0810  1739         BLBC    R0,15$                  ; It's not - build the lock
       0096    31    0813  1740 10$:     BRW     DIR_LOOKUP              ; Do a directory lookup
                     0816  1741
                     0816  1742 EXISTING_RESOURCE:
                     0816  1743         ; We've found a matching RSB.  If the resource is being
                     0816  1744         ; managed by another system (CSID non-zero) then treat
                     0816  1745         ; this request as a directory lookup.  Otherwise, handle it
                     0816  1746         ; in this system.  Registers contain:
                     0816  1747         ;
                     0816  1748         ;       R5      Address of matching RSB
                     0816  1749         ;       R6      Address of parent LKB (or 0)
                     0816  1750         ;       R7      Address of parent RSB (or 0)
                     0816  1751         ;       R9      Address of input message
                     0816  1752
   58    55    D0    0816  1753         MOVL    R5,R8                   ; R8 will be used to point to RSB
54    38 A8    D0    0819  1754         MOVL    RSB$L_CSID(R8),R4       ; Get CSID
         E5    12    081D  1755         BNEQ    5$                      ; Do a directory lookup
                     081F  1756
         00000002    081F  1757 15$:     .IF NE  CA$_MEASURE
00000000'GF    D6    081F  1758         INCL    G^PMS$GL_ENQNEW_IN
                     0825  1759         .ENDC
                     0825  1760
       02AF    30    0825  1761         BSBW    BUILD_LKB               ; Build the lock block
      17 50    E9    0828  1762         BLBC    R0,19$                  ; Exit on any error
58 A6    6E    D0    082B  1763         MOVL    (SP),LKB$L_CSID(R6)     ; Store CSID of requesting system
      09 A9    91    082F  1764         CMPB    CLSMSG$B_FUNC(R9),-     ; Is this a normal lock message?
         01          0832  1765                 #LKMSG$K_NEWLOCK        ; (as opposed to a failover message)
         10    13    0833  1766         BEQL    20$                     ; Yes
      09 A9    91    0835  1767         CMPB    CLSMSG$B_FUNC(R9),-     ; Is this a rebuild lock message?
         08          0838  1768                 #LKMSG$K_REBLDLOCK
         03    12    0839  1769         BNEQ    18$                     ; No, must be a directory entry message
       01C0    31    083B  1770         BRW     REBLD_LOCK              ; Yes, rebuild the lock
                     083E  1771 18$:     BUG_CHECK       LOCKMGRERR,FATAL; Disagreement regarding who's mastering
                     0842  1772                                         ; this lock tree
       01D1    31    0842  1773 19$:     BRW     BLDLKB_ERROR2           ; Branch extender
                     0845  1774
                     0845  1775 20$:     ; Now see if the lock can be granted.  To be granted the conversion
                     0845  1776         ; and wait queues must be empty and the requested lock mode must be
                     0845  1777         ; compatible with the group grant mode of the resource.  (Actually,
                     0845  1778         ; any locks on the wait queue in states other than LKB$W_WAITING
```

```
                                0845  1779          ; are ignored).
                                0845  1780
                                0845  1781          ASSUME  RSB$L_WTQFL  EQ  RSB$L_CVTQFL+8
                                0845  1782
                50   18 A8   DE 0845  1783          MOVAL   RSB$L_CVTQFL(R8),R0     ; Get address of conversion queue
                   60   50   D1 0849  1784          CMPL    R0,(R0)                 ; Queue empty?
                        2F   12 084C  1785          BNEQ    50$                     ; No
                   50   08   C0 084E  1786          ADDL    #8,R0                   ; Get address of wait queue
                   51   50   D0 0851  1787          MOVL    R0,R1                   ; Save in R1
                   51   60   D1 0854  1788 30$:     CMPL    (R0),R1                 ; Queue empty?
                        1A   12 0857  1789          BNEQ    40$                     ; No
                                0859  1790
                                0859  1791 35$:     ; No waiting requests (or RECOVER bit is set); is the lock compatible?
                                0859  1792
                51   34 A6   9A 0859  1793          MOVZBL  LKB$B_RQMODE(R6),R1     ; Get lock mode
                55   0C A8   9A 085D  1794          MOVZBL  RSB$B_GGMODE(R8),R5     ; Get group grant mode
      18 00000000'GF45   51   E1 0861  1795          BBC     R1,G^LCK$COMPAT_TBL[R5],52$ ; Branch if incompatible
                                086A  1796
                                086A  1797          ; Lock can be granted
                                086A  1798
            00000000'GF   16    086A  1799          JSB     G^LCK$GRANT_LOCK                ; Grant the lock
                    016C   31   0870  1800          BRW     LOCK_GRANTED
                                0873  1801
                                0873  1802 40$:     ; Wait queue was not empty.  Skip over locks not in LKB$K_WAITING
                                0873  1803          ; state.  If we find even one lock in LKB$K_WAITING state, then
                                0873  1804          ; this lock is not grantable.
                                0873  1805
                50   60   D0 0873  1806          MOVL    (R0),R0                 ; Get next lock on wait queue
                  FF 8F   91 0876  1807          CMPB    #LKB$K_WAITING,-        ; Is it in LKB$K_WAITING state?
                  FE A0         0879  1808                  LKB$B_STATE-LKB$L_SQFL(R0)
                     D7   12    087B  1809          BNEQ    30$                     ; No, skip over it
                                087D  1810
                                087D  1811 50$:     ; Lock cannot be granted now due to other locks in the waiting
                                087D  1812          ; or conversion queue.  However, if the RECOVER bit is set, then
                                087D  1813          ; ignore these locks and try granting it anyway
                                087D  1814
                     07   E0    087D  1815          BBS     #LCK$V_RECOVER,-        ; Branch if recovering a lock
                D7 28 A6         087F  1816                  LKB$W_FLAGS(R6),35$
                                0882  1817
                                0882  1818 52$:     ; Lock cannot be granted now.  Queue it unless LCK$M_NOQUEUE
                                0882  1819          ; is set.
                                0882  1820
                     02   E0    0882  1821          BBS     #LCK$V_NOQUEUE,-        ; Branch if the NOQUEUE bit is set
                15 28 A6         0884  1822                  LKB$W_FLAGS(R6),55$
                   46 A8   B0    0887  1823          MOVW    RSB$W_RQSEQNM(R8),-     ; Store request sequence number
                   10 A6         088A  1824                  LKB$W_RQSEQNM(R6)
                   46 A8   B6    088C  1825          INCW    RSB$W_RQSEQNM(R8)       ; Increment sequence number
      00000000'GF   16    088F  1826          JSB     G^LCK$QUEUEWAIT         ; Queue it
       50   FB 8F   90    0895  1827          MOVB    #LKB$K_RSPQUEUED,R0     ; Set response state
                014B   31    0899  1828          BRW     LOCK_QUEUED
                                089C  1829
                                089C  1830 55$:     ; Lock should not be queued.  Clean up and exit.
                                089C  1831
                    00000002    089C  1832          .IF NE  CA$_MEASURE
      00000000'GF   D6    089C  1833          INCL    G^PMS$GL_ENQNOTQD
                                08A2  1834          .ENDC
                                08A2  1835
```

```
                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  1842 ;*****************************************************************************
                        08AC  1843 ;
                        08AC  1844 ;               Directory handling and error routines
                        08AC  1845 ;
                        08AC  1846 ;*****************************************************************************
                        08AC  1847
                        08AC  1848         .ENABL  LSB
                        08AC  1849
                        08AC  1850 DIR_LOOKUP:
                        08AC  1851         ; Treat this request as a directory lookup request by returning
                        08AC  1852         ; the CSID of the system managing this resource.  If the
                        08AC  1853         ; CSID matches that of the requestor then return a different
                        08AC  1854         ; response state to indicate this.  R4 contains CSID of
                        08AC  1855         ; system managing resource.
                        08AC  1856
          00000002      08AC  1857         .IF NE  CA$_MEASURE
00000000'GF    D6       08AC  1858         INCL    G^PMS$GL_DIR_IN
                        08B2  1859         .ENDC
                        08B2  1860
      48 A8    D5       08B2  1861         TSTL    RSB$L_PARENT(R8)        ; Directory entries must be roots
         17    12       08B5  1862         BNEQ    40$                     ; Error
                        08B7  1863
                        08B7  1864 RESPOND_RESEND:
         53 8ED0        08B7  1865         POPL    R3                      ; Restore CSID of incoming message
         55 8ED0        08BA  1866         POPL    R5                      ; Restore CDRP address
      54 53    D1       08BD  1867         CMPL    R3,R4                   ; Is requestor managing resource?
         4A    13       08C0  1868         BEQL    RESPOND_DOLOCL          ; Yes, do locally (on remote system)
   2C A5 54    D0       08C2  1869         MOVL    R4,CDRP$L_VAL1(R5)      ; No, return CSID of other system
      F8 8F    90       08C6  1870         MOVB    #LKB$K_RSPRESEND,-      ; Set response state
      32 A5             08C9  1871                 CDRP$L_VAL2+2(R5)
       0054    31       08CB  1872         BRW     NOTQUEUED_EXIT
                        08CE  1873
                        08CE  1874 40$:     BUG_CHECK       LOCKMGRERR,FATAL
                        08D2  1875
                        08D2  1876 RELOOKUP:
                        08D2  1877         ; Send a response telling requesting system to resend this
                        08D2  1878         ; request to directory system (CSID in R4).  Note that it is
                        08D2  1879         ; possible for the requesting system to be the directory system
                        08D2  1880         ; for this resource in which case a "do locally" response is sent.
                        08D2  1881         ; This situation is caused by the following race:
                        08D2  1882         ;       o  This system masters the resource
                        08D2  1883         ;       o  The other system (directory system) sends a request here
                        08D2  1884         ;       o  This system removes the directory entry
                        08D2  1885         ;       o  This system receives the request and returns "do locally"
                        08D2  1886         ; The danger with this race is that the other system doesn't realize
                        08D2  1887         ; it has a directory entry so doesn't set the DIRENTRY bit.  What
                        08D2  1888         ; saves us is that it is already set and LCK$RCV_RMVDIR will not
                        08D2  1889         ; delete the RSB.
                        08D2  1890
         6B    D4       08D2  1891         CLRL    RSB$L_HSHCHN(R11)       ; Remove RSB from hash chain
      50 58    D0       08D4  1892         MOVL    R8,R0
00000000'GF    16       08D7  1893         JSB     G^EXE$DEANONPAGED       ; Deallocate RSB
         D8    11       08DD  1894         BRB     RESPOND_RESEND
                        08DF  1895
                        08DF  1896 DIR_ENTER:
                        08DF  1897         ; If we are the directory system for this resource, then handle
                        08DF  1898         ; this request by making a directory entry.  Otherwise, tell requesting
```

```
                                0DF   1899          ; system to re-lookup the request in the directory as we've been
                                0DF   1900          ; bagged by a directory lookup race.
                                0DF   1901
        53   00000000'GF    D0  0DF   1902          MOVL    G^LCK$GL_DIRVEC,R3      ; Get address of directory vector
             51    44 A8    3C  0E6   1903          MOVZWL  RSB$W_HASHVAL(R8),R1    ; Get hash value
                   52       D4  0EA   1904          CLRL    R2                      ; Clear high order hash value
  51    50   51    F4 A3    7B  0EC   1905          EDIV    -12(R3),R1,R0,R1        ; Compute hash index (in R1)
              54     6341    D0  0F2   1906          MOVL    (R3)[R1],R4             ; Get directory system
                       DA    12  0F6   1907          BNEQ    RELOOKUP                ; It's not us
                       01    A8  0F8   1908          BISW    #RSB$M_DIRENTRY,-       ; Set directory entry bit
                    0E A8          0FA   1909                  RSB$W_STATUS(R8)
                                0FC   1910
                         00000002  0FC   1911          .IF NE  CA$_MEASURE
                 00000000'GF    D6  0FC   1912          INCL    G^PMS$GL_DIR_IN
                                902   1913          .ENDC
                                902   1914
                       53 8ED0  902   1915          POPL    R3                      ; Restore CSID
                       55 8ED0  905   1916          POPL    R5                      ; Restore CDRP address
                38 A8    53    D0  908   1917          MOVL    R3,RSB$L_CSID(R8)       ; Store CSID in RSB to form dir. entry
                                90C   1918
                                90C   1919 RESPOND_DOLOCL:
                                90C   1920          ; Send a response that tells the requestor to handle the lock request
                                90C   1921          ; locally.
                                90C   1922
                    F9 8F    90  90C   1923          MOVB    #LKB$K_RSPDOLOCL,-      ; Store response state
                    32 A5          90F   1924                  CDRP$L_VAL2+2(R5)
                       0F    11  911   1925          BRB     NOTQUEUED_EXIT
                                913   1926
                                913   1927 RESPOND_NOTQED:
                                913   1928          ; Lock was not queued due to an error.  Status is in R0
                                913   1929          ;  (e.g. SS$_NOTQUEUED)
                                913   1930
                       53 8ED0  913   1931          POPL    R3                      ; Restore CSID
                       55 8ED0  916   1932          POPL    R5                      ; Restore CDRP address
                30 A5    50    B0  919   1933          MOVW    R0,CDRP$L_VAL2(R5)      ; Store error status
                     FC 8F    90  91D   1934          MOVB    #LKB$K_RSPNOTQED,-      ; Store response state
                     32 A5          920   1935                  CDRP$L_VAL2+2(R5)       ; and fall thru to ...
                                922   1936
                                922   1937 NOTQUEUED_EXIT:
                0A5A'CF    9E  922   1938          MOVAB   W^BLD_RSPMSG,-          ; Store address of message build
                     4C A5          926   1939                  CDRP$L_MSGBLD(R5)       ; routine
                                928   1940 QUEUED_EXIT:
  50    00000000'GF    D0  928   1941          MOVL    G^CLU$GL_CLUB,R0        ; Get address of CLUB
                00AC C0    B0  92F   1942          MOVW    CLUB$W_MEMSEQ(R0),-     ; Store membership seq. num.
                     34 A5          933   1943                  CDRP$L_VAL3(R5)
                0FC6 8F    BA  935   1944          POPR    #^M<R1,R2,R6,R7,R8,R9,R10,R11>
                   F6C4'    30  939   1945          BSBW    CNX$SEND_MSG_RESP       ; Send message response
           0000'8F    50    B1  93C   1946          CMPW    R0,#SS$_NOSUCHNODE      ; Is status SS$_NOSUCHNODE?
                       09    13  941   1947          BEQL    60$                     ; Yes, error
                50    55    D0  943   1948          MOVL    R5,R0                   ; No, SS$_NODELEAVE is okay
            00000000'GF    17  946   1949          JMP     G^EXE$DEANONPAGED       ; Deallocate CDRP
                                94C   1950
                                94C   1951 60$:     BUG_CHECK       LOCKMGRERR,FATAL; CSID invalid
                                950   1952
                                950   1953          .DSABL  LSB
```

```
                     0950  1955 ;*******************************************************************************
                     0950  1956 ;
                     0950  1957 ;                       New Resource
                     0950  1958 ;
                     0950  1959 ;*******************************************************************************
                     0950  1960
                     0950  1961         ; Resource was not found so it must be created.  If it's a root
                     0950  1962         ; resource and we are the directory system, then a directory
                     0950  1963         ; entry will be made but the lock will be handled by the remote
                     0950  1964         ; system.  If it's a root resource but we are not the directory
                     0950  1965         ; system then the remote system will be told to re-lookup this
                     0950  1966         ; resource in the directory.  If it's not a root resource then the
                     0950  1967         ; lock request is handled here.  Registers contain:
                     0950  1968         ;
                     0950  1969         ;       R6      Address of parent LKB (or 0)
                     0950  1970         ;       R7      Address of parent RSB (or 0)
                     0950  1971         ;       R9      Address of input message
                     0950  1972         ;       R10     Length of resource name+8
                     0950  1973         ;       R11     Address of last RSB in resource hash chain
                     0950  1974
                     0950  1975 NEW_RESOURCE:
  51   48 AA    9E  0950  1976         MOVAB   RSB$K_LENGTH-8(R10),R1  ; Compute size of RSB
00000000'GF    16  0954  1977         JSB     G^EXE$ALONONPAGED       ; Allocate RSB
     4B 50     E9  095A  1978         BLBC    R0,40$                  ; Didn't get one
  08 A2   51   3C  095D  1979         MOVZWL  R1,RSB$W_SIZE(R2)       ; Store size and zero depth
  0A A2   36   90  0961  1980         MOVB    #DYN$C_RSB,RSB$B_TYPE(R2) ; Store type
     58   52   D0  0965  1981         MOVL    R2,R8                   ; R8 points to RSB
                     0968  1982
                     0968  1983         ; Initialize various fields in RSB
                     0968  1984
                     0968  1985         ASSUME  RSB$B_CGMODE  EQ  RSB$B_GGMODE+1
                     0968  1986         ASSUME  RSB$W_STATUS  EQ  RSB$B_CGMODE+1
                     0968  1987         ASSUME  RSB$W_REFCNT  EQ  RSB$L_VALSEQNUM+4
                     0968  1988         ASSUME  RSB$W_BLKASTCNT  EQ RSB$W_REFCNT+2
                     0968  1989         ASSUME  RSB$L_CVTQFL  EQ  RSB$L_GRQFL+8
                     0968  1990         ASSUME  RSB$L_WTQFL  EQ  RSB$L_CVTQFL+8
                     0968  1991         ASSUME  RSB$W_RQSEQNM  EQ  RSB$W_HASHVAL+2
                     0968  1992
     6B   58   D0  0968  1993         MOVL    R8,RSB$L_HSHCHN(R11)    ; Make last RSB point to this one
          68   D4  096B  1994         CLRL    RSB$L_HSHCHN(R8)        ; This one ends the chain
04 A8     5B   D0  096D  1995         MOVL    R11,RSB$L_HSHCHNBK(R8)  ; Point to previous one
50   10 A8     DE  0971  1996         MOVAL   RSB$L_GRQFL(R8),R0      ; Initialize all three queue headers
     51   50   D0  0975  1997         MOVL    R0,R1
     81   50   D0  0978  1998         MOVL    R0,(R1)+                ; Granted queue
     81   80   7E  097B  1999         MOVAQ   (R0)+,(R1)+
     81   50   D0  097E  2000         MOVL    R0,(R1)+                ; Conversion queue
     81   80   7E  0981  2001         MOVAQ   (R0)+,(R1)+
     81   50   D0  0984  2002         MOVL    R0,(R1)+                ; Waiting queue
     61   50   D0  0987  2003         MOVL    R0,(R1)
        28 A8   7C  098A  2004         CLRQ    RSB$Q_VALBLK(R8)        ; Clear value block
        30 A8   7C  098D  2005         CLRQ    RSB$Q_VALBLK+8(R8)
        3C A8   7C  0990  2006         CLRQ    RSB$L_VALSEQNUM(R8)     ; Clear value block seq. number,
                    0993  2007                                         ; ref. count and blocking AST cnt.
        0C A8   D4  0993  2008         CLRL    RSB$B_GGMODE(R8)        ; Clear modes, status
        0E A9   3C  0996  2009         MOVZWL  LKMSG$W_HASHVAL(R9),-   ; Store hash value, clear request
        44 A8       0999  2010                 RSB$W_HASHVAL(R8)       ; sequence number
            5A   28  099B  2011         MOVC3   R10,-                   ; Move parent RSB address, group number,
```

```
                                09A1  2012                    LKMSG$L_PARMSTLKID(R9),-; access mode, res. name length and
                48 A8   24 A9   09A1  2013                    RSB$L_PARENT(R8)        ; resource name to RSB
                                09A1  2014
                                09A1  2015            ; If this is a root resource, then just create a directory entry
                                09A1  2016            ; and don't build a LKB.
                                09A1  2017
                   57   D5      09A1  2018            TSTL    R7                      ; Is this a root resource?
                   0A   12      09A3  2019            BNEQ    60$                     ; No
                 FF37   31      09A5  2020            BRW     DIR_ENTER               ; Yes
                                09A8  2021
                 006B   31      09A8  2022 40$:       BRW     BLDLKB_ERROR2           ; Branch extender
                                09AB  2023
                                09AB  2024 50$:       ; Received a lock request for a lock whose parent is not managed
                                09AB  2025            ; by this system.
                                09AB  2026
                                09AB  2027            BUG_CHECK       LOCKMGRERR,FATAL
                                09AF  2028
            0B A7    01   81    09AF  2029 60$:       ADDB3   #1,RSB$B_DEPTH(R7),-    ; Set depth
                  0B A8         09B3  2030                    RSB$B_DEPTH(R8)
                  40 A7   B6    09B5  2031            INCW    RSB$W_REFCNT(R7)        ; Incr. parent's reference count
                  38 A7   D0    09B8  2032 65$:       MOVL    RSB$L_CSID(R7),-        ; Clear CSID by copying parent's
                  38 A8         09BB  2033                    RSB$L_CSID(R8)          ; and verifying it's zero
                     EC   12    09BD  2034            BNEQ    50$                     ; Error
                                09BF  2035
                                09BF  2036            ; Build the LKB.
                                09BF  2037
                   00000002     09BF  2038            .IF NE  CA$_MEASURE
          00000000'GF   D6      09BF  2039            INCL    G^PMS$GL_ENQNEW_IN
                                09C5  2040            .ENDC
                                09C5  2041
                   010F   30    09C5  2042            BSBW    BUILD_LKB
                  38 50   E9    09C8  2043            BLBC    R0,BLDLKB_ERROR         ; Unable to build LKB.  Status in R0
           58 A6     6E   D0    09CB  2044            MOVL    (SP),LKB$L_CSID(R6)     ; Store CSID or requesting system
                09 A9   91    09CF  2045            CMPB    CLSMSG$B_FUNC(R9),-     ; Is this a rebuild lock message
                     08         09D2  2046                    #LKMSG$K_REBLDLOCK      ; (only sent during failover)?
                     29   13    09D3  2047            BEQL    REBLD_LOCK              ; Yes
                                09D5  2048
                                09D5  2049            ; Grant the lock.
                                09D5  2050
           51     34 A6   9A    09D5  2051            MOVZBL  LKB$B_RQMODE(R6),R1     ; Get requested mode
          00000000'GF   16      09D9  2052            JSB     G^LCK$GRANT_LOCK_ALT    ; Grant the lock
                                09DF  2053
                                09DF  2054 LOCK_GRANTED:
                   02   AA      09DF  2055            BICW    #LKB$M_DBLKAST,-        ; Got set spuriously by common code
                  2A A6         09E1  2056                    LKB$W_STATUS(R6)
           50     FA 8F   90    09E3  2057            MOVB    #LKB$K_RSPGRANTD,R0     ; Set response state
                                09E7  2058
                                09E7  2059 LOCK_QUEUED:
                                09E7  2060            ; Lock response state is in R0.  It is either RSPGRANTD or
                                09E7  2061            ; RSPQUEUED.  I.e. it is one of the states that actually
                                09E7  2062            ; built and queued a LKB.  Send response message.
                                09E7  2063
                      53 8ED0   09E7  2064            POPL    R3                      ; Restore CSID
                      55 8ED0   09EA  2065            POPL    R5                      ; Restore CDRP address
                0A6B'CF   9E    09ED  2066            MOVAB   W^BLD_LOCKRSP,-         ; Store address of message build routine
                   4C A5        09F1  2067                    CDRP$L_MSGBLD(R5)
                56   D0   09F3  2068            MOVL    R6,CDRP$L_VAL1(R5)      ; Store LKB address
```

```
                32 A5    50   90  09F7  2069          MOVB    R0,CDRP$L_VAL2+2(R5)    ; Store response state
                     FF2A   31   09FB  2070          BRW     QUEUED_EXIT
                            09FE  2071
                            09FE  2072 REBLD_LOCK:
                            09FE  2073          ; Rebuild lock (only performed during failover)
                            09FE  2074
                     F5FF'  30   09FE  2075          BSBW    LCK$REBLD_LOCK          ; Rebuild the lock
                       DC   11   0A01  2076          BRB     LOCK_GRANTED            ; Send the response
                            0A03  2077
                            0A03  2078 BLDLKB_ERROR:
                            0A03  2079          ; Error status in R0.  Parent RSB address in R7.
                            0A03  2080          ; Address of previous RSB in hash chain in R11.  Deallocate RSB.
                            0A03  2081
                         50 DD  0A03  2082          PUSHL   R0                      ; Save R0
                      40 A7 B7  0A05  2083          DECW    RSB$W_REFCNT(R7)        ; Decrement's parent's reference count
                         6B D4  0A08  2084          CLRL    RSB$L_HSHCHN(R11)       ; Remove from hash chain
                   50    58 D0  0A0A  2085          MOVL    R8,R0
               00000000'GF   16  0A0D  2086          JSB     G^EXE$DEANONPAGED       ; Deallocate RSB
                      50 8ED0  0A13  2087          POPL    R0                      ; Restore R0
                            0A16  2088
                            0A16  2089 BLDLKB_ERROR2:
                            0A16  2090          ; Error handling depends on whether this is a rebuild lock
                            0A16  2091          ; (during failover) or a lock request during normal operation.
                            0A16  2092          ; During normal operation, all errors other than SS$_INSFMEM
                            0A16  2093          ; are sent back to the other system.  SS$_INSFMEM is handled
                            0A16  2094          ; by breaking our connection as a form of waiting.
                            0A16  2095          ; During failover, we bugcheck here with an appropriate
                            0A16  2096          ; type of resource exhausted bugcheck.
                            0A16  2097
                      09 A9 91  0A16  2098          CMPB    CLSMSG$B_FUNC(R9),-     ; Is this a rebuild lock message?
                         08             0A19  2099                  #LKMSG$K_REBLDLOCK
                         24   13  0A1A  2100          BEQL    40$                     ; Yes
               0000'8F   50   B1  0A1C  2101          CMPW    R0,#SS$_INSFMEM         ; Is it insufficient memory?
                         03   13  0A21  2102          BEQL    30$                     ; Yes
                      FEED   31  0A23  2103          BRW     RESPOND_NOTQED          ; No, send error back
                      53 8ED0  0A26  2104 30$:       POPL    R3                      ; Restore CSID
                      55 8ED0  0A29  2105          POPL    R5                      ; Restore CDRP address
                   50    55 D0  0A2C  2106          MOVL    R5,R0
               00000000'GF   16  0A2F  2107          JSB     G^EXE$DEANONPAGED       ; Deallocate CDRP
                   0FC6 8F   BA  0A35  2108          POPR    #^M<R1,R2,R6,R7,R8,R9,R10,R11>
                   53   51   D0  0A39  2109          MOVL    R1,R3                   ; Move CSB address
                      F5C1'  30  0A3C  2110          BSBW    CNX$RCV_REJECT          ; Reject this message
                            05  0A3F  2111          RSB
                            0A40  2112
                            0A40  2113 40$:        ; Unable to rebuild a lock during failover.  Bugcheck
                            0A40  2114          ; with a (hopefully) useful bugcheck message.
                            0A40  2115
               0000'8F   50   B1  0A40  2116          CMPW    R0,#SS$_INSFMEM         ; Is it insufficient non-paged pool?
                         04   12  0A45  2117          BNEQ    50$                     ; No
                            0A47  2118          BUG_CHECK       INSFPOOL,FATAL  ; Yes
               0000'8F   50   B1  0A4B  2119 50$:       CMPW    R0,#SS$_NOLOCKID        ; Is it no lockids?
                         04   12  0A50  2120          BNEQ    60$                     ; No
                            0A52  2121          BUG_CHECK       INSFLOCKID,FATAL; Yes
                            0A56  2122 60$:        BUG_CHECK       LOCKMGRERR,FATAL; Unknown error (status in R0)
                            0A5A  2123
```

```
                                0A5A  2125 ;*****************************************************************************
                                0A5A  2126 ;
                                0A5A  2127 ;                   Message Build Routines
                                0A5A  2128 ;
                                0A5A  2129 ;*****************************************************************************
                                0A5A  2130
                                0A5A  2131         ; BLD_RSPMSG is a generic message building routine that simply
                                0A5A  2132         ; copies a fixed quadword from the CDRP into the message
                                0A5A  2133         ; buffer.  Since this is a response message, the RSPID sent
                                0A5A  2134         ; to us is returned.  Registers contain:
                                0A5A  2135         ;       R2      Address of message buffer
                                0A5A  2136         ;       R5      Address of CDRP
                                0A5A  2137
                                0A5A  2138         ASSUME  LKMSG$B_STATE  EQ  LKMSG$L_CSID+6
                                0A5A  2139
                                0A5A  2140 BLD_RSPMSG:
                                0A5A  2141         ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
        08 A2    0182 8F    B0  0A5A  2142         MOVW    #LKMSG$K_NEWLOCK@8-     ; Store function and facility codes
                                0A60  2143                 !CLSMSG$M_RESPMSG-
                                0A60  2144                 !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
              2C A5    7D      0A60  2145         MOVQ    CDRP$L_VAL1(R5),-       ; Copy value into message buffer
              18 A2            0A63  2146                 LKMSG$L_CSID(R2)
              34 A5    B0      0A65  2147         MOVW    CDRP$L_VAL3(R5),-       ; Store membership seq. num.
              0C A2            0A68  2148                 LKMSG$W_MEMSEQ(R2)
                       05      0A6A  2149         RSB
                                0A6B  2150
                                0A6B  2151
                                0A6B  2152         ; BLD_LOCKRSP builds a response message for lock requests.
                                0A6B  2153         ; This response message is only used if this system actually
                                0A6B  2154         ; created (and queued) a LKB.  In all other cases, (e.g.
                                0A6B  2155         ; directory lookups) a simple response is built by BLD_RSPMSG.
                                0A6B  2156         ; Registers contain:
                                0A6B  2157         ;       R2      Address of message buffer
                                0A6B  2158         ;       R3      Address of CSB
                                0A6B  2159         ;       R5      Address of CDRP
                                0A6B  2160         ;       CDRP$L_VAL1 contains address of the LKB
                                0A6B  2161         ; All registers except R0 and R1 must be preserved
                                0A6B  2162         ;
                                0A6B  2163         ; Note:  This message build routine uses pointers (to LKB and RSB)
                                0A6B  2164         ; that will not be valid if we started a state change.  Therefore,
                                0A6B  2165         ; it is necessary to validiate the membership sequence number
                                0A6B  2166         ; before using any pointers stored in the CDRP.
                                0A6B  2167
                                0A6B  2168 BLD_LOCKRSP:
                                0A6B  2169         ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
    08 A2    0182 8F    B0  0A6B  2170         MOVW    #LKMSG$K_NEWLOCK@8-     ; Store function and facility codes
                                0A71  2171                 !CLSMSG$M_RESPMSG-
                                0A71  2172                 !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
              34 A5    B0      0A71  2173         MOVW    CDRP$L_VAL3(R5),-       ; Store membership seq. num.
              0C A2            0A74  2174                 LKMSG$W_MEMSEQ(R2)
           51 64 A3    D0      0A76  2175         MOVL    CSB$L_CLUB(R3),R1       ; Get address of CLUB
           00AC C1    B1      0A7A  2176         CMPW    CLUB$W_MEMSEQ(R1),-     ; Have we done a state change?
              34 A5            0A7E  2177                 CDRP$L_VAL3(R5)
                 3C    12      0A80  2178         BNEQ    20$                     ; Yes
           50 2C A5    D0      0A82  2179         MOVL    CDRP$L_VAL1(R5),R0      ; Get address of LKB
              30 A0    D0      0A86  2180         MOVL    LKB$L_LKID(R0),-        ; Store master lockid
              10 A2            0A89  2181                 LKMSG$L_MSTLKID(R2)
```

```
                54 A0   D0  0A8B  2182          MOVL    LKB$L_REMLKID(R0),-     ; Store process lockid
                14 A2       0A8E  2183                  LKMSG$L_PRCLKID(R2)
                32 A5   90  0A90  2184          MOVB    CDRP$L_VAL2+2(R5),-     ; Store response state
                1E A2       0A93  2185                  LKMSG$B_STATE(R2)
                2A A0   B0  0A95  2186          MOVW    LKB$W_STATUS(R0),-      ; Store LKB status
                18 A2       0A98  2187                  LKMSG$W_LKBSTATUS(R2)
                            0A9A  2188          ASSUME  LKB$K_GRANTED  GT  0
                            0A9A  2189          ASSUME  LKB$K_WAITING  LT  0
                36 A0   95  0A9A  2190          TSTB    LKB$B_STATE(R0)         ; Is lock granted?
                    19  15  0A9D  2191          BLEQ    10$                     ; No
            51  50 A0   D0  0A9F  2192          MOVL    LKB$L_RSB(R0),R1        ; Get RSB address
                28 A1   7D  0AA3  2193          MOVQ    RSB$Q_VALBLK(R1),-      ; Store value block
                20 A2       0AA6  2194                  LKMSG$Q_VALBLK(R2)
                30 A1   7D  0AA8  2195          MOVQ    RSB$Q_VALBLK+8(R1),-
                28 A2       0AAB  2196                  LKMSG$Q_VALBLK+8(R2)
                3C A1   D0  0AAD  2197          MOVL    RSB$L_VALSEQNUM(R1),-   ; Store sequence number
                30 A2       0AB0  2198                  LKMSG$L_VALSEQNUM(R2)
                0E A1   B0  0AB2  2199          MOVW    RSB$W_STATUS(R1),-      ; Store RSB status
                1A A2       0AB5  2200                  LKMSG$W_RSBSTATUS(R2)
                        05  0AB7  2201          RSB
                            0AB8  2202
                10 A0   B0  0AB8  2203  10$:    MOVW    LKB$W_RQSEQNM(R0),-     ; Store request sequence number
                30 A2       0ABB  2204                  LKMSG$W_RQSEQNM(R2)
                        05  0ABD  2205          RSB
                            0ABE  2206
                            0ABE  2207  20$:    ; We've done a membership state change between our SEND_MSG
                            0ABE  2208          ; and calling the message build routine.  Just send back a RETRY
                            0ABE  2209          ; status.
                            0ABE  2210
                FE 8F   90  0ABE  2211          MOVB    #LKB$K_RETRY,-          ; Store RETRY status
                1E A2       0AC1  2212                  LKMSG$B_STATE(R2)
                        05  0AC3  2213          RSB
```

```
                        0AC4  2215          .SBTTL  BUILD_LKB - Build lock block
                        0AC4  2216
                        0AC4  2217 ;++
                        0AC4  2218 ; FUNCTIONAL DESCRIPTION:
                        0AC4  2219 ;
                        0AC4  2220 ;       This routine builds LKBs for received lock requests that are
                        0AC4  2221 ;       being handled by this system.
                        0AC4  2222 ;
                        0AC4  2223 ; CALLING SEQUENCE:
                        0AC4  2224 ;
                        0AC4  2225 ;       BSBW    BUILD_LKB
                        0AC4  2226 ;
                        0AC4  2227 ; INPUT PARAMETERS:
                        0AC4  2228 ;
                        0AC4  2229 ;       R6      Address of parent LKB (or 0)
                        0AC4  2230 ;       R8      Address of RSB
                        0AC4  2231 ;       R9      Address of input message
                        0AC4  2232 ;
                        0AC4  2233 ; IMPLICIT INPUTS:
                        0AC4  2234 ;
                        0AC4  2235 ;       Various fields in the input message are used
                        0AC4  2236 ;
                        0AC4  2237 ; OUTPUT PARAMETERS:
                        0AC4  2238 ;
                        0AC4  2239 ;       R0      Completion code
                        0AC4  2240 ;       R6      Address of LKB
                        0AC4  2241 ;
                        0AC4  2242 ; COMPLETION CODES:
                        0AC4  2243 ;
                        0AC4  2244 ;       SS$_NORMAL      Successful completion
                        0AC4  2245 ;       SS$_INSFMEM     Insufficent memory to allocate LKB
                        0AC4  2246 ;       SS$_NOLOCKID    No lock ids
                        0AC4  2247 ;
                        0AC4  2248 ; SIDE EFFECTS:
                        0AC4  2249 ;
                        0AC4  2250 ;       R0 - R4 are destroyed
                        0AC4  2251 ;--
                        0AC4  2252
                        0AC4  2253
                        0AC4  2254 NO_LOCKIDS:
                        0AC4  2255          ; No lock ids.  Deallocate LKB and try to extend table.
                        0AC4  2256
            50    52    D0  0AC4  2257          MOVL    R2,R0                   ; Address of LKB
     00000000'GF    16    0AC7  2258          JSB     G^EXE$DEANONPAGED       ; Deallocate it
     00000000'GF    16    0ACD  2259          JSB     G^LCK$EXTEND_IDTBL      ; Extend table
            01 50    E8  0AD3  2260          BLBS    R0,BUILD_LKB            ; Success, repeat routine
                        0AD6  2261 BUILD_LKB_ERR:
                    05  0AD6  2262          RSB
                        0AD7  2263
                        0AD7  2264 BUILD_LKB:
          51  0060 8F    3C  0AD7  2265          MOVZWL  #LKB$K_LENGTH,R1        ; Size of LKB
     00000000'GF    16    0ADC  2266          JSB     G^EXE$ALONONPAGED       ; Allocate it
            F1 50    E9  0AE2  2267          BLBC    R0,BUILD_LKB_ERR                ; Unable to allocate it
08 A2   00350060 8F    D0  0AE5  2268          MOVL    #<DYN$C_LKB@16>!-       ; Store size and type
                        0AED  2269                  LKB$K_LENGTH,LKB$W_SIZE(R2)
            0C A2    D4  0AED  2270          CLRL    LKB$L_PID(R2)           ; Clear PID
     50     00000000'GF    D0  0AF0  2271          MOVL    G^LCK$GL_NXTID,R0       ; Get next lock id
```

```
                CB   13   0AF7  2272          BEQL    NO_LOCKIDS                      ; No more - return error
51   00000C00'GF   D0   0AF9  2273          MOVL    G^LCK$GL_IDTBL,R1               ; Get address of lock id table.  *** May
                   0B00  2274                                                        ; combine with next instr. if no loading
      30 A2   50   B0   0B00  2275          MOVW    R0,LKB$L_LKID(R2)               ; Store lockid index
       51   6140   DE   0B04  2276          MOVAL   (R1)[R0],R1                     ; Get address of lockid table entry
00000000'GF   61   3C   0B08  2277          MOVZWL  (R1),G^LCK$GL_NXTID             ; Update ptr to next free id
   32 A2   02 A1   B0   0B0F  2278          MOVW    2(R1),LKB$L_LRID+2(R2)          ; Store lockid sequence number
       61   52   D0   0B14  2279          MOVL    R2,(R1)                         ; Store LKB address in table entry
      50 A2   58   D0   0B17  2280          MOVL    R8,LKB$L_RSB(R2)                ; Make LKB point to RSB
      48 A2   56   D0   0B1B  2281          MOVL    R6,LKB$L_PARENT(R2)             ; Point to parent LKB
            03   13   0B1F  2282          BEQL    10$                             ; No parent
         4C A6   B6   0B21  2283          INCW    LKB$W_REFCNT(R6)                ; Incr. parent's reference count
                   0B24  2284
                   0B24  2285 10$:       ; Build portion of LKB that comes from input message
                   0B24  2286
                   0B24  2287          ASSUME  LKMSG$W_FLAGS  EQ  LKMSG$L_PRCLKID+4
                   0B24  2288          ASSUME  LKMSG$B_RQMODE  EQ  LKMSG$W_FLAGS+2
                   0B24  2289          ASSUME  LKMSG$B_GRMODE  EQ  LKMSG$B_RQMODE+1
                   0B24  2290          ASSUME  LKMSG$L_BLKASTFLG  EQ  LKMSG$B_GRMODE+1
                   0B24  2291
      56   52   D0   0B24  2292          MOVL    R2,R6                           ; Use R6 for LKB from now on
   53   14 A9   9E   0B27  2293          MOVAB   LKMSG$L_PRCLKID(R9),R3          ; Point into message
   54 A6   83   D0   0B2B  2294          MOVL    (R3)+,LKB$L_REMLKID(R6)         ; Store process lock id
   28 A6   83   B0   0B2F  2295          MOVW    (R3)+,LKB$W_FLAGS(R6)           ; Store flags
   34 A6   83   B0   0B33  2296          MOVW    (R3)+,LKB$B_RQMODE(R6)          ; Store lock modes
   20 A6   83   D0   0B37  2297          MOVL    (R3)+,LKB$L_BLKASTADR(R6)       ; Store blocking AST flag
                   0B3B  2298
                   0B3B  2299          ; Store EPID of owner of lock unless LCK$M_CVTSYS is set in
                   0B3B  2300          ; which case, store 0.  Also store deadlock priority if not system
                   0B3B  2301          ; owned.
                   0B3B  2302
      50   D4   0B3B  2303          CLRL    R0                              ; Assume system owned
      06   E0   0B3D  2304          BBS     #LCK$V_CVTSYS,-                 ; Branch if it is system owned
09 28 A6          0B3F  2305                  LKB$W_FLAGS(R6),20$
50    10 A9   D0   0B42  2306          MOVL    LKMSG$L_EPIDNEW(R9),R0          ; Get EPID
   4C A9   D0   0B46  2307          MOVL    LKMSG$L_DLCKPRI_NEW(R9),-
   24 A6          0B49  2308                  LKB$L_DLCKPRI(R6)               ; Store deadlock priority
14 A6    50   D0   0B4B  2309 20$:     MOVL    R0,LKB$L_EPID(R6)               ; Store EPID or 0
                   0B4F  2310
                   0B4F  2311          ; Finish remainder of LKB
                   0B4F  2312
      10   B0   0B4F  2313          MOVW    #LKB$M_MSTCPY,-                 ; Store master copy flag
   2A A6          0B51  2314                  LKB$W_STATUS(R6)
      08   E1   0B53  2315          BBC     #LCK$V_PROTECT,-                ; Branch if this is not a protected lock
06 28 A6          0B55  2316                  LKB$W_FLAGS(R6),30$
 0200 8F   A8   0B58  2317          BISW    #LKB$M_PROTECT,-                ; Set PROTECT status bit
   2A A6          0B5C  2318                  LKB$W_STATUS(R6)
   4C A6   B4   0B5E  2319 30$:     CLRW    LKB$W_REFCNT(R6)                ; Clear ref. count
 50   00'   D0   0B61  2320          MOVL    S^#SS$_NORMAL,R0                ; Store success status
            05   0B64  2321          RSB
```

```
                     0B65  2323          .SBTTL  LCK$DEALLOC_LKB - Deallocate a LKB
                     0B65  2324
                     0B65  2325 ;++
                     0B65  2326 ; FUNCTIONAL DESCRIPTION:
                     0B65  2327 ;
                     0B65  2328 ;       This routine is used to deallocate a LKB and perform
                     0B65  2329 ;       additional cleanup such as deallocating the lock id. and
                     0B65  2330 ;       adjusting the parent's reference count.
                     0B65  2331 ;
                     0B65  2332 ; CALLING SEQUENCE:
                     0B65  2333 ;
                     0B65  2334 ;       BSBW    LCK$DEALLOC_LKB
                     0B65  2335 ;
                     0B65  2336 ; INPUT PARAMETERS:
                     0B65  2337 ;
                     0B65  2338 ;       R6      Address of LKB
                     0B65  2339 ;
                     0B65  2340 ; OUTPUT PARAMETERS:
                     0B65  2341 ;
                     0B65  2342 ;       None
                     0B65  2343 ;
                     0B65  2344 ; SIDE EFFECTS:
                     0B65  2345 ;
                     0B65  2346 ;       R0 - R3 destroyed
                     0B65  2347 ;--
                     0B65  2348
                     0B65  2349 LCK$DEALLOC_LKB::
                     0B65  2350          ; Remove lock from timeout queue, if it's on it.
                     0B65  2351
          06   E5    0B65  2352          BBCC    #LKB$V_TIMOUTQ,-        ; Is lock on timeout queue?
03 2A A6               0B67  2353                  LKB$W_STATUS(R6),5$
 50   66   0F         0B6A  2354          REMQUE  LKB$L_ASTQFL(R6),R0     ; Yes, remove it
                     0B6D  2355
                     0B6D  2356 5$:      ; Deallocate lock id.
                     0B6D  2357
       50   30 A6    3C  0B6D  2358          MOVZWL  LKB$L_LKID(R6),R0       ; Get lock id index
 51   00000000'GF    D0  0B71  2359          MOVL    G^LCK$GL_IDTBL,R1       ; *** Combine with next instr.
       51   6140     DE  0B78  2360          MOVAL   (R1)[R0],R1             ; Point to table entry
 61   00000000'GF    B0  0B7C  2361          MOVW    G^LCK$GL_NXTID,(R1)     ; Store next id in this id's slot
02 A1   32 A6    01    A1  0B83  2362          ADDW3   #1,LKB$L_LKID+2(R6),2(R1) ; Incr. and store sequence number
                  04    1C  0B89  2363          BVC     20$                     ; Didn't overflow to a system address
       02 A1    01    B0  0B8B  2364          MOVW    #1,2(R1)                ; Overflowed - restart seq. number at 1
00000000'GF     50    D0  0B8F  2365 20$:     MOVL    R0,G^LCK$GL_NXTID       ; This id becomes the next one
                     0B96  2366
                     0B96  2367          ; Decrement parent LKB's reference count
                     0B96  2368
       50   48 A6    D0  0B96  2369          MOVL    LKB$L_PARENT(R6),R0     ; Get address of parent
                  05    13  0B9A  2370          BEQL    30$                     ; No parent
            4C A0    B7  0B9C  2371          DECW    LKB$W_REFCNT(R0)        ; Decr. parent's reference count
                  0A    19  0B9F  2372          BLSS    80$                     ; Went negative
                     0BA1  2373
                     0BA1  2374 30$:     ; Deallocate LKB
                     0BA1  2375
          50   56    D0  0BA1  2376          MOVL    R6,R0
       00000000'GF    16  0BA4  2377          JSB     G^EXE$DEANONPAGED       ; Deallocate LKB
                  05    0BAA  2378          RSB
                     0BAB  2379
```

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

```
                                                B 4
                                .SBTTL  LCK$SND_GRANTED - Send a lock granted message

                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;
                                ;       This routine sends a lock granted message to the process system.
                                ;       We are the master system for this lock.
                                ;
                                ; CALLING SEQUENCE:
                                ;
                                ;       JSB     LCK$SND_GRANTED
                                ;       IPL must be at IPL$_SYNCH
                                ;
                                ; INPUT PARAMETERS:
                                ;
                                ;       R6      Address of LKB
                                ;       R8      Address of RSB
                                ;
                                ; OUTPUT PARAMETERS:
                                ;
                                ;       None
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;       R0 - R5 are not preserved.
                                ;--

                                LCK$SND_GRANTED::
      06   E5                   BBCC    #LKB$V_TIMOUTQ,-        ; Remove lock from timeout queue
03 2A A6                        LKB$W_STATUS(R6),10$    ; if it was on it
   50   66   0F                REMQUE  LKB$L_ASTQFL(R6),R0
      02   AA                   BICW    #LKB$M_DBLKAST,-        ; Clear deliver blocking AST bit
   2A A6                        LKB$W_STATUS(R6)        ; that was set spuriously
53 58 A6    D0                  MOVL    LKB$L_CSID(R6),R3       ; Get CSID of remote system
    F43E'   30                  BSBW    CNX$ALLOC_CDRP          ; Alloc. a CDRP and convert CSID to CSB
   25 50    E9                  BLBC    R0,80$                  ; Unable to allocate one or CSID error

                                ; Put lock ids. into CDRP and then send message.
                                ; Note:  It might seem preferable to put the LKB address into
                                ; the CDRP rather than the lock ids.  The problem with this approach
                                ; is that the LKB could be deallocated while the thread to
                                ; build and send the message was waiting for resources.
                                ; So instead, we use the lock ids. and re-validate the LKB.
                                ; If we find that the LKB has been deallocated, we no longer
                                ; need to send a message.  However, there is no way to stop
                                ; sending a message once we are inside the message build routine.
                                ; So instead, we send the message and depend on a lockid mismatch
                                ; at the remote end.

   30 A6    D0                  MOVL    LKB$L_LKID(R6),-        ; Store master lockid
   2C A5                        CDRP$L_VAL1(R5)
   54 A6    D0                  MOVL    LKB$L_REMLKID(R6),-     ; Store process lockid
   30 A5                        CDRP$L_VAL2(R5)
50 64 A3    D0                  MOVL    CSB$L_CLUB(R3),R0       ; Get address of CLUB
00AC C0     B0                  MOVW    CLUB$W_MEMSEQ(R0),-     ; Store membership seq. num.
   34 A5                        CDRP$L_VAL3(R5)
```

```
                                0BD9  2439          ; Store address of message build routine and send message
                                0BD9  2440
          0E'AF    9E           0BD9  2441          MOVAB   B^BLD_GRANTEDMSG,-      ; Store address of message
          4C A5                 0BDC  2442                  CDRP$L_MSGBLD(R5)       ; build routine
           F41F'    30          0BDE  2443          BSBW    CNX$SEND_MSG_CSB        ; Send the message
       50   55    D0            0BE1  2444          MOVL    R5,R0                   ; Address of CDRP
 00000000'GF     17            0BE4  2445          JMP     G^EXE$DEANONPAGED       ; Deallocate it
                                0BEA  2446
       54   56    D0            0BEA  2447 80$:     MOVL    R6,R4                   ; Move LKB address
                                0BED  2448
                                0BED  2449 INSERT_ON_TIMOUTQ:
                                0BED  2450          ; CSID conversion error or no CDRPs.  Queue LKB to timeout queue
                                0BED  2451          ; if insufficient memory.  LKB address is in R4.
                                0BED  2452
  0000'8F   50    B1            0BED  2453          CMPW    R0,#SS$_INSFMEM         ; Is it insufficient memory?
            16    12            0BF2  2454          BNEQ    90$                     ; No
       0040 8F    A8            0BF4  2455          BISW    #LKB$M_TIMOUTQ,-        ; Set timeout bit
          2A A4                 0BF8  2456                  LKB$W_STATUS(R4)
 00000000'GF      D0            0BFA  2457          MOVL    G^EXE$GL_ABSTIM,-       ; Store immediate timeout time
          18 A4                 0C00  2458                  LKB$L_DUETIME(R4)
          64    0E              0C02  2459          INSQUE  LKB$L_ASTQFL(R4),-      ; Insert at head of timeout queue
 00000000'GF                    0C04  2460                  G^LCK$GL_TIMOUTQ
                05              0C09  2461          RSB
                                0C0A  2462
                                0C0A  2463 90$:     BUG_CHECK       LOCKMGRERR,FATAL
                                0C0E  2464
                                0C0E  2465          ; Action routine to build the lock granted message.  Inputs are:
                                0C0E  2466          ;       R2      Address of message buffer
                                0C0E  2467          ;       R5      Address of CDRP
                                0C0E  2468
                                0C0E  2469 BLD_GRANTEDMSG:
             54    DD           0C0E  2470          PUSHL   R4                      ; Save registers
             56    DD           0C10  2471          PUSHL   R6
                                0C12  2472          ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
08 A2   0202 8F    B0           0C12  2473          MOVW    #LKMSG$K_GRANTED@8-     ; Store function and facility codes
                                0C18  2474                  !CLSMSG$R_FAC_LCK,CLSMSG$B_FACILITY(R2)
          34 A5    B0           0C18  2475          MOVW    CDRP$L_VAL3(R5),-       ; Store membership seq. num.
          0C A2                 0C1B  2476                  LKMSG$W_MEMSEQ(R2)
       51 2C A5    D0           0C1D  2477          MOVL    CDRP$L_VAL1(R5),R1      ; Get master lockid
       50 30 A5    D0           0C21  2478          MOVL    CDRP$L_VAL2(R5),R0      ; Get process lockid
          2C A5    7D           0C25  2479          MOVQ    CDRP$L_VAL1(R5),-       ; Store lockids
          10 A2                 0C28  2480                  LKMSG$L_MSTLKID(R2)
           03AA    30           0C2A  2481          BSBW    VERIFYREMLKID
          22 50    E9           0C2D  2482          BLBC    R0,60$                  ; Branch if LKB went away
       50 50 A6    D0           0C30  2483          MOVL    LKB$L_RSB(R6),R0        ; Get RSB address
                                0C34  2484
                                0C34  2485          ; Note we are only sending the low byte of the RSB status
                                0C34  2486          ; as the high byte overlaps the LKMSG$B_GRMODE field.
                                0C34  2487          ; We can get away with this because we only need to send
                                0C34  2488          ; the RSB$V_VALINVLD bit.  However, this should be fixed.
                                0C34  2489
                                0C34  2490          ASSUME  LKMSG$B_GRMODE  EQ  LKMSG$W_RSBSTATUS+1
                                0C34  2491          ASSUME  RSB$V_VALINVLD  LE  7
                                0C34  2492
          2A A6    B0           0C34  2493          MOVW    LKB$W_STATUS(R6),-      ; Store LKB status
          18 A2                 0C37  2494                  LKMSG$W_LKBSTATUS(R2)
          0E A0    90           0C39  2495          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  2509          .SBTTL  LCK$RCV_GRANTED - Receive lock granted message
                     0C59  2510
                     0C59  2511 ;++
                     0C59  2512 ; FUNCTIONAL DESCRIPTION:
                     0C59  2513 ;
                     0C59  2514 ;       This routine handles the receiving of a lock granted message
                     0C59  2515 ;       sent by the master system.  This system is the process system for
                     0C59  2516 ;       this lock.
                     0C59  2517 ;
                     0C59  2518 ; CALLING SEQUENCE:
                     0C59  2519 ;
                     0C59  2520 ;       JSB     LCK$RCV_GRANTED (called by SCS received message routine)
                     0C59  2521 ;       IPL must be at IPL$_SYNCH
                     0C59  2522 ;
                     0C59  2523 ; INPUT PARAMETERS:
                     0C59  2524 ;
                     0C59  2525 ;       R2      Address of message buffer
                     0C59  2526 ;       R3      Address of CSB
                     0C59  2527 ;
                     0C59  2528 ; OUTPUT PARAMETERS:
                     0C59  2529 ;
                     0C59  2530 ;       None
                     0C59  2531 ;
                     0C59  2532 ; SIDE EFFECTS:
                     0C59  2533 ;
                     0C59  2534 ;       R0 - R5 Destroyed
                     0C59  2535 ;--
                     0C59  2536
                     0C59  2537 LCK$RCV_GRANTED::
    0FCC 8F    BB    0C59  2538          PUSHR   #^M<R2,R3,R6,R7,R8,R9,R10,R11>
                     0C5D  2539          ASSUME  LKMSG$L_PRCLKID  EQ  LKMSG$L_MSTLKID+4
50    10 A2    7D    0C5D  2540          MOVQ    LKMSG$L_MSTLKID(R2),R0  ; Get process and master lockids
      0373     30    0C61  2541          BSBW    VERIFYREMLKID           ; Convert to LKB address
     63 50     E9    0C64  2542          BLBC    R0,40$                  ; Invalid lockid - ignore message
58    50 A6    D0    0C67  2543          MOVL    LKB$L_RSB(R6),R8        ; Get RSB address
                     0C6B  2544
                     0C6B  2545          ; Store value block unless the one we already have is later.
                     0C6B  2546          ; Note we are only sending the low byte of the RSB status
                     0C6B  2547          ; as the high byte overlaps the LKMSG$B_GRMODE field.
                     0C6B  2548          ; We can get away with this because we only need to send
                     0C6B  2549          ; the RSB$V_VALINVLD bit.  However, this should be fixed.
                     0C6B  2550
                     0C6B  2551          ASSUME  RSB$V_VALINVLD  LE  7
                     0C6B  2552
     3C A8    C3    0C6B  2553          SUBL3   RSB$L_VALSEQNUM(R8),-   ; Compare sequence numbers
50   30 A2          0C6E  2554                  LKMSG$L_VALSEQNUM(R2),R0
       1C    15    0C71  2555          BLEQ    10$                     ; We already have a newer one
     20 A2    7D    0C73  2556          MOVQ    LKMSG$Q_VALBLK(R2),-    ; Store value block
     28 A8          0C76  2557                  RSB$Q_VALBLK(R8)
     28 A2    7D    0C78  2558          MOVQ    LKMSG$Q_VALBLK+8(R2),-
     30 A8          0C7B  2559                  RSB$Q_VALBLK+8(R8)
     30 A2    D0    0C7D  2560          MOVL    LKMSG$L_VALSEQNUM(R2),- ; Store sequence number
     3C A8          0C80  2561                  RSB$L_VALSEQNUM(R8)
        02    AA    0C82  2562          BICW    #RSB$M_VALINVLD,-       ; Clear value block invalid flag
     0E A8          0C84  2563                  RSB$W_STATUS(R8)
        01    E1    0C86  2564          BBC     #RSB$V_VALINVLD,-       ; and conditionally set it
  04 1A A2          0C88  2565                  LKMSG$Q_RSBSTATUS(R2),10$
```

```
                                                F 4
    02   A8  0C8B  2566          BISW    #RSB$M_VALINVLD,-
OE  A8       0C8D  2567                  RSB$W_STATUS(R8)
             0C8F  2568
             0C8F  2569 10$:     ; There are a number of race conditions that must be handled
             0C8F  2570          ; correctly when this message is received.  These are:
             0C8F  2571          ;       1)  We receive this message before the process has
             0C8F  2572          ;           been scheduled.  The LKB state is RSPQUEUED.  In this
             0C8F  2573          ;           case change the state to RSPGRANTD.
             0C8F  2574          ;       2)  The lock was dequeued before this message arrived.
             0C8F  2575          ;           In this case we will get an invalid lock id above
             0C8F  2576          ;           and will ignore this message.
             0C8F  2577          ;       3)  The lock was a conversion and it was canceled before
             0C8F  2578          ;           this message arrived.  The LKB state is GRANTED.  Ignore
             0C8F  2579          ;           this message unless the BLKASTQED bit is set.  In this
             0C8F  2580          ;           case handle this message as if it were a blocking AST
             0C8F  2581          ;           message.
             0C8F  2582          ;       4)  Case # 3 but another conversion has already been issued.
             0C8F  2583          ;           Lock state is SCSWAIT, RETRY, RSPQUEUED, or RSPGRANTD.
             0C8F  2584          ;           This condition is also determined by the granted
             0C8F  2585          ;           lock mode in the message not being equal to the requested
             0C8F  2586          ;           mode in the LKB.  Ignore this message.
             0C8F  2587
             0C8F  2588          DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
             0C8F  2589                  <-
             0C8F  2590                  <GRANTED,20$>,-         ; Ignore unless BLKASTQED set
             0C8F  2591                  <CONVERT,50$>,-         ; Handle normally
             0C8F  2592                  <WAITING,50$>,-         ; Handle normally
             0C8F  2593                  <RETRY,80$>,-           ; Ignore message
             0C8F  2594                  <SCSWAIT,80$>,-         ; Ignore message
             0C8F  2595                  <RSPQUEUED,30$>,-       ; Change state to RSPGRANTD
             0C8F  2596                  <RSPGRANTD,80$>,-       ; Ignore message
             0C8F  2597                  >
             0CA5  2598          BUG_CHECK       LOCKMGRERR,FATAL; Invalid state
             0CA9  2599
             0CA9  2600 20$:     ; The lock state is granted.  This should only occur if the lock
             0CA9  2601          ; request was canceled.  If the BLKASTQED bit is set then treat this
             0CA9  2602          ; message as a blocking AST message.
             0CA9  2603
      03     E1   0CA9  2604          BBC     #LKB$V_BLKASTQED,-      ; Branch if should not queue blocking
52 18  A2          0CAB  2605                  LKMSG$W_LKBSTATUS(R2),80$ ; AST
 0FCC  8F     BA   0CAE  2606          POPR    #^M<R2,R3,R6,R7,R8,R9,R10,R11> ; Restore registers
    00B8     31   0CB2  2607          BRW     LCK$RCV_BLKING          ; Use common code
             0CB5  2608
             0CB5  2609 30$:     ; The lock state is RSPQUEUED.  Change to RSPGRANTD and store
             0CB5  2610          ; BLKASTQED bit.
             0CB5  2611
   1B  A2    91   0CB5  2612          CMPB    LKMSG$B_GRMODE(R2),-    ; Is granted mode (in message) equal
   34  A6         0CB8  2613                  LKB$B_RQMODE(R6)        ; to requested mode (in lock)?
       44    12   0CBA  2614          BNEQ    80$                     ; No, ignore message
   FA  8F    90   0CBC  2615          MOVB    #LKB$K_RSPGRANTD,-      ; Yes, store new state
   36  A6         0CBF  2616                  LKB$B_STATE(R6)
       03    E1   0CC1  2617          BBC     #LKB$V_BLKASTQED,-      ; Branch if should not queue blocking
3A 18  A2          0CC3  2618                  LKMSG$W_LKBSTATUS(R2),80$ ; AST
       08    A8   0CC6  2619          BISW    #LKB$M_BLKASTQED,-      ; Set BLKASTQED bit
   2A  A6         0CC8  2620                  LKB$W_STATUS(R6)
       34    11   0CCA  2621 40$:     BRB     80$
             0CCC  2622
```

```
                     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_IOST1(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  2651          .SBTTL  LCK$SND_BLKING  - Send a blocking message
                     0D07  2652
                     0D07  2653 ;++
                     0D07  2654 ; FUNCTIONAL DESCRIPTION:
                     0D07  2655 ;
                     0D07  2656 ;       This routine sends a blocking message to the process system.
                     0D07  2657 ;       We are the master system for this lock.
                     0D07  2658 ;
                     0D07  2659 ; CALLING SEQUENCE:
                     0D07  2660 ;
                     0D07  2661 ;       JSB     LCK$SND_BLKING
                     0D07  2662 ;       IPL must be at IPL$_SYNCH
                     0D07  2663 ;
                     0D07  2664 ; INPUT PARAMETERS:
                     0D07  2665 ;
                     0D07  2666 ;       R5      Address of LKB blocking another lock
                     0D07  2667 ;       R6      Address of LKB being blocked
                     0D07  2668 ;
                     0D07  2669 ; OUTPUT PARAMETERS:
                     0D07  2670 ;
                     0D07  2671 ;       None
                     0D07  2672 ;
                     0D07  2673 ; SIDE EFFECTS:
                     0D07  2674 ;
                     0D07  2675 ;       R0 - R5 are not preserved
                     0D07  2676 ;--
                     0D07  2677
                     0D07  2678          .ENABL  LSB
                     0D07  2679
                     0D07  2680 LCK$SND_BLKING::
          08   A8    0D07  2681          BISW    #LKB$M_BLKASTQED,-      ; Set blocking AST queued bit
        2A A5         0D09  2682                  LKB$W_STATUS(R5)
                     0D0B  2683
                     0D0B  2684          ; If the blocking LKB (in R5) is on the same system as the
                     0D0B  2685          ; LKB being blocked (in R6) then we don't send a message as
                     0D0B  2686          ; this case is handled locally on the other system.
                     0D0B  2687          ; Also note that the comment in LCK$SND_GRANTED about the LKB
                     0D0B  2688          ; being deallocated out from under us applies here also.  However,
                     0D0B  2689          ; in this case, copying the info. is not as bad since much less
                     0D0B  2690          ; is copied.
                     0D0B  2691
    53   58 A5   D0  0D0B  2692          MOVL    LKB$L_CSID(R5),R3       ; Get CSID
            04   E1  0D0F  2693          BBC     #LKB$V_MSTCPY,-         ; Branch if blockee is local
      06 2A A6         0D11  2694                  LKB$W_STATUS(R6),10$
    58 A6   53   D1  0D14  2695          CMPL    R3,LKB$L_CSID(R6)       ; Does CSID match other lock?
            3C   13  0D18  2696          BEQL    30$                     ; Yes
       54   55   D0  0D1A  2697 10$:     MOVL    R5,R4                   ; R4 will point to LKB
                     0D1D  2698
                     0D1D  2699 SEND_BLOCKING:
                     0D1D  2700          ; Allocate a CDRP and copy lockids into it.  R3 contains CSID
                     0D1D  2701          ; and R4 points to LKB.
                     0D1D  2702
        F2E0'   30   0D1D  2703          BSBW    CNX$ALLOC_CDRP          ; Alloc. CDRP and convert CSID to CSB
      03 50     E8   0D20  2704          BLBS    R0,20$
        FEC7    31   0D23  2705          BRW     INSERT_ON_TIMOUTQ       ; None available or CSID error
      30 A4     D0   0D26  2706 20$:     MOVL    LKB$L_LKID(R4),-        ; Copy master lockid
      2C A5          0D29  2707                  CDRP$L_VAL1(R5)
```

```
                54 A4    D0  0D2B  2708          MOVL    LKB$L_REMLKID(R4),-     ; Copy process lockid
                30 A5        0D2E  2709                  CDRP$L_VAL2(R5)
           50    64 A3    D0  0D30  2710          MOVL    CSB$L_CLUB(R3),R0       ; Get address of CLUB
              00AC C0    B0  0D34  2711          MOVW    CLUB$W_MEMSEQ(R0),-     ; Store membership seq. num.
                34 A5        0D38  2712                  CDRP$L_VAL3(R5)
                35 A4    90  0D3A  2713          MOVB    LKB$B_GRMODE(R4),-      ; Copy granted mode of lock
                36 A5        0D3D  2714                  CDRP$L_VAL3+2(R5)
                             0D3F  2715
                             0D3F  2716          ; Store address of message build routine and send message
                             0D3F  2717
                   00000002  0D3F  2718          .IF NE  CA$_MEASURE
          00000000'GF    D6  0D3F  2719          INCL    G^PMS$GL_BLK_OUT
                             0D45  2720          .ENDC
                             0D45  2721
                57'AF    9E  0D45  2722          MOVAB   B^BLD_BLKINGMSG,-
                4C A5        0D48  2723                  CDRP$L_MSGBLD(R5)
                 F2B3'   30  0D4A  2724          BSBW    CNX$SEND_MSG_CSB        ; Send the message
           50    55    D0  0D4D  2725          MOVL    R5,R0                   ; Address of CDRP
          00000000'GF    17  0D50  2726          JMP     G^EXE$DEANONPAGED       ; Deallocate it
                             0D56  2727
                05           0D56  2728 30$:     RSB
                             0D57  2729
                             0D57  2730          ; Action routine to build the blocking message.  Inputs are:
                             0D57  2731          ;       R2      Address of message buffer
                             0D57  2732          ;       R5      Address of CDRP
                             0D57  2733
                             0D57  2734 BLD_BLKINGMSG:
                             0D57  2735          ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
     08 A2    0502 8F    B0  0D57  2736          MOVW    #LKMSG$K_BLKING@8-      ; Store function and facility codes
                             0D5D  2737                  !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
                2C A5    7D  0D5D  2738          MOVQ    CDRP$L_VAL1(R5),-       ; Store master and process lockids
                10 A2        0D60  2739                  LKMSG$L_MSTLKID(R2)
                34 A5    B0  0D62  2740          MOVW    CDRP$L_VAL3(R5),-       ; Store membership seq. num.
                0C A2        0D65  2741                  LKMSG$W_MEMSEQ(R2)
                36 A5    90  0D67  2742          MOVB    CDRP$L_VAL3+2(R5),-     ; Copy granted mode of lock
                1B A2        0D6A  2743                  LKMSG$B_GRMODE(R2)
                05           0D6C  2744          RSB
                             0D6D  2745
                             0D6D  2746          .DSABL  LSB
```

```
                        0D6D  2748          .SBTTL  LCK$RCV_BLKING - Receive a blocking message
                        0D6D  2749
                        0D6D  2750 ;++
                        0D6D  2751 ; FUNCTIONAL DESCRIPTION:
                        0D6D  2752 ;
                        0D6D  2753 ;       This routine handles receiving a blocking AST message from
                        0D6D  2754 ;       the master system.  This system is the process system.
                        0D6D  2755 ;
                        0D6D  2756 ; CALLING SEQUENCE:
                        0D6D  2757 ;
                        0D6D  2758 ;       JSB     LCK$RCV_BLKING (called by SCS received message routine)
                        0D6D  2759 ;       IPL must be at IPL$_SYNCH
                        0D6D  2760 ;
                        0D6D  2761 ; INPUT PARAMETERS:
                        0D6D  2762 ;
                        0D6D  2763 ;       R2      Address of message buffer
                        0D6D  2764 ;       R3      Address of CSB
                        0D6D  2765 ;
                        0D6D  2766 ; OUPUT PARAMETERS:
                        0D6D  2767 ;
                        0D6D  2768 ;       None
                        0D6D  2769 ;
                        0D6D  2770 ; SIDE EFFECTS:
                        0D6D  2771 ;
                        0D6D  2772 ;       R0 - R5 destroyed
                        0D6D  2773 ;--
                        0D6D  2774
                        0D6D  2775 LCK$RCV_BLKING::
                        0D6D  2776
          00000002      0D6D  2777          .IF NE  CA$_MEASURE
00000000'GF   D6        0D6D  2778          INCL    G^PMS$GL_BLK_IN
                        0D73  2779          .ENDC
                        0D73  2780
     004C 8F    BB      0D73  2781          PUSHR   #^M<R2,R3,R6>
                        0D77  2782          ASSUME  LKMSG$L_PRCLKID  EQ  LKMSG$L_MSTLKID+4
 50    10 A2    7D      0D77  2783          MOVQ    LKMSG$L_MSTLKID(R2),R0  ; Get master and process lockids
       0259     30      0D7B  2784          BSBW    VERIFYREMLKID           ; Verify lockid and convert to LKB addr.
       4A 50    E9      0D7E  2785          BLBC    R0,80$                  ; Lock is gone - ignore message
                        0D81  2786
                        0D81  2787          ; Queue a blocking AST unless one has been queued before
                        0D81  2788          ; (in which case we shouldn't have received the message; but
                        0D81  2789          ; there are race conditions that will cause this to happen).
                        0D81  2790          ; we will ignore that detail).  Also, state must be GRANTED
                        0D81  2791          ; at the same lock mode to get a blocking AST.  It may not be
                        0D81  2792          ; if we are in the middle of a conversion, or have canceled a
                        0D81  2793          ; conversion, for example.  Also, we have to handle the case
                        0D81  2794          ; that this message came in before the process has been awakened to see
                        0D81  2795          ; that the request was granted.
                        0D81  2796
                        0D81  2797          DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
                        0D81  2798                  <-
                        0D81  2799                  <GRANTED,50$>,-         ; Handle normally
                        0D81  2800                  <CONVERT,80$>,-         ; Ignore message
                        0D81  2801                  <WAITING,80$>,-         ; Ignore message
                        0D81  2802                  <RETRY,80$>,-           ; Ignore message
                        0D81  2803                  <SCSWAIT,80$>,-         ; Ignore message
                        0D81  2804                  <RSPQUEUED,80$>,-       ; Ignore message
```

```
                                                K 4
                     0D81  2805                     <RSPNOTQED,30$>,-       ; Set BLKASTQED bit
                     0D81  2806                     <RSPGRANTD,40$>,-       ; Set BLKASTQED bit
                     0D81  2807                     >
                     0D97  2808             BUG_CHECK       LOCKMGRERR,FATAL; Invalid state
                     0D9B  2809
                     0D9B  2810 30$:        ; Lock state is RSPNOTQED.
                     0D9B  2811
           35 A6  91 0D9B  2812             CMPB    LKB$B_GRMODE(R6),-      ; Verify granted modes are the same
           1B A2     0D9E  2813                     LKMSG$B_GRMODE(R2)
              29 12  0DA0  2814             BNEQ    80$                     ; They're not - ignore message
              07 11  0DA2  2815             BRB     45$
                     0DA4  2816
                     0DA4  2817 40$:        ; Lock state is RSPGRANTD.
                     0DA4  2818
           34 A6  91 0DA4  2819             CMPB    LKB$B_RQMODE(R6),-      ; Verify requested mode (in lock)
           1B A2     0DA7  2820                     LKMSG$B_GRMODE(R2)      ; is the same as granted mode (in msg)
              20 12  0DA9  2821             BNEQ    80$                     ; They're not - ignore message
              08 A8  0DAB  2822 45$:        BISW    #LKB$M_BLKASTQED,-      ; Set BLKASTQED bit
           2A A6     0DAD  2823                     LKB$W_STATUS(R6)
              1A 11  0DAF  2824             BRB     80$
                     0DB1  2825
                     0DB1  2826 50$:        ; Lock state is GRANTED.
                     0DB1  2827
           35 A6  91 0DB1  2828             CMPB    LKB$B_GRMODE(R6),-      ; Verify granted modes are the same
           1B A2     0DB4  2829                     LKMSG$B_GRMODE(R2)
              13 12  0DB6  2830             BNEQ    80$                     ; They're not - ignore message
              03 E0  0DB8  2831             BBS     #LKB$V_BLKASTQED,-      ; Branch if a blocking ast has been
        0E 2A A6     0DBA  2832                     LKB$W_STATUS(R6),80$    ; queued
           20 A6  D5 0DBD  2833             TSTL    LKB$L_BLKASTADR(R6)     ; Verify there is a blocking AST routine
              09 13  0DC0  2834             BEQL    80$
        55    56  D0 0DC2  2835             MOVL    R6,R5                   ; Move LKB address for subroutine call
00000000'GF    16  0DC5  2836             JSB     G^LCK$QUEUE_BLKAST      ; Queue a blocking AST
                     0DCB  2837
                     0DCB  2838 80$:        ; Deallocate message buffer and exit
                     0DCB  2839
        004C 8F    BA 0DCB  2840             POPR    #^M<R2,R3,R6>
           F22E'   31 0DCF  2841             BRW     CNX$DEALL_MSG_BUF_CSB   ; Deallocate message buffer and return
```

```
                        ODD2  2843          .SBTTL  LCK$SND_DEQ - Send dequeue message
                        ODD2  2844
                        ODD2  2845 ;++
                        ODD2  2846 ; FUNCTIONAL DESCRIPTION:
                        ODD2  2847 ;
                        ODD2  2848 ;       This routine sends a dequeue lock message to a remote system.
                        ODD2  2849 ;       This system is the process system and the remote system is the
                        ODD2  2850 ;       master system.
                        ODD2  2851 ;
                        ODD2  2852 ; CALLING SEQUENCE:
                        ODD2  2853 ;
                        ODD2  2854 ;       JSB     LCK$SND_DEQGR   (Lock is in granted state)
                        ODD2  2855 ;       JSB     LCK$SND_DEQCV   (Lock is in conversion wait state)
                        ODD2  2856 ;       JSB     LCK$SND_DEQWT   (Lock is in waiting state)
                        ODD2  2857 ;       IPL must be at IPL$_SYNCH
                        ODD2  2858 ;
                        ODD2  2859 ; INPUT PARAMETERS:
                        ODD2  2860 ;
                        ODD2  2861 ;       R3      Address of CSB (not CSID!)
                        ODD2  2862 ;       R4      Dequeue flags
                        ODD2  2863 ;       R5      Address of CDRP
                        ODD2  2864 ;       R6      Address of LKB
                        ODD2  2865 ;       R7      Final completion status to store in LKB$L_LKST1 or 0
                        ODD2  2866 ;               which indicates a default status should be used
                        ODD2  2867 ;       R8      Address of RSB
                        ODD2  2868 ;       R9      Address of value block (LCK$SND_DEQGR entry only)
                        ODD2  2869 ;
                        ODD2  2870 ; OUTPUT PARAMETERS:
                        ODD2  2871 ;
                        ODD2  2872 ;       None
                        ODD2  2873 ;
                        ODD2  2874 ; SIDE EFFECTS:
                        ODD2  2875 ;
                        ODD2  2876 ;       R0 - R5 destroyed
                        ODD2  2877 ;--
                        ODD2  2878
                        ODD2  2879          .ENABL  LSB
                        ODD2  2880
                        ODD2  2881 LCK$SND_DEQWT::
                        ODD2  2882          ; Lock is waiting
                        ODD2  2883
54   06   AA            ODD2  2884          BICW    #LCK$M_CANCEL-          ; Clear cancel
                        ODD5  2885                  !LCK$M_INVVALBLK,R4     ; and invalidate valblk flags
     59   D4            ODD5  2886          CLRL    R9                      ; Indicate no value block
     1B   11            ODD7  2887          BRB     40$
                        ODD9  2888
                        ODD9  2889 LCK$SND_DEQCV::
                        ODD9  2890          ; The lock is on the conversion queue.
                        ODD9  2891
54   04   AA            ODD9  2892          BICW    #LCK$M_INVVALBLK,R4     ; Clear invalidate valblk flag
  35 A6   90            ODDC  2893          MOVB    LKB$B_GRMODE(R6),-      ; Store granted mode in case
  37 A5                 ODDF  2894                  CDRP$[_VAL3+3(R5)       ; this is a CANCEL function
     59   D4            ODE1  2895          CLRL    R9                      ; Indicate no value block
     0F   11            ODE3  2896          BRB     40$
                        ODE5  2897
                        ODE5  2898 LCK$SND_DEQGR::
                        ODE5  2899          ; Lock is granted
```

```
                                ;  M 4
                        0DE5  2900
          59   D5   0DE5  2901          TSTL    R9                      ; Is there a value block?
          0B   13   0DE7  2902          BEQL    40$                     ; No value block
 38 A5    89   7D   0DE9  2903          MOVQ    (R9)+,CDRP$L_VAL4(R5)   ; Copy value block into CDRP
 40 A5    69   7D   0DED  2904          MOVQ    (R9),CDRP$L_VAL6(R5)
    59    01   D0   0DF1  2905          MOVL    #1,R9                   ; Set low byte non-zero
                        0DF4  2906
                        0DF4  2907 40$:     ; Put additional information for message into CDRP.
                        0DF4  2908
          00000002      0DF4  2909          .IF NE  CA$_MEASURE
00000000'GF    D6   0DF4  2910          INCL    G^PMS$GL_DEQ_OUT
00000000'GF    D7   0DFA  2911          DECL    G^PMS$GL_DEQ_LOC        ; Counter was incorrectly incremented
                        0E00  2912          .ENDC
                        0E00  2913
 36 A5    59   90   0E00  2914          MOVB    R9,CDRP$L_VAL3+2(R5)    ; Store value block flag
      54 A6    D0   0E04  2915          MOVL    LKB$L_REMLKID(R6),-     ; Store master lockid
      2C A5         0E07  2916                  CDRP$L_VAL1(R5)
      30 A6    D0   0E09  2917          MOVL    LKB$L_LKID(R6),-        ; Store process lockid
      30 A5         0E0C  2918                  CDRP$L_VAL2(R5)
 34 A5    54   B0   0E0E  2919          MOVW    R4,CDRP$L_VAL3(R5)      ; Store flags
                        0E12  2920
                        0E12  2921          ; Remove LKB from RSB and deallocate RSB, if necessary (don't
                        0E12  2922          ; do this if we are only cancelling the lock).
                        0E12  2923
 10 54    01   E0   0E12  2924          BBS     #LCK$V_CANCEL,R4,50$    ; Branch if cancelling lock
          55   DD   0E16  2925          PUSHL   R5                      ; Save address of CDRP
          53   DD   0E18  2926          PUSHL   R3                      ; Save address of CSB
 50   38 A6    0F   0E1A  2927          REMQUE  LKB$L_SQFL(R6),R0       ; Remove lock from RSB
00000000'GF    16   0E1E  2928          JSB     G^LCK$CHECK_RSB         ; Check RSB and delete if necessary
          28   BA   0E24  2929          POPR    #^M<R3,R5>              ; Restore registers
                        0E26  2930
                        0E26  2931 50$:     ; Send the message
                        0E26  2932
 50   64 A3    D0   0E26  2933          MOVL    CSB$L_CLUB(R3),R0       ; Get address of CLUB
    00AC C0    B0   0E2A  2934          MOVW    CLUB$W_MEMSEQ(R0),-     ; Store membership seq. num.
      48 A5         0E2E  2935                  CDRP$L_VAL8(R5)
      41'AF    9E   0E30  2936          MOVAB   B^BLD_DEQMSG,-          ; Address of message build routine
      4C A5         0E33  2937                  CDRP$L_MSGBLD(R5)
       F1C8'   30   0E35  2938          BSBW    CNX$SEND_MSG_CSB
                        0E38  2939
                        0E38  2940          ; Deallocate the CDRP.  This is done on success or failure
                        0E38  2941          ; since the lock is gone from this system (or CANCELed back to it's
                        0E38  2942          ; old mode) as far as failover is concerned.
                        0E38  2943
    50   55   D0   0E38  2944          MOVL    R5,R0                   ; Address of CDRP
00000000'GF    17   0E3B  2945          JMP     G^EXE$DEANONPAGED
                        0E41  2946
                        0E41  2947          .DSABL  LSB
                        0E41  2948
                        0E41  2949 BLD_DEQMSG:
                        0E41  2950          ; Action routine to build the actual dequeue lock message
                        0E41  2951          ; Inputs are:
                        0E41  2952          ;       R2      Address of message buffer
                        0E41  2953          ;       R5      Address of CDRP
                        0E41  2954          ; All registers except R0 and R1 must be preserved
                        0E41  2955
                        0E41  2956          ASSUME  LKMSG$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_VA[8(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$[_MSTLKID(R2)
          34 A5   D0     0E51  2966          MOVL    CDRP$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$[_VALBLKFLG(R2)
             0A   13     0E5B  2970          BEQL    30$                      ; 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  2977          .SBTTL  LCK$RCV_DEQ - Receive a dequeue lock message
                        0E68  2978
                        0E68  2979 ;++
                        0E68  2980 ; FUNCTIONAL DESCRIPTION:
                        0E68  2981 ;
                        0E68  2982 ;       This routine dequeues a lock on our system (the master) upon
                        0E68  2983 ;       receiving a message from the process system.
                        0E68  2984 ;
                        0E68  2985 ; CALLING SEQUENCE:
                        0E68  2986 ;
                        0E68  2987 ;       JSB     LCK$RCV_DEQ (called by the SCS received message routine)
                        0E68  2988 ;       IPL must be at IPL$_SYNCH
                        0E68  2989 ;
                        0E68  2990 ; INPUT PARAMETERS:
                        0E68  2991 ;
                        0E68  2992 ;       R2      Address of message buffer
                        0E68  2993 ;       R3      Address of CSB
                        0E68  2994 ;
                        0E68  2995 ; OUTPUT PARAMETERS:
                        0E68  2996 ;
                        0E68  2997 ;       None
                        0E68  2998 ;
                        0E68  2999 ; SIDE EFFECTS:
                        0E68  3000 ;
                        0E68  3001 ;       R0 - R5 destroyed
                        0E68  3002 ;--
                        0E68  3003
                        0E68  3004 LCK$RCV_DEQ::
              00000002  0E68  3005          .IF NE  CA$_MEASURE
  00000000'GF   D6      0E68  3006          INCL    G^PMS$GL_DEQ_IN
                        0E6E  3007          .ENDC
                        0E6E  3008
     0FFC   8F    BB    0E6E  3009          PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
   51    10   A2    D0  0E72  3010          MOVL    LKMSG$L_MSTLKID(R2),R1  ; Get master lockid
   50    14   A2    D0  0E76  3011          MOVL    LKMSG$L_PRCLKID(R2),R0  ; Get process lockid (for verify)
         015A   30      0E7A  3012          BSBW    VERIFYREMLKID           ; Convert to LKB address
        62   50   E9      0E7D  3013          BLBC    R0,80$                  ; Invalid lock id
             06   E5      0E80  3014          BBCC    #LKB$V_TIMOUTQ,-        ; Is lock on timeout queue?
      03   2A   A6          0E82  3015                  LKB$W_STATUS(R6),5$
     50     66   0F      0E85  3016          REMQUE  LKB$L_ASTQFL(R6),R0     ; Yes, remove it
   59    1C   A2    D0  0E88  3017  5$:     MOVL    LKMSG$L_VALBLKFLG(R2),R9; Is there a value block?
             04   13      0E8C  3018          BEQL    10$                     ; No
   59    20   A2    9E  0E8E  3019          MOVAB   LKMSG$Q_VALBLK(R2),R9   ; Yes, set address of value block
   54    18   A2    3C  0E92  3020  10$:    MOVZWL  LKMSG$W_FLAGS(R2),R4    ; Get user flags
 00000000'GF   16      0E96  3021          JSB     G^LCK$DEQLOCK           ; Dequeue the lock
        07   50   E9      0E9C  3022          BLBC    R0,60$                  ; Error
     0FFC   8F    BA      0E9F  3023  20$:    POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
         F15A'   31      0EA3  3024          BRW     CNX$DEALL_MSG_BUF_CSB   ; Deallocate message buffer and return
                         0EA6  3025
                         0EA6  3026  60$:    ; The dequeue failed.  The only failure allowed is SS$_CANCELGRANT
                         0EA6  3027          ; which can occur if the process copy was dequeued while in
                         0EA6  3028          ; the conversion queue while at the same time the master copied
                         0EA6  3029          ; was granted.  In this case we have to regrant the master copy
                         0EA6  3030          ; at the old mode to make it consistent with the process copy.
                         0EA6  3031
  0000'8F    50    B1    0EA6  3032          CMPW    R0,#SS$_CANCELGRANT     ; Is it this race condition?
             39    12      0EAB  3033          BNEQ    90$                     ; No - must be SS$_SUBLOCKS
```

```
                58    50 A6    D0  0EAD  3034          MOVL    LKB$L_RSB(R6),R8        ; Get address of RSB
                50    38 A6    0F  0EB1  3035          REMQUE  LKB$L_SQFL(R6),R0       ; Remove lock from granted queue
                      20 A6    D5  0EB5  3036          TSTL    LKB$L_BLKASTADR(R6)     ; Is there an old blocking AST flag?
                         03    13  0EB8  3037          BEQL    65$                     ; No
                      42 A8    B7  0EBA  3038          DECW    RSB$W_BLKASTCNT(R8)     ; Yes, adjust count
                      1C A2    D0  0EBD  3039 65$:     MOVL    LKMSG$L_BLKASTFLG(R2),- ; Store new blocking AST flag
                      20 A6                0EC0  3040          LKB$L_BLKASTADR(R6)
                51    1B A2    9A  0EC2  3041          MOVZBL  LKMSG$B_GRMODE(R2),R1   ; Get old granted mode
            00000000'GF    16  0EC6  3042          JSB     G^LCK$COMP_GGMODE       ; Compute a new group grant mode in R5
             0C A8     55    90  0ECC  3043          MOVB    R5,RSB$B_GGMODE(R8)     ; Store it
             0D A8     55    90  0ED0  3044          MOVB    R5,RSB$B_CGMODE(R8)
            00000000'GF    16  0ED4  3045          JSB     G^LCK$GRANT_LOCK        ; Grant the lock
            00000000'GF    16  0EDA  3046          JSB     G^LCK$GRANTCVTS         ; Grant any waiters
                         BD    11  0EE0  3047          BRB     20$
                               0EE2  3048
                               0EE2  3049
                               0EE2  3050 80$:     BUG_CHECK       INVLOCKID,FATAL ; Invalid lock id
                               0EE6  3051
                               0EE6  3052 90$:     BUG_CHECK       LOCKMGRERR,FATAL; Dequeue failed
```

```
                                0EEA  3054          .SBTTL  LCK$SND_RMVDIR - Send remove directory entry message
                                0EEA  3055
                                0EEA  3056 ;++
                                0EEA  3057 ; FUNCTIONAL DESCRIPTION:
                                0EEA  3058 ;
                                0EEA  3059 ;       This routine sends a message to the directory system specifying
                                0EEA  3060 ;       that a root resource directory entry be removed.  This routine
                                0EEA  3061 ;       is called when the last lock is dequeued from a root resource that
                                0EEA  3062 ;       is being managed by this system and this system is not the
                                0EEA  3063 ;       directory system for this resource (RSB$M_DIRENTRY is clear)
                                0EEA  3064 ;
                                0EEA  3065 ; CALLING SEQUENCE:
                                0EEA  3066 ;
                                0EEA  3067 ;       JSB     LCK$SND_RMVDIR
                                0EEA  3068 ;       IPL must be at IPL$_SYNCH
                                0EEA  3069 ;
                                0EEA  3070 ; INPUT PARAMETERS:
                                0EEA  3071 ;
                                0EEA  3072 ;       R8      Address of RSB
                                0EEA  3073 ;
                                0EEA  3074 ; OUTPUT PARAMETERS:
                                0EEA  3075 ;
                                0EEA  3076 ;       None
                                0EEA  3077 ;
                                0EEA  3078 ; SIDE EFFECTS:
                                0EEA  3079 ;
                                0EEA  3080 ;       R0 - R5 destroyed
                                0EEA  3081 ;--
                                0EEA  3082
                                0EEA  3083 LCK$SND_RMVDIR::
                                0EEA  3084          ; Compute CSID of directory system
                                0EEA  3085
     53   00000000'GF    D0     0EEA  3086          MOVL    G^LCK$GL_DIRVEC,R3      ; Get address of directory vector
          51    44 A8    3C     0EF1  3087          MOVZWL  RSB$W_HASHVAL(R8),R1    ; Get hash value
                52        D4     0EF5  3088          CLRL    R2                      ; Clear high order hash value
51   50   51    F4 A3    7B     0EF7  3089          EDIV    -12(R3),R1,R0,R1        ; Compute hash index (in R1)
          53     6341    D0     0EFD  3090          MOVL    (R3)[R1],R3             ; Get directory system
                   35    13     0F01  3091          BEQL    70$                     ; Error
                                0F03  3092
                 FOFA'    30     0F03  3093          BSBW    CNX$ALLOC_CDRP          ; Alloc. a CDRP and convert CSID to CSB
                33 50     E9     0F06  3094          BLBC    R0,80$                  ; No memory or CSID error
           2C A5    58     D0     0F09  3095          MOVL    R8,CDRP$L_VAL1(R5)      ; Store RSB address in CDRP
           50    64 A3     D0     0F0D  3096          MOVL    CSB$L_CLUB(R3),R0       ; Get address of CLUB
              00AC C0     B0     0F11  3097          MOVW    CLUB$W_MEMSEQ(R0),-     ; Store membership seq. num.
                34 A5            0F15  3098                  CDRP$L_VAL3(R5)
                42'AF     9E     0F17  3099          MOVAB   B^BLD_RMVDIRMSG,-       ; Store address of message build routine
                4C A5            0F1A  3100                  CDRP$L_MSGBLD(R5)
                                 0F1C  3101
                     00000002    0F1C  3102          .IF NE  CA$_MEASURE
           00000000'GF    D6     0F1C  3103          INCL    G^PMS$GL_DIR_OUT
                                 0F22  3104          .ENDC
                                 0F22  3105
                 FODB'    30     0F22  3106          BSBW    CNX$SEND_MSG_CSB        ; Send the message
           50    2C A5    D0     0F25  3107          MOVL    CDRP$L_VAL1(R5),R0      ; Get address of RSB
           00000000'GF    16     0F29  3108          JSB     G^EXE$DEANONPAGED       ; Deallocate it
              50    55    D0     0F2F  3109          MOVL    R5,R0                   ; Get address of CDRP
           00000000'GF    17     0F32  3110          JMP     G^EXE$DEANONPAGED       ; Deallocate it
```

```
                     0F38  3111
                     0F38  3112 70$:     ; We are the directory system for this resource, yet RSB$M_DIRENTRY
                     0F38  3113          ; is not set.  See comment under RELOOKUP for a description of
                     0F38  3114          ; a race condition that is similar to what may have caused this.
                     0F38  3115
                     0F38  3116          BUG_CHECK       LOCKMGRERR,FATAL
                     0F3C  3117
                     0F3C  3118 80$:     ; No CDRP to allocate or CSID conversion error.  If insufficient
                     0F3C  3119          ; memory, then place the RSB on the timeout queue.  Since we
                     0F3C  3120          ; normally place LKBs on the timeout queue, we need to verify
                     0F3C  3121          ; that the fields we use in the RSB are not needed.  Henre,
                     0F3C  3122          ; the following ASSUME statements.
                     0F3C  3123
                     0F3C  3124          ASSUME  RSB$L_HSHCHN  EQ  LKB$L_ASTQFL
                     0F3C  3125          ASSUME  RSB$L_CVTQFL  EQ  LKB$L_DUETIME
                     0F3C  3126          ASSUME  RSB$Q_VALBLK+2  EQ  LKB$W_STATUS
                     0F3C  3127
          54   58   D0  0F3C  3128          MOVL    R8,R4                   ; Move RSB address
              FCAB   31  0F3F  3129          BRW     INSERT_ON_TIMOUTQ
                     0F42  3130
                     0F42  3131          ; Action routine to build the actual remove directory
                     0F42  3132          ; entry message.
                     0F42  3133          ; Inputs are:
                     0F42  3134          ;       R2      Address of message buffer
                     0F42  3135          ;       R5      Address of CDRP
                     0F42  3136          ; All registers except R0 and R1 must be preserved
                     0F42  3137
                     0F42  3138          ASSUME  LKMSG$B_RMOD  EQ  LKMSG$W_GROUP+2
                     0F42  3139          ASSUME  LKMSG$B_RSNLEN  EQ  LKMSG$B_RMOD+1
                     0F42  3140          ASSUME  LKMSG$T_RESNAM  EQ  LKMSG$B_RSNLEN+1
                     0F42  3141          ASSUME  RSB$B_RMOD  EQ  RSB$W_GROUP+2
                     0F42  3142          ASSUME  RSB$B_RSNLEN  EQ  RSB$B_RMOD+1
                     0F42  3143          ASSUME  RSB$T_RESNAM  EQ  RSB$B_RSNLEN+1
                     0F42  3144
                     0F42  3145 BLD_RMVDIRMSG:
               3C    BB    0F42  3146          PUSHR   #^M<R2,R3,R4,R5>
                     0F44  3147          ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
     08 A2   0402 8F    B0  0F44  3148          MOVW    #LKMSG$K_RMVDIR@8-      ; Store function and facility codes
                     0F4A  3149                  !CLSMSG$K_FAC_LCK,CLSMSG$B_FACILITY(R2)
               34 A5    B0  0F4A  3150          MOVW    CDRP$L_VAL3(R5),-       ; Store membership seq. num.
               0C A2        0F4D  3151                  LKMSG$Q_MEMSEQ(R2)
          51   2C A5    D0  0F4F  3152          MOVL    CDRP$L_VAL1(R5),R1      ; Get address of RSB
               44 A1    B0  0F53  3153          MOVW    RSB$W_HASHVAL(R1),-     ; Store hash value
               0E A2        0F56  3154                  LKMSG$W_HASHVAL(R2)
          50   4F A1    9A  0F58  3155          MOVZBL  RSB$B_RSNLEN(R1),R0     ; Get length of resource name
             50   04    C0  0F5C  3156          ADDL    #4,R0                   ; Account for group, access mode, etc.
          4C A1   50    28  0F5F  3157          MOVC3   R0,RSB$W_GROUP(R1),-    ; Move group no., resource name, etc.
               28 A2        0F63  3158                  LKMSG$W_GROUP(R2)       ; inot message buffer
               3C    BA  0F65  3159          POPR    #^M<R2,R3,R4,R5>
                     05  0F67  3160          RSB
```

```
                        0F68  3162          .SBTTL  LCK$RCV_RMVDIR - Receive remove directory entry message
                        0F68  3163
                        0F68  3164 ;++
                        0F68  3165 ; FUNCTIONAL DESCRIPTION:
                        0F68  3166 ;
                        0F68  3167 ;       This routine handles receiving a remove directory entry message.
                        0F68  3168 ;       This system must be the directory system for this resource.
                        0F68  3169 ;       The specified root resource is located in the resource hash table
                        0F68  3170 ;       and if all of its queues are empty, then the resource is deallocated.
                        0F68  3171 ;       If the wait queue has locks on it, then these must represent
                        0F68  3172 ;       requests from this system being sent to the system that formerly
                        0F68  3173 ;       managed this resource (the system that sent us this message).  In
                        0F68  3174 ;       this case the resource is not deallocated.  Instead, the RSB$L_CSID
                        0F68  3175 ;       field is cleared and we master the resource.  Clearing the field
                        0F68  3176 ;       must be done now instead of when our lock messages are returned
                        0F68  3177 ;       with the response "do locally".  Otherwise there is a race if the
                        0F68  3178 ;       remote system follows the RMVDIR message with a directory lookup
                        0F68  3179 ;       lock request.  We will then erroneously tell the other system to
                        0F68  3180 ;       do it locally while the other system is telling us the same thing.
                        0F68  3181 ;       Another race condition occurs if this message is delayed on the
                        0F68  3182 ;       sending side dur to insufficient pool.  In that case, it is
                        0F68  3183 ;       possible for us to receive a do locally lock response before
                        0F68  3184 ;       receiving this message.  That is why we tolerate the RSB$L_CSID
                        0F68  3185 ;       field being zero if we it doesn't match the remote system's CSID.
                        0F68  3186 ;
                        0F68  3187 ; CALLING SEQUENCE:
                        0F68  3188 ;
                        0F68  3189 ;       JSB     LCK$RCV_RMVDIR (called by the SCS received message routine)
                        0F68  3190 ;       IPL must be at IPL$_SYNCH
                        0F68  3191 ;
                        0F68  3192 ; INPUT PARAMETERS:
                        0F68  3193 ;
                        0F68  3194 ;       R2      Address of message buffer
                        0F68  3195 ;       R3      Address of CSB
                        0F68  3196 ;
                        0F68  3197 ; OUTPUT PARAMETERS:
                        0F68  3198 ;
                        0F68  3199 ;       None
                        0F68  3200 ;
                        0F68  3201 ; SIDE EFFECTS:
                        0F68  3202 ;
                        0F68  3203 ;       R0 - R5 destroyed
                        0F68  3204 ;--
                        0F68  3205
                        0F68  3206 LCK$RCV_RMVDIR::
          00000002      0F68  3207          .IF NE  CA$_MEASURE
00000000'GF   D6        0F68  3208          INCL    G^PMS$GL_DIR_IN
                        0F6E  3209          .ENDC
                        0F6E  3210
    0D0C 8F   BB        0F6E  3211          PUSHR   #^M<R2,R3,R8,R10,R11>
 54   24 A2   9E        0F72  3212          MOVAB   LKMSG$L_PARMSTLKID(R2),R4 ; Set start of resource description
          64   D4        0F76  3213          CLRL    (R4)                      ; Clear parent lock id
 5A   2B A2   9A        0F78  3214          MOVZBL  LKMSG$B_RSNLEN(R2),R10    ; Get length of resource name
    5A   08   C0        0F7C  3215          ADDL    #8,R10                    ; Account for extra fields to match
 51   0E A2   3C        0F7F  3216          MOVZWL  LKMSG$W_HASHVAL(R2),R1    ; Get hash value
00000000'GF   16        0F83  3217          JSB     G^LCK$SRCH_HSHTBL         ; Search hash table for resource
      47 50   E8        0F89  3218          BLBS    R0,80$                    ; Resource not found
```

```
                     0F8C  3219
                     0F8C  3220          ; Resource was been located, R5 points to RSB.  Verify it is a
                     0F8C  3221          ; directory entry mastered by other system.  Then verify all queues
                     0F8C  3222          ; are empty.  If granted or conversion queues are not empty then
                     0F8C  3223          ; we are in serious trouble.   It's okay if wait queue is not empty,
                     0F8C  3224          ; as this represents the race condition described above.
                     0F8C  3225
                     0F8C  3226          ASSUME  RSB$L_CVTQFL  EQ  RSB$L_GRQFL+8
                     0F8C  3227          ASSUME  RSB$L_WTQFL  EQ  RSB$L_CVTQFL+8
                     0F8C  3228          ASSUME  RSB$M_DIRENTRY  EQ  1
                     0F8C  3229
   58    55    D0  0F8C  3230          MOVL    R5,R8                   ; Move address of RSB
  40 0E  A8    E9  0F8F  3231          BLBC    RSB$W_STATUS(R8),80$    ; Verify directory entry bit is set
53    04  AE    D0  0F93  3232          MOVL    4(SP),R3                ; Get address of CSB
     4C  A3    D1  0F97  3233          CMPL    CSB$L_CSID(R3),-        ; Verify sending system is
     38  A8        0F9A  3234                  RSB$L_CSID(R8)          ; mastering resource
         07    13  0F9C  3235          BEQL    20$                     ; It is
     38  A8    D5  0F9E  3236          TSTL    RSB$L_CSID(R8)          ; It's not, but it's ok if this system
         29    13  0FA1  3237          BEQL    40$                     ; is mastering the resource
         2E    12  0FA3  3238          BNEQ    80$                     ; Error
50    10  A8    DE  0FA5  3239 20$:     MOVAL   RSB$L_GRQFL(R8),R0      ; Get address of granted queue
   50     60    D1  0FA9  3240          CMPL    (R0),R0                 ; Is granted queue empty?
         25    12  0FAC  3241          BNEQ    80$                     ; No
   50     08    C0  0FAE  3242          ADDL    #8,R0                   ; Yes, get address of conversion queue
   50     60    D1  0FB1  3243          CMPL    (R0),R0                 ; Is conversion queue empty?
         1D    12  0FB4  3244          BNEQ    80$                     ; No
   50     08    C0  0FB6  3245          ADDL    #8,R0                   ; Yes, get address of wait queue
   50     60    D1  0FB9  3246          CMPL    (R0),R0                 ; Is wait queue empty?
         05    13  0FBC  3247          BEQL    30$                     ; Yes
     38  A8    D4  0FBE  3248          CLRL    RSB$L_CSID(R8)          ; No, we master resource
         09    11  0FC1  3249          BRB     40$
                     0FC3  3250
                     0FC3  3251 30$:     ; All queues are empty and checks verified.  Deallocate RSB
                     0FC3  3252
         38  A8    D4  0FC3  3253          CLRL    RSB$L_CSID(R8)          ; Do this so that LCK$DEALLOC_RSB
                     0FC6  3254                                          ; really deallocates it
00000000'GF    16    0FC6  3255          JSB     G^LCK$DEALLOC_RSB       ; Deallocate it
                     0FCC  3256
    0D0C  8F    BA  0FCC  3257 40$:     POPR    #^M<R2,R3,R8,R10,R11>
      F02D'     31  0FD0  3258          BRW     CNX$DEALL_MSG_BUF_CSB   ; Deallocate message buffer and return
                     0FD3  3259
                     0FD3  3260
                     0FD3  3261 80$:     ; One of the following bugs:
                     0FD3  3262          ;       Resource not found (R5 = 0)
                     0FD3  3263          ;       Not marked as a directory entry (RSB$M_DIRENTRY not set)
                     0FD3  3264          ;       Not mastered by sending system (RSB$L_CSID not equal CSB$L_CSID
                     0FD3  3265          ;       and also not zero)
                     0FD3  3266          ;       Conversion or granted queue not empty (check queues)
                     0FD3  3267
                     0FD3  3268          BUG_CHECK       LOCKMGRERR,FATAL
```

```
                                3270          .SBTTL  VERIFYREMLKID - Verify remote lock id
                                3271
                                3272 ;++
                                3273 ; FUNCTIONAL DESCRIPTION:
                                3274 ;
                                3275 ;       This routine verifies a lock id sent by another system
                                3276 ;       and converts it into a LKB address.
                                3277 ;
                                3278 ; CALLING SEQUENCE:
                                3279 ;
                                3280 ;       BSBW    VERIFYREMLKID
                                3281 ;       IPL must be at IPL$_SYNCH
                                3282 ;
                                3283 ; INPUT PARAMETERS:
                                3284 ;
                                3285 ;       R0      Lock id on remote system
                                3286 ;       R1      Lock id on this system
                                3287 ;
                                3288 ; OUTPUT PARAMETERS:
                                3289 ;
                                3290 ;       R0      Completion code
                                3291 ;       R6      Address of LKB
                                3292 ;
                                3293 ; COMPLETION CODES:
                                3294 ;
                                3295 ;       SS$_NORMAL      Lock id was valid can converted to LKB address
                                3296 ;       SS$_IVLOCKID    Invalid lock id
                                3297 ;
                                3298 ; SIDE EFFECTS:
                                3299 ;
                                3300 ;       R4 is destroyed
                                3301 ;
                                3302 ; NOTE:
                                3303 ;
                                3304 ;       This routine does two consistency checks.  The first is that
                                3305 ;       it verifies the lock id is valid via the sequence number check.
                                3306 ;       If the lock id fails this check, then an error is returned to
                                3307 ;       the caller as this is allowed in some cases and is fatal in others.
                                3308 ;       However, if the lock id passes this check, then another check
                                3309 ;       is made that compares the remote lock id as sent by the remote
                                3310 ;       system with the remote lock id stored here in the LKB.  If this
                                3311 ;       check fails then it is immediately fatal as the first check should
                                3312 ;       catch all races that cause lock ids to not match across systems.
                                3313 ;       Also note that this second check is not perfect in that it should
                                3314 ;       also check CSB addresses.  This is considered unnecessary as the
                                3315 ;       additional protection that check offers is small.
                                3316 ;--
                                3317
                                3318 VERIFYREMLKID:
          56    51    3C        3319          MOVZWL  R1,R6                   ; Put lockid index in R6
00000000'GF     56    D1        3320          CMPL    R6,G^LCK$GL_MAXID       ; Is the lock id too big?
                1D    1A        3321          BGTRU   40$                     ; Yes
54    00000000'GF     D0        3322          MOVL    G^LCK$GL_IDTBL,R4       ; *** May combine with next instr.
        56    6446    D0        3323          MOVL    (R4)[R6],R6             ; Get LKB address
                10    18        3324          BGEQ    40$                     ; Unallocated id
        30 A6   51    D1        3325          CMPL    R1,LKB$L_LKID(R6)       ; Check sequence number
                0A    12        3326          BNEQ    40$                     ; Not valid
```

```
                54 A6    50    D1  0FF6  3327          CMPL    R0,LKB$L_REMLKID(R6)    ; Check remote lock id
                   0A    12    12  0FFA  3328          BNEQ    50$                     ; Doesn't match
                   50    00'   3C  0FFC  3329          MOVZWL  S^#SS$_NORMAL,R0        ; Success
                                 05  0FFF  3330          RSB
                                     1000  3331
                50   0000'8F    3C  1000  3332 40$:     MOVZWL  #SS$_IVLOCKID,R0        ; Invalid lock id
                                 05  1005  3333          RSB
                                     1006  3334
                                     1006  3335 50$:     BUG_CHECK       INVLOCKID,FATAL; Invalid remote lockid
                                     100A  3336
                                     100A  3337
                                     100A  3338
                                     100A  3339
                                     100A  3340          .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_CLOB              = 00000064
CSB$L_CSID              = 0000004C
CSID_ERROR                00000540 R      02
CVTQED_FORK               00000281 R      02
CVTQED_PROC               00000193 R      02
CVTRETRY_PROC             00000149 R      02
CVT_INVLKID               00000306 R      02
CVT_RETRY                 00000238 R      02
CVT_SCS_WAIT              000000F0 R      02
CVT_WAIT                  000000F3 R      02
DEACL_WARMCDRP            000006DD R      02
DIR_ENTER                 000008DF R      02
DIR_LOOKUP                000008AC R      02
DOLOCL_FORK               0000068C R      02
DOLOCL_PROC               00000568 R      02
DYN$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$DEACLOC_LKB           00000B65 RG     02
LCK$DEALLOC_RSB           ********    X   02
LCK$DEQLOCK               ********    X   02
LCK$DISPATCH              00000006 RG     02
LCK$EXTEND_IDTBL          ********    X   02
LCK$GB_RECD_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$M_CANCEL                    = 00000002
LCK$M_INVVALBLK                 = 00000004
LCK$NOT_QUEUED                    ********    X   02
LCK$QUEUECVT                      ********    X   02
LCK$QUEUED_EXIT                   ********    X   02
LCK$QUEUEWAIT                     ********    X   02
LCK$QUEUE_BLKAST                  ********    X   02
LCK$QUEUE_REM                     ********    X   02
LCK$RCV_BLKING                  00000D6D RG       02
LCK$RCV_CVTMSG                  0000030A RG       02
LCK$RCV_CVTREQ                  0000030C RG       02
LCK$RCV_DEQ                     00000E68 RG       02
LCK$RCV_DLCKFND                   ********    X   02
LCK$RCV_GRANTED                 00000C59 RG       02
LCK$RCV_LOCKREQ                 000007B6 RG       02
LCK$RCV_REDO_SRCH                 ********    X   02
LCK$RCV_RMVDIR                  00000F68 RG       02
LCK$RCV_SRCHDLCK                  ********    X   02
LCK$RCV_TIMESTAMP_RQST            ********    X   02
LCK$REBLD_LOCK                    ********    X   02
LCK$SND_BLKING                  00000D07 RG       02
LCK$SND_CVTREQ                  00000091 RG       02
LCK$SND_DEQCV                   00000DD9 RG       02
LCK$SND_DEQGR                   00000DE5 RG       02
LCK$SND_DEQWT                   00000DD2 RG       02
LCK$SND_GRANTED                 00000BAF RG       02
LCK$SND_LOCKREQ                 000004E2 RG       02
LCK$SND_RMVDIR                  00000EEA RG       02
LCK$SRCH_HSHTBL                   ********    X   02
LCK$SYNC_EXIT                     ********    X   02
LCK$V_CANCEL                    = 00000001
LCK$V_CVTSYS                    = 00000006
LCK$V_NOQUEUE                   = 00000002
LCK$V_PROTECT                   = 00000008
LCK$V_RECOVER                   = 00000007
LCK$V_VALBLK                    = 00000000
LCKRETRY_PROC                     00000562 R        02
LCK_RETRY                         00000680 R        02
LCK_SCS_WAIT                      00000505 R        02
LCK_SENDERR                       00000625 R        02
LCK_WAIT                          00000508 R        02
LKB$B_GRMODE                    = 00000035
LKB$B_RMOD                      = 0000000B
LKB$B_RQMODE                    = 00000034
LKB$B_STATE                     = 00000036
LKB$K_CONVERT                   = 00000000
LKB$K_GRANTED                   = 00000001
LKB$K_LENGTH                    = 00000060
LKB$K_RETRY                     = FFFFFFFE
LKB$K_RSPDOLOCL                 = FFFFFFF9
LKB$K_RSPGRANTD                 = FFFFFFFA
LKB$K_RSPNOTQED                 = FFFFFFFC
LKB$K_RSPQUEUED                 = FFFFFFFB
LKB$K_RSPRESEND                 = FFFFFFF8
LKB$K_SCSWAIT                   = FFFFFFFD
LKB$K_WAITING                   = FFFFFFFF
LKB$L_AST                       = 00000010
LKB$L_ASTQFL                    = 00000000
LKB$L_BLKASTADR                 = 00000020
LKB$L_CSID                      = 00000058
LKB$L_DLCKPRI                   = 00000024
LKB$L_DUETIME                   = 00000018
LKB$L_EPID                      = 00000014
LKB$L_LKID                      = 00000030
LKB$L_LKST1                     = 0000002C
LKB$L_OLDBLKAST                 = 0C00005C
LKB$L_PARENT                    = 00000048
LKB$L_PID                       = 0000000C
LKB$L_REMLKID                   = 00000054
LKB$L_RSB                       = 00000050
LKB$L_SQFL                      = 00000038
LKB$M_ASYNC                     = 00000004
LKB$M_BLKASTQED                 = 00000008
LKB$M_CVTTOSYS                  = 00000100
LKB$M_DBLKAST                   = 00000002
LKB$M_DCPLAST                   = 00000001
LKB$M_MSTCPY                    = 00000010
LKB$M_PKAST                     = 00000010
LKB$M_PROTECT                   = 00000200
LKB$M_TIMOUTQ                   = 00000040
LKB$M_WASSYSOWN                 = 00000080
LKB$V_ASYNC                     = 00000002
LKB$V_BLKASTQED                 = 00000003
LKB$V_DBLKAST                   = 00000001
LKB$V_MSTCPY                    = 00000004
LKB$V_PROTECT                   = 00000009
LKB$V_TIMOUTQ                   = 00000006
LKB$W_FLAGS                     = 00000028
LKB$W_REFCNT                    = 0000004C
LKB$W_RQSEQNM                   = 00000010
LKB$W_SIZE                      = 00000008
LKB$W_STATUS                    = 0000002A
LKMSG$B_GRMODE                  = 0000001B
LKMSG$B_LCKSTATE                = 0000006A
LKMSG$B_LSTATUS                 = 00000068
LKMSG$B_RMOD                    = 0000002A
LKMSG$B_RQMODE                  = 0000001A
LKMSG$B_RSNLEN                  = 0000002B
LKMSG$B_RSTATUS                 = 00000069
LKMSG$B_STATE                   = 0000001E
LKMSG$K_BLKING                  = 00000005
LKMSG$K_CVTLCKM                 = 00000006
LKMSG$K_CVTLCKR                 = 00000007
LKMSG$K_DEQ                     = 00000003
LKMSG$K_DLCKFND                 = 0000000B
LKMSG$K_GRANTED                 = 00000002
LKMSG$K_NEWLOCK                 = 00000001
LKMSG$K_REBLDLOCK               = 00000008
LKMSG$K_REDO_SRCH               = 0000000C
LKMSG$K_RMVDIR                  = 00000004
LKMSG$K_SRCHDLCK                = 0000000A
LKMSG$K_TSRQST                  = 00000009
LKMSG$L_BLKASTFLG               = 0000001C
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$W_STATUS                  =  0000001C
LOCK_GRANTED                       000009DF R      02
LOCK_QUEUED                        000009E7 R      02
NEW_RESOURCE                       00000950 R      02
NOTQED_FORK                        00000697 R      02
NOTQED_PROC                        000005B1 R      02
NOTQUEQED_EXIT                     00000922 R      02
NO_LOCKIDS                         00000AC4 R      02
PCB$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$L_PARFNT                    =  00000048
RSB$L_VA_EQNUM                  =  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
_255DUA28:[SYSLOA.OBJ]CLUSTER.MLB;1	1
_255DUA28:[SYS.OBJ]LIB.MLB;1	16
_255DUA28:[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
```

CSPOPCOM.LIS

CSPWAIT.LIS

CSPRCPCAC.LIS

CSPCJFRES.LIS

CSPQUORUM.LIS

DISTRLKI.LIS

CSPMOUNT.LIS

CSPVECTOR.LIS

CSPCLIENT.LIS

DSTRLOCK.LIS

DSTRLOCK.LIS

ERRSUBUV1
LIS

ERRSUB790
LIS

ERRSUB780
LIS

ERRSUB750
LIS

ERRSUB730
LIS

FALMAN
LIS