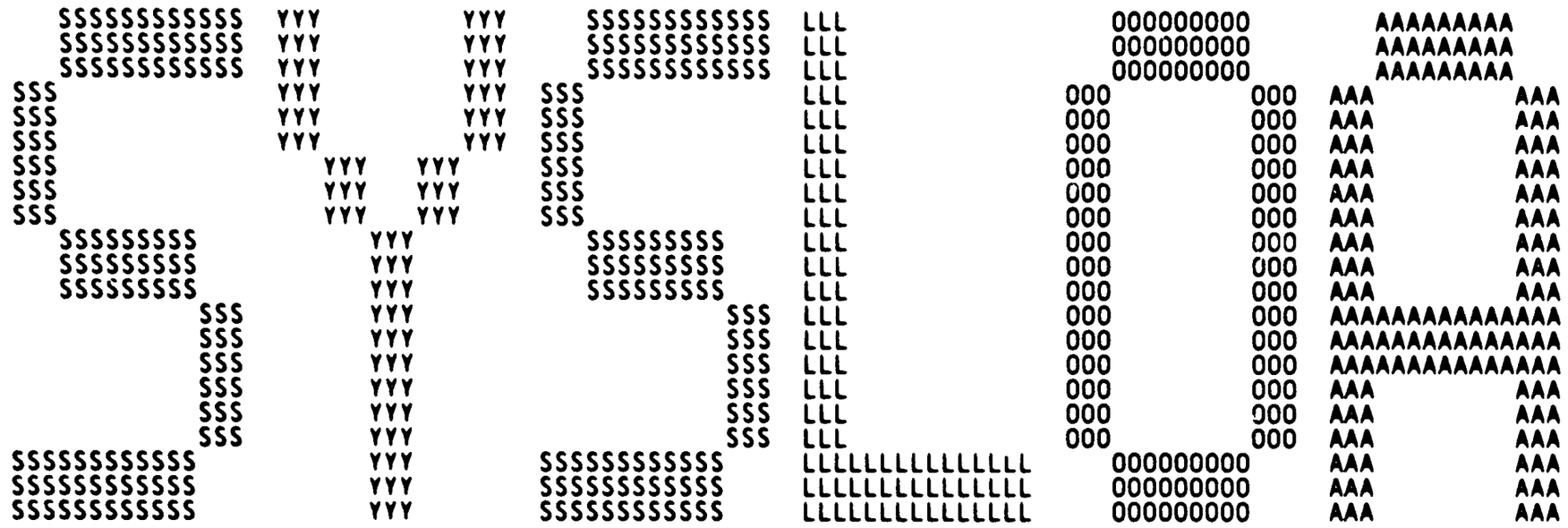
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 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

```
DDDDDDDD      IIIIII      SSSSSSSS  TTTTTTTTTT  RRRRRRRR    LL          KK      KK    IIIIII
DDDDDDDD      IIIIII      SSSSSSSS  TTTTTTTTTT  RRRRRRRR    LL          KK      KK    IIIIII
DD      DD      II      SS            TT      RR      RR  LL          KK      KK      II
DD      DD      II      SS            TT      RR      RR  LL          KK      KK      II
DD      DD      II      SS            TT      RR      RR  LL          KK    KK        II
DD      DD      II      SS            TT      RR      RR  LL          KK    KK        II
DD      DD      II        SSSSSS      TT      RRRRRRRR    LL          KKKKKK          II
DD      DD      II        SSSSSS      TT      RRRRRRRR    LL          KKKKKK          II
DD      DD      II              SS    TT      RR  RR      LL          KK    KK        II
DD      DD      II              SS    TT      RR  RR      LL          KK    KK        II
DD      DD      II              SS    TT      RR    RR    LL          KK      KK      II        ....
DD      DD      II              SS    TT      RR    RR    LL          KK      KK      II        ....
DDDDDDDD      IIIIII    SSSSSSSS      TT      RR      RR  LLLLLLLLLL  KK      KK    IIIIII      ....
DDDDDDDD      IIIIII    SSSSSSSS      TT      RR      RR  LLLLLLLLLL  KK      KK    IIIIII      ....

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

(2)	74	DECLARATIONS
(3)	120	LKI$DISPATCH - Dispatch incoming GETLKI message
(4)	164	LKI$SND_STDREQ - Send a standard information request message
(5)	252	SENDSTDREQ - Send standard information request to remote system
(7)	383	LKI$RCV_STDINFO - Receive standard information request
(8)	519	LKI$SND_BLKING - Send a request for list of blocking locks
(8)	520	LKI$SND_BLKBY - Send a request for list of blockedby locks
(8)	521	LKI$SND_LOCKS - Send a request for list of all locks
(9)	616	SENDBLKINGREQ - Send request for blocking locks to remote system
(9)	617	SENDBLKBYREQ - Send request for blocked locks to remote system
(9)	618	SENDLOCKSREQ - Send request for all locks to remote system
(11)	766	LKI$RCV_BLKING - Receive a request for list of blocking locks
(11)	767	LKI$RCV_BLKBY - Receive a request for list of blocked locks
(11)	768	LKI$RCV_LOCKS - Receive a request for list of all locks
(15)	1084	VERIFYREMLKID - Verify remote lock id

```
0000     1          .TITLE  DISTRLKI - Distributed GETLKI Loadable Code
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27  ;
0000    28
0000    29  ;++
0000    30  ;
0000    31  ; FACILITY:     Executive, system services and fork level code
0000    32  ;
0000    33  ; ABSTRACT:
0000    34  ;       This module contains routines used to implement distributed
0000    35  ;       GETLKI system service functions.
0000    36  ;
0000    37  ; ENVIRONMENT:  Kernel mode, fork level, loadable code
0000    38  ;
0000    39  ;--
0000    40  ;
0000    41  ; AUTHOR: Rod Gamache,           CREATION DATE: 3-Jun-1983
0000    42  ;
0000    43  ; MODIFIED BY:
0000    44  ;
0000    45  ;       V03-007 RNG0007         Rod N. Gamache          22-Aug-1984
0000    46  ;               Fix race condition in check for spurious wake-ups, by
0000    47  ;               raising IPL.
0000    48  ;
0000    49  ;       V03-006 RNG0006         Rod N. Gamache          3-Aug-1984
0000    50  ;               Map all Lock waiting states to LKI$C_WAITING code.
0000    51  ;
0000    52  ;       V03-005 RNG0005         Rod N. Gamache          13-Apr-1984
0000    53  ;               Fix bug where process was waiting and pre-mature wakes
0000    54  ;               caused the process to continue.
0000    55  ;
0000    56  ;       V03-004 RNG0004         Rod N. Gamache          26-Mar-1984
0000    57  ;               Return the LOCKID as the REMLKID on Local copy locks.
```

```
0000     58 ;
0000     59 ;       V03-003 RNG0003         Rod N. Gamache          21-Mar-1984
0000     60 ;               Add REMLKID and remCSID to returns for list items.
0000     61 ;               Save/restore R2 on call to list of locks routine.
0000     62 ;
0000     63 ;       V03-002 RNG0002         Rod N. Gamache          05-Dec-1983
0000     64 ;               Change reference to RSB reference count from longword to
0000     65 ;               word - this reflects changes made to the Lock Manager.
0000     66 ;
0000     67 ;       V03-001 RNG0001         Rod N. Gamache          05-Aug-1983
0000     68 ;               Add support for distributed list items (LOCKS, BLOCKEDBY
0000     69 ;               and BLOCKING).
0000     70 ;               Fix problem with standard information processing, must
0000     71 ;               not hold onto the CDRP message buffer for return data.
0000     72 ;
```

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

                ;
                ; MACROS:
                ;

                        $CDRPDEF                        ; CDRP offsets
                        $CLSMSGDEF                      ; Cluster message offsets
                        $CLUBDEF                        ; Cluster block definitions
                        $CSBDEF                         ; CSB offsets
                        $DYNDEF                         ; Data structure names
                        $FKBDEF                         ; Fork block offsets
                        $IPLDEF                         ; IPL definitions
                        $LCKDEF                         ; LCK definitions
                        $LKBDEF                         ; LKB offsets
                        $LKIDEF                         ; LKI item codes
                        $PCBDEF                         ; PCB offsets
                        $PDTDEF                         ; PDT offsets
                        $PRIDEF                         ; Priority definitions
                        $PSLDEF                         ; PSL definitions
                        $RSBDEF                         ; RSB offsets
                        $RSNDEF                         ; Resource numbers
                        $VADEF                          ; Define virtual addresses

                ;
                ; EQUATED SYMBOLS:
                ;

                ;
                ; OWN STORAGE:
                ;

                        .PSECT  $$$020

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

                        ASSUME  IPL$_SYNCH  EQ  IPL$_SCS
```

```
0000    120          .SBTTL  LKI$DISPATCH - Dispatch incoming GETLKI message
0000    121
0000    122 ;++
0000    123 ; FUNCTIONAL DESCRIPTION:
0000    124 ;
0000    125 ;       This routine dispatches incoming GETLKI messages to
0000    126 ;       the appropriate routine.
0000    127 ;
0000    128 ; CALLING SEQUENCE:
0000    129 ;
0000    130 ;       JSB     LKI$DISPATCH    (called from connection manager received
0000    131 ;                               message routine)
0000    132 ;       IPL is at IPL$_SYNCH
0000    133 ;
0000    134 ; INPUT PARAMETERS:
0000    135 ;
0000    136 ;       R2      Address of message
0000    137 ;       R3      CSID
0000    138 ;       R4      Address of PDT
0000    139 ;       R5      Address of CDRP (if this message needs a response)
0000    140 ;
0000    141 ; OUTPUT PARAMETERS:
0000    142 ;
0000    143 ;       None
0000    144 ;
0000    145 ; SIDE EFFECTS:
0000    146 ;
0000    147 ;       R0 - R5 destroyed
0000    148 ;--
0000    149
0000    150 LKI$DISPATCH::
0000    151
0000    152          DISPATCH        CLSMSG$B_FUNC(R2),TYPE=B,PREFIX=LIMSG$K_,-
0000    153                  <-
0000    154                  <STDINFO,LKI$RCV_STDINFO>,-     ; Standard lock information
0000    155                  <BLKING,LKI$RCV_BLKING>,-       ; List of locks blocking this lock
0000    156                  <BLKBY,LKI$RCV_BLKBY>,-         ; List of locks blocked by this lock
0000    157                  <LOCKS,LKI$RCV_LOCKS>,-         ; List of locks on resource
0000    158                  >
000D    159
000D    160          ; Unrecognized function code
000D    161
000D    162          BUG_CHECK       LOCKMGRERR,FATAL
```

```
                                0011   164          .SBTTL  LKI$SND_STDREQ - Send a standard information request message
                                0011   165
                                0011   166 ;++
                                0011   167 ; FUNCTIONAL DESCRIPTION:
                                0011   168 ;
                                0011   169 ;       This routine handles getting standard information requests that must
                                0011   170 ;       be forwarded to the master system for this lock.  This system is the
                                0011   171 ;       process system.
                                0011   172 ;
                                0011   173 ; CALLING SEQUENCE:
                                0011   174 ;
                                0011   175 ;       BSB/JSB         LKI$SND_STDREQ
                                0011   176 ;
                                0011   177 ;       IPL must be at IPL$_SYNCH
                                0011   178 ;
                                0011   179 ; INPUT PARAMETERS:
                                0011   180 ;
                                0011   181 ;       R3      CSID of destination system
                                0011   182 ;       R4      Address of PCB
                                0011   183 ;       R8      Address of RSB
                                0011   184 ;       R9      Address of LKB
                                0011   185 ;       R11     Scratch
                                0011   186 ;
                                0011   187 ; OUTPUT PARAMETERS:
                                0011   188 ;
                                0011   189 ;       R0      Completion code
                                0011   190 ;       R11     Address of REMOTE LKI INFORMATION BLOCK or zero
                                0011   191 ;
                                0011   192 ; SIDE EFFECTS:
                                0011   193 ;
                                0011   194 ;       The process will go into MWAIT until the response from the remote
                                0011   195 ;       system arrives.
                                0011   196 ;
                                0011   197 ;       IPL = ASTDEL
                                0011   198 ;
                                0011   199 ;       R1 - R5 destroyed
                                0011   200 ;--
                                0011   201
                                0011   202 LKI$SND_STDREQ::
                                0011   203
                                0011   204          ; Send a message and wait for response.
                                0011   205
            5B    D4            0011   206          CLRL    R11                             ; Zero remote lock block address
51    0040 8F    3C            0013   207          MOVZWL  #LIMSG$K_STDINFO_LEN,R1 ; Get size of return info
 00000000'GF    16            0018   208          JSB     G^EXE$ALONONPAGED       ; Allocate a buffer
       49 50    E9            001E   209          BLBC    R0,110$                 ; Br on error
                                0021   210          ASSUME  FKB$B_TYPE EQ FKB$W_SIZE+2
 00130040 8F    D0            0021   211          MOVL    #<DYN$C_BUFIO@16>!LIMSG$K_STDINFO_LEN,-
       08 A2                    0027   212                  FKB$W_SIZE(R2)          ; Store size and type
       5B   52  D0            0029   213          MOVL    R2,R1T                  ; Copy buffer address
         FFD1' 30            002C   214          BSBW    CNX$ALLOC_WARMCDRP      ; Alloc. a CDRP with RSPID and cvt CSID
       37 50    E9            002F   215          BLBC    R0,90$                  ; No CDRPs or CSID error
  38 A5   5B    D0            0032   216          MOVL    R11,CDRP$L_VAL4(R5)     ; Save address of system buffer in CDRP
  6B    00'8F    9A            0036   217          MOVZBL  #SS$_NORMAL,(R11)       ; Assume success, zero completion word
       60 A4    D0            003A   218          MOVL    PCB$L_PID(R4),-         ; Save PID in CDRP
       30 A5                    003D   219                  CDRP$L_VAL2(R5)
  2C A5   59    D0            003F   220          MOVL    R9,CDRP$L_VAL1(R5)      ; Copy address of LKB
```

```
                002A   30  0043   221          BSBW    SENDSTDREQ              ; Send standard request
                       0046   222
            19 02 AB   E8  0046   223 30$:     BLBS    2(R11),50$              ; Continue if transaction completed
                       004A   224
                       004A   225          ; Put the process into MWAIT until the response arrives.
                       004A   226
             50   0D   D0  004A   227          MOVL    #RSN$_SCS,R0            ; Go into MWAIT for resource RSN$_SCS
                  10   78  004D   228          ASHL    #PSL$V_IPL,-            ; Create a PSL on stack with IPL
             7E   02        004F   229                  #IPL$_ASTDEL,-(SP)      ; set to ASTDEL
     54 00000000'GF   D0  0051   230          MOVL    G^SCH$GL_CURPCB,R4      ; Get our PCB address
        00000000'GF   16  0058   231          JSB     G^SCH$RWAIT             ; Wait
                       005E   232
                       005E   233          ; Upon reawakening, IPL = ASTDEL; we must raise to IPL = SYNCH
                       005E   234
                       005E   235          SETIPL  #IPL$_SYNCH             ; Raise IPL - to lock out completions
                E3     11  0061   236          BRB     30$                     ; Make sure we have completed
                       0063   237
                       0063   238 50$:     ; Lower IPL to IPL = ASTDEL
                       0063   239          ;
                       0063   240          ; Inputs:
                       0063   241          ;       R11     Address of return buffer
                       0063   242
                       0063   243          SETIPL  #IPL$_ASTDEL            ; Lower IPL, in case we raised it
             50   6B   3C  0066   244          MOVZWL  (R11),R0                ; Get status
                       0069   245
                       05  0069   246 90$:     RSB                             ; Return to caller
                       006A   247
     50 0000'8F   3C  006A   248 110$:    MOVZWL  #SS$_INSFMEM,R0         ; Return error
                       05  006F   249          RSB                             ; Return to caller
                       0070   250
```

```
                        0070   252          .SBTTL  SENDSTDREQ - Send standard information request to remote system
                        0070   253
                        0070   254 ;++
                        0070   255 ; FUNCTIONAL DESCRIPTION:
                        0070   256 ;
                        0070   257 ;       This routine takes care of building and sending the standard lock
                        0070   258 ;       information request.  It also handles the fork level processing of
                        0070   259 ;       the response.  We are the process system; the remote system is the
                        0070   260 ;       master system.  SENDSTDREQ is called to send a standard lock inform-
                        0070   261 ;       ation request that requires a  response.
                        0070   262 ;
                        0070   263 ; CALLING SEQUENCE:
                        0070   264 ;
                        0070   265 ;       BSBW    SENDSTDREQ
                        0070   266 ;
                        0070   267 ;       IPL must be at SCS fork IPL (8)
                        0070   268 ;
                        0070   269 ;       This routine operates as a fork process so it may return
                        0070   270 ;       to its caller before completing.
                        0070   271 ;
                        0070   272 ; INPUT PARAMETERS:
                        0070   273 ;
                        0070   274 ;       R3      CSB address of destination system
                        0070   275 ;       R5      CDRP address
                        0070   276 ;
                        0070   277 ; OUTPUT PARAMETERS:
                        0070   278 ;
                        0070   279 ;       NONE
                        0070   280 ;
                        0070   281 ; SIDE EFFECTS:
                        0070   282 ;
                        0070   283 ;       NONE
                        0070   284 ;
                        0070   285 ;--
                        0070   286          .ENABL  LSB
                        0070   287
                        0070   288 SENDSTDREQ:
    00CF'CF    9E       0070   289          MOVAB   W^BLD_STDMSGR,-         ; Store address of message build
      4C A5              0074   290                  CDRP$L_MSGBLD(R5)       ; routine
       FF87'   30        0076   291          BSBW    CNX$SEND_MSG_CSB        ; Send the message
51    38 A5    D0        0079   292          MOVL    CDRP$L_VAL4(R5),R1      ; Get address of return buffer
      02 A1    B6        007D   293          INCW    2(R1)                   ; Indicate transaction complete
      3A 50    E9        0080   294          BLBC    R0,100$                 ; Dest. system is no longer in cluster
                        0083   295
                        0083   296          ; We are resumed here when the response message arrives.
                        0083   297          ; Registers contain:
                        0083   298          ;       R2      Address of message buffer
                        0083   299          ;       R3      CSB address
                        0083   300          ;       R4      Address of PDT
                        0083   301          ;       R5      Address of CDRP
                        0083   302
                        0083   303          ; Request for information was successfull, wake process
                        0083   304
                        0083   305          DISPATCH        LIMSG$B_STATUS(R2),TYPE=B,PREFIX=LIMSG$K_,-
                        0083   306                  <-
                        0083   307                  <RSPSUCCESS,70$>,-      ; Success
                        0083   308                  <RSPIVLKID,50$>,-       ; Invalid lockid
```

```
                                0083    309                     >
                                008C    310
                                008C    311             BUG_CHECK       LOCKMGRERR,FATAL
                                0090    312
                                0090    313
                                0090    314             ; Invalid lock id
                                0090    315
          61    0000'8F   B0    0090    316 50$:        MOVW    #SS$_IVLOCKID,(R1)      ; Return error
                      0A    11  0095    317             BRB     DEALL_WARMCDRP          ; Deallocate resources
                                0097    318
                                0097    319             ; Success
                                0097    320
                          2C    BB  0097    321 70$:        PUSHR   #^M<R2,R3,R5>           ; Save registers
                                0099    322             ASSUME  LIMSG$L_MSTLKID GE FKB$W_SIZE+4
                          28    28  0099    323             MOVC3   #LIMSG$R_STDINFO_LEN-LIMSG$L_MSTLKID,-
                18 A1     18 A2       009B    324                     LIMSG$L_MSTLKID(R2),LIMSG$L_MSTLKID(R1) ; Copy the data
                          2C    BA  009F    325             POPR    #^M<R2,R3,R5>           ; Restore registers
                                00A1    326
                                00A1    327 DEALL_WARMCDRP:
                                00A1    328             ; Deallocate CDRP (R5), message buffer (R2), and RSPID (in CDRP).
                                00A1    329             ; CSB address is in R3.
                                00A1    330
            54    30 A5     3C  00A1    331             MOVZWL  CDRP$L_VAL2(R5),R4      ; Get process index
                   FF58'    30  00A5    332             BSBW    CNX$DEALL_WARMCDRP_CSB  ; Deallocate the package
                                00A8    333
                                00A8    334
                                00A8    335 WAKE_PROCESS:
                                00A8    336             ; R4 contains process index
                                00A8    337
          50    00000000'GF   D0  00A8    338             MOVL    G^SCH$GL_PCBVEC,R0      ; Get address of PCB vector
                  54    6044  D0  00AF    339             MOVL    (R0)[R4],R4             ; Get address of PCB
                                00B3    340
                                00B3    341             ; Change process state to executable
                                00B3    342
                          52    D4  00B3    343             CLRL    R2                      ; No priority increment
                                00B5    344             RPTEVT  AST,JSB                 ; Report event (use JSB G^)
                          05    00BC    345             RSB
                                00BD    346
                                00BD    347             ; Destination system left the cluster
                                00BD    348
            61    50    B0    00BD    349 100$:       MOVW    R0,(R1)                 ; Return error
            50    55    D0    00C0    350             MOVL    R5,R0                   ; Get address of CDRP
      54    30 A5     3C    00C3    351             MOVZWL  CDRP$L_VAL2(R5),R4      ; Get process index
00000000'GF       16    00C7    352             JSB     G^EXE$DEANONPAGED       ; Deallocate the CDRP
                  D9    11  00CD    353             BRB     WAKE_PROCESS            ; And wake process
                                00CF    354
                                00CF    355             .DSABL  LSB
                                00CF    356
```

```
                        00CF    358 ;****************************************************************************
                        00CF    359 ;
                        00CF    360 ;               Build standard information request message action routine
                        00CF    361 ;
                        00CF    362 ;****************************************************************************
                        00CF    363
                        00CF    364         ; Action routine to build the actual request message
                        00CF    365         ; Inputs are:
                        00CF    366         ;       R2      Address of message buffer
                        00CF    367         ;       R5      Address of CDRP
                        00CF    368         ; All registers except R0 and R1 must be preserved
                        00CF    369
                        00CF    370 BLD_STDMSGR:
                        00CF    371
                        00CF    372         ASSUME  LIMSG$L_PRCLKID  EQ  LIMSG$L_MSTLKID+4
                        00CF    373         ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
                        00CF    374
08 A2    0105 8F    B0  00CF    375         MOVW    #LIMSG$K_STDINFO@8-     ; Store function and facility codes
                        00D5    376                 !CLSMSG$K_FAC_LKI,CLSMSG$B_FACILITY(R2)
    50     2C A5    D0  00D5    377         MOVL    CDRP$L_VAL1(R5),R0      ; Get LKB address
 51    52     18    C1  00D9    378         ADDL3   #LIMSG$L_MSTLKID,R2,R1  ; Point into message buffer
   81     54 A0    D0  00DD    379         MOVL    LKB$L_REMLKID(R0),(R1)+ ; Store master lockid
   81     30 A0    D0  00E1    380         MOVL    LKB$L_LKID(R0),(R1)+    ; Store process lockid
                    05  00E5    381         RSB
```

```
                     00E6   383          .SBTTL  LKI$RCV_STDINFO - Receive standard information request
                     00E6   384
                     00E6   385 ;++
                     00E6   386 ; FUNCTIONAL DESCRIPTION:
                     00E6   387 ;
                     00E6   388 ;       This routine receives conversion requests from the remote system.
                     00E6   389 ;       The remote system is the process system and we are the master
                     00E6   390 ;       system for this lock.  The conversion request is performed and
                     00E6   391 ;       a response message is returned if called at entry point
                     00E6   392 ;       LKI$RCV_STDREQ.
                     00E6   393 ;
                     00E6   394 ; CALLING SEQUENCE:
                     00E6   395 ;
                     00E6   396 ;       JSB     LKI$RCV_STDREQ (called from CNX received message routine)
                     00E6   397 ;
                     00E6   398 ;       IPL must be at IPL$_SYNCH
                     00E6   399 ;
                     00E6   400 ; INPUT PARAMETERS:
                     00E6   401 ;
                     00E6   402 ;       R2      Address of message buffer
                     00E6   403 ;       R3      Address of CSB
                     00E6   404 ;       R5      Address of CDRP
                     00E6   405 ;
                     00E6   406 ; OUTPUT PARAMETERS:
                     00E6   407 ;
                     00E6   408 ;       None
                     00E6   409 ;
                     00E6   410 ; SIDE EFFECTS:
                     00E6   411 ;
                     00E6   412 ;       A message is sent back as a response
                     00E6   413 ;
                     00E6   414 ;       R0 - R5 destroyed
                     00E6   415 ;--
                     00E6   416
                     00E6   417          ASSUME  LIMSG$L_PRCLKID  EQ  LIMSG$L_MSTLKID+4
                     00E6   418
                     00E6   419 LKI$RCV_STDINFO::
          01CC 8F    BB   00E6   420          PUSHR   #^M<R2,R3,R6,R7,R8>
               52    DD   00EA   421          PUSHL   R2                      ; Save message buffer address
             FF11'   30   00EC   422          BSBW    CNX$INIT_CDRP           ; Initialize CDRP
               52 8ED0    00EF   423          POPL    R2                      ; Restore message buffer address
57    52     18    C1   00F2   424          ADDL3   #LIMSG$L_MSTLKID,R2,R7  ; Point R7 into message buffer
      51     87    D0   00F6   425          MOVL    (R7)+,R1                ; Get master lockid (id on this system)
      50     87    D0   00F9   426          MOVL    (R7)+,R0                ; Get process lockid (for verify)
             02    90   00FC   427          MOVB    #LIMSG$K_RSPIVLKID,-    ; Assume error on lock id
          3C A5          00FE   428                  CDRP$L_VAL5(R5)
           0337   30   0100   429          BSBW    VERIFYREMLKID           ; Convert to LKB address
          08 50   E9   0103   430          BLBC    R0,STD_IVLKID           ; Invalid lock id
                       0106   431
                       0106   432          ; Have LKB address in R6, pointer into message in R7.  Set
                       0106   433          ; up other registers as needed.
                       0106   434
  58    50 A6    D0   0106   435          MOVL    LKB$L_RSB(R6),R8        ; Get RSB address
                       010A   436
                       010A   437          ; Send response message.  CDRP address is in R5.
                       010A   438
             01    90   010A   439          MOVB    #LIMSG$K_RSPSUCCESS,-   ; Store status indicator
```

```
                3C A5        010C   440                     CDRP$L_VAL5(R5)
                             010E   441 STD_IVLKID:
              2C A5    56    D0  010E   442             MOVL    R6,CDRP$L_VAL1(R5)      ; Store LKB address
                 01CC 8F    BA  0112   443             POPR    #^M<R2,R3,R6,R7,R8>
                    36'AF   9E  0116   444             MOVAB   B^BLD_STDRSP,-          ; Store address of message build
                       4C A5    0119   445                     CDRP$L_MSGBLD(R5)       ; routine
              53       4C A3    D0  011B   446             MOVL    CSB$L_CSID(R3),R3       ; Get CSID out of CSB
                        FEDE'   30  011F   447             BSBW    CNX$SEND_MSG_RESP       ; Send message response
                                    0122   448
                                    0122   449             ; Return here only on errors
                                    0122   450
          0000'8F    50    B1  0122   451             CMPW    R0,#SS$_NOSUCHNODE      ; Is status SS$_NOSUCHNODE?
                        09   13  0127   452             BEQL    10$                     ; Yes, error
                50      55    D0  0129   453             MOVL    R5,R0                   ; No, SS$_NODELEAVE is okay
           00000000'GF    17  012C   454             JMP     G^EXE$DEANONPAGED       ; Deallocate CDRP
                                    0132   455
                                    0132   456 10$:    BUG_CHECK       LOCKMGRERR,FATAL; CSID invalid
                                    0136   457
                                    0136   458 BLD_STDRSP:
                                    0136   459             ; This routine builds a response message for conversion requests
                                    0136   460             ; Inputs:
                                    0136   461             ;       R2      Address of message buffer
                                    0136   462             ;       R5      Address of CDRP
                                    0136   463             ; All registers except R0 and R1 must be preserved
                                    0136   464
                                    0136   465             ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
                                    0136   466
        08 A2      0185 8F    B0  0136   467             MOVW    #LIMSG$K_STDINFO@8-     ; Store function and facility codes
                                    013C   468                     !CLSMSG$M_RESPMSG-
                                    013C   469                     !CLSMSG$K_FAC_LKI,CLSMSG$B_FACILITY(R2)
                3C A5    90  013C   470             MOVB    CDRP$L_VAL5(R5),-       ; Store response status
                20 A2       013F   471                     LIMSG$B_STATUS(R2)
                                    0141   472             DISPATCH        LIMSG$B_STATUS(R2),TYPE=B,PREFIX=LIMSG$K_,-
                                    0141   473                     <-
                                    0141   474                     <RSPSUCCESS,5$>,-       ; Success
                                    0141   475                     <RSPIVLKID,40$>,-       ; Invalid lockid
                                    0141   476                     >
                                    014A   477             BUG_CHECK       LOCKMGRERR,FATAL
                                    014E   478
    50  2C A5    D0  014E   479 5$:      MOVL    CDRP$L_VAL1(R5),R0      ; Get address of LKB
        30 A0    D0  0152   480         MOVL    LKB$L_LKID(R0),-        ; Store master lockid
        18 A2        0155   481                 LIMSG$L_MSTLKID(R2)
        54 A0    D0  0157   482         MOVL    LKB$L_REMLKID(R0),-     ; Store process lockid
        1C A2        015A   483                 LIMSG$L_PRCLKID(R2)
                             015C   484
                             015C   485         ASSUME  LKB$B_GRMODE EQ LKB$B_RQMODE+1
                             015C   486         ASSUME  LKB$B_STATE EQ LKB$B_GRMODE+1
                             015C   487
        35 A0    D0  015C   488         MOVL    LKB$B_GRMODE(R0),-      ; Get state information
        24 A2        015F   489                 LIMSG$L_STATE(R2)
        36 A0    95  0161   490         TSTB    LKB$B_STATE(R0)         ; Is state information okay?
           05    18  0164   491         BGEQ    7$                      ; Br if yes, continue
        FF 8F    90  0166   492         MOVB    #LKI$C_WAITING,-        ; Else, map all waiting states
        26 A2        0169   493                 LIMSG$L_STATE+2(R2)     ;   to same code
                             016B   494 7$:
    51  50 A0    D0  016B   495         MOVL    LKB$L_RSB(R0),R1        ; Get RSB address
        28 A1    7D  016F   496         MOVQ    RSB$Q_VALBLK(R1),-      ; Get VALBLK information
```

```
                30 A2        0172   497                 LIMSG$Q_VALBLK(R2)
                30 A1   7D   0174   498         MOVQ    RSB$Q_VALBLK+8(R1),-    ; Get rest of VALBLK information
                38 A2        0177   499                 LIMSG$Q_VALBLK+8(R2)
                40 A1   3C   0179   500         MOVZWL  RSB$W_REFCNT(R1),-      ; Get sub-RSB reference count
                28 A2        017C   501                 LIMSG$L_RSBREFCNT(R2)
                             017E   502
                             017E   503         ; Calculate the LOCK COUNT on the RSB
                             017E   504
                50   D4      017E   505         CLRL    R0                      ; Init count
                18   BB      0180   506         PUSHR   #^M<R3,R4>              ; Save registers
          53    10 A1   9E   0182   507         MOVAB   RSB$L_GRQFL(R1),R3      ; Get address of granted queue listhead
          54    53      D0   0186   508         MOVL    R3,R4                   ; Save listhead address
          54    63      D1   0189   509 10$:    CMPL    (R3),R4                 ; End of list?
                07      13   018C   510         BEQL    20$                     ; Br if yes
                50      D6   018E   511         INCL    R0                      ; Increment count
          53    63      D0   0190   512         MOVL    (R3),R3                 ; Move down list
                F4      11   0193   513         BRB     10$                     ; Look for more
          2C A2 50      D0   0195   514 20$:    MOVL    R0,LIMSG$L_LCKCOUNT(R2); Return the Lock Count
                18      BA   0199   515         POPR    #^M<R3,R4>              ; Restore registers
                05      019B   516 40$:    RSB
                             019C   517
```

```
                     519          .SBTTL  LKI$SND_BLKING - Send a request for list of blocking locks
                     520          .SBTTL  LKI$SND_BLKBY - Send a request for list of blockedby locks
                     521          .SBTTL  LKI$SND_LOCKS - Send a request for list of all locks
                     522
                     523 ;++
                     524 ; FUNCTIONAL DESCRIPTION:
                     525 ;
                     526 ;       This routine handles getting a list of locks blocking/blocked by/
                     527 ;       associated with the given lock. The given lock must be valid.
                     528 ;
                     529 ; CALLING SEQUENCE:
                     530 ;
                     531 ;       BSB/JSB         LKI$SND_BLKING
                     532 ;       BSB/JSB         LKI$SND_BLKBY
                     533 ;       BSB/JSB         LKI$SND_LOCKS
                     534 ;
                     535 ;       IPL must be at IPL$_SYNCH
                     536 ;
                     537 ; INPUT PARAMETERS:
                     538 ;
                     539 ;       R3      CSID of destination system
                     540 ;       R4      Address of system buffer to receive information
                     541 ;       R6      Size of user's return buffer
                     542 ;       R9      Address of LKB
                     543 ;       R10     Address of length longword for return + scratch longword
                     544 ;
                     545 ; OUTPUT PARAMETERS:
                     546 ;
                     547 ;       R0      Completion code
                     548 ;
                     549 ; SIDE EFFECTS:
                     550 ;
                     551 ;       The process will go into MWAIT until the response from the remote
                     552 ;       system arrives.
                     553 ;
                     554 ;       IPL = ASTDEL
                     555 ;
                     556 ;       R1 - R3, R5 destroyed
                     557 ;
                     558 ;--
                     559          .ENABL  LSB
                     560
                     561 LKI$SND_BLKING::
6A   0206'CF   9E    562          MOVAB   W^SENDBLKINGREQ,(R10)   ; Store address of action routine
       0C      11    563          BRB     10$                     ; Join common code
                     564
                     565 LKI$SND_BLKBY::
6A   020E'CF   9E    566          MOVAB   W^SENDBLKBYREQ,(R10)    ; Store address of action routine
       05      11    567          BRB     10$                     ; Join common code
                     568
                     569 LKI$SND_LOCKS::
6A   0216'CF   9E    570          MOVAB   W^SENDLOCKSREQ,(R10)    ; Store address of action routine
                     571
       54      DD    572 10$:     PUSHL   R4                      ; Save system buffer address
    FE4C'      30    573          BSBW    CNX$ALLOC_WARMCDRP      ; Alloc. a CDRP with RSPID and cvt CSID
  4B 50        E9    574          BLBC    R0,90$                  ; No CDRPs or CSID error
3C A5  54      D0    575          MOVL    R4,CDRP$L_VAL5(R5)      ; Save system buffer address
```

```
                                                J 11
50   00000000'GF   D0  01BB   576          MOVL    G^SCH$GL_CURPCB,R0      ; Get our PCB address
       60 A0       D0  01C2   577          MOVL    PCB$L_PID(R0),-         ; Save PID in CDRP
         30 A5           01C5   578                  CDRP$L_VAL2(R5)
     34 A5   59    D0  01C7   579          MOVL    R9,CDRP$L_VAL3(R5)      ; Copy address of LKB
     38 A5   5A    D0  01CB   580          MOVL    R10,CDRP$L_VAL4(R5)     ; Save length longword
       50    6A    D0  01CF   581          MOVL    (R10),R0                ; Copy address of action routine
   02 AA     00'   B0  01D2   582          MOVW    S^#SS$_NORMAL,2(R10)    ; Assume success
       6A    56    B0  01D6   583          MOVW    R6,(R10)                ; Set size of return
          04 AA    B4  01D9   584          CLRW    4(R10)                  ; Indicate transaction not complete yet
             60    16  01DC   585          JSB     (R0)                    ; Call action routine
                       01DE   586
     19 04 AA      E8  01DE   587 30$:     BLBS    4(R10),50$              ; Continue if transaction completed
                       01E2   588
                       01E2   589          ; Put the process into MWAIT until the response arrives.
                       01E2   590
       50    0D    D0  01E2   591          MOVL    #RSN$_SCS,R0            ; Go into MWAIT for resource RSN$_SCS
             10    78  01E5   592          ASHL    #PSL$V_IPL,-            ; Create a PSL on stack with IPL
          7E   02          01E7   593                  #IPL$_ASTDEL,-(SP)      ;   set to ASTDEL
54   00000000'GF   D0  01E9   594          MOVL    G^SCH$GL_CURPCB,R4      ; Get our PCB address
     00000000'GF   16  01F0   595          JSB     G^SCH$RWAIT             ; Wait
                       01F6   596
                       01F6   597          ; Upon reawakening, IPL = ASTDEL; we must raise to IPL = SYNCH
                       01F6   598
                       01F6   599          SETIPL  #IPL$_SYNCH             ; Raise IPL - to lock out completions
             E3   11   01F9   600          BRB     30$                     ; Check that transaction is complete
                       01FB   601
                       01FB   602 50$:     ; Lower IPL to IPL = ASTDEL
                       01FB   603          ;
                       01FB   604          ; Inputs:
                       01FB   605          ;       R10     Address of length longword
                       01FB   606
                       01FB   607          SETIPL  #IPL$_ASTDEL            ; Lower IPL, in case we raised it
50   02 AA     3C    01FE   608          MOVZWL  2(R10),R0               ; Get status return
                       0202   609
       54 8ED0         0202   610 90$:     POPL    R4                      ; Restore system buffer address
             05        0205   611          RSB
                       0206   612
                       0206   613          .DSABL  LSB
                       0206   614
```

```
                        0206   616          .SBTTL  SENDBLKINGREQ - Send request for blocking locks to remote system
                        0206   617          .SBTTL  SENDBLKBYREQ - Send request for blocked locks to remote system
                        0206   618          .SBTTL  SENDLOCKSREQ - Send request for all locks to remote system
                        0206   619
                        0206   620 ;++
                        0206   621 ; FUNCTIONAL DESCRIPTION:
                        0206   622 ;
                        0206   623 ;       This routine takes care of building and sending the request for
                        0206   624 ;       list of blocking/blocked/all locks.  It also handles the fork level
                        0206   625 ;       processing of the response.  We are the process system; the remote
                        0206   626 ;       system is the master system.  SENDxxxREQ is called to send an
                        0206   627 ;       information request that requires a response.
                        0206   628 ;
                        0206   629 ; CALLING SEQUENCE:
                        0206   630 ;
                        0206   631 ;       BSB/JSB         SENDBLKINGREQ
                        0206   632 ;       BSB/JSB         SENDBLKBYREQ
                        0206   633 ;       BSB/JSB         SENDLOCKSREQ
                        0206   634 ;
                        0206   635 ;       IPL must be at SCS fork IPL (8)
                        0206   636 ;
                        0206   637 ;       This routine operates as a fork process so it may return
                        0206   638 ;       to its caller before completing.
                        0206   639 ;
                        0206   640 ; INPUT PARAMETERS:
                        0206   641 ;
                        0206   642 ;       R3      CSB address of destination system
                        0206   643 ;       R5      CDRP address
                        0206   644 ;
                        0206   645 ; OUTPUT PARAMETERS:
                        0206   646 ;
                        0206   647 ;       NONE
                        0206   648 ;
                        0206   649 ; SIDE EFFECTS:
                        0206   650 ;
                        0206   651 ;       NONE
                        0206   652 ;
                        0206   653 ;--
                        0206   654          .ENABL  LSB
                        0206   655
                        0206   656 SENDBLKINGREQ:
027D'CF    9E           0206   657          MOVAB   W^BLD_BLKINGMSGR,-      ; Store address of message build
  4C A5                 020A   658                  CDRP$L_MSGBLD(R5)       ; routine
      0E    11          020C   659          BRB     10$                     ; Join common code
                        020E   660
                        020E   661 SENDBLKBYREQ:
0285'CF    9E           020E   662          MOVAB   W^BLD_BLKBYMSGR,-       ; Store address of message build
  4C A5                 0212   663                  CDRP$L_MSGBLD(R5)       ; routine
      06    11          0214   664          BRB     10$                     ; Join common code
                        0216   665
                        0216   666 SENDLOCKSREQ:
028D'CF    9E           0216   667          MOVAB   W^BLD_LOCKSMSGR,-       ; Store address of message build
  4C A5                 021A   668                  CDRP$L_MSGBLD(R5)       ; routine
                        021C   669
FE00 8F    AB           021C   670 10$:     BICW3   #^C<VA$M_BYTE>,-        ; Get byte offset
  3C A5                 0220   671                  CDRP$L_VAL5(R5),-       ;
  44 A5                 0222   672                  CDRP$W_CNXBOFF(R5)      ;  to start of buffer
```

```
                 15   09    EF  0224   673          EXTZV   #VA$V_VPN,#VA$S_VPN,-   ; Get virtual page number
               52   3C A5         0227   674                  CDRP$L_VAL5(R5),R2      ;  of buffer
          50   00000000'GF    D0  022A   675          MOVL    G^MMG$GL_SPTBASE,R0     ; Get the base address of the SPT
              40 A5   6042    DE  0231   676          MOVAL   (R0)[R2],CDRP$L_CNXSVAPTE(R5) ; Set address of the first PTE
                    38 B5     3C  0236   677          MOVZWL  @CDRP$L_VAL4(R5),-      ; Set size of system buffer
                    46 A5         0239   678                  CDRP$L_CNXBCNT(R5)      ;
                    4A A5     94  023B   679          CLRB    CDRP$B_CNXRMOD(R5)      ; In kernel mode.
               53   4C A3    D0  023E   680          MOVL    CSB$L_CSID(R3),R3       ; Get CSID
                     FDBB'    30  0242   681          BSBW    CNX$BLOCK_XFER          ; Request block transfer
               51   38 A5    D0  0245   682          MOVL    CDRP$L_VAL4(R5),R1      ; Get return longword
                    04 A1    B6  0249   683          INCW    4(R1)                   ; Indicate that transaction is complete
                    1A 50    E9  024C   684          BLBC    R0,90$                  ; Dest. system is no longer in cluster
                                024F   685
                                024F   686          ; We are resumed here when the response message arrives.
                                024F   687          ; Registers contain:
                                024F   688          ;       R2      Address of message buffer
                                024F   689          ;       R3      CSB address
                                024F   690          ;       R4      Address of PDT
                                024F   691          ;       R5      Address of CDRP
                                024F   692
                                024F   693          ; Request for information was successfull, wake process
                                024F   694
                                024F   695          DISPATCH        LIMSG$B_STATUS(R2),TYPE=B,PREFIX=LIMSG$K_,-
                                024F   696                  <-
                                024F   697                  <RSPSUCCESS,70$>,-      ; Success
                                024F   698                  <RSPIVLKID,50$>,-       ; Invalid lockid
                                024F   699                  >
                                0258   700
                                0258   701          BUG_CHECK       LOCKMGRERR,FATAL
                                025C   702
                                025C   703
                                025C   704          ; Invalid lock id
                                025C   705
02 A1    0000'8F    B0  025C   706 50$:     MOVW    #SS$_IVLOCKID,2(R1)     ; Return error
                                0262   707
                                0262   708          ; Success
                                0262   709
                 61   22 A2    B0  0262   710 70$:     MOVW    LIMSG$W_LISTSIZE(R2),(R1) ; Set size of received message
                       FE38    31  0266   711          BRW     DEALL_WARMCDRP          ; Deallocate CDRP etc, wake up process
                                0269   712
                                0269   713          ; Error - node is leaving the cluster
                                0269   714
                 02 A1    50  B0  0269   715 90$:     MOVW    R0,2(R1)                ; Set return status
                    50    55  D0  026D   716          MOVL    R5,R0                   ; Get address of CDRP
                54   30 A5    3C  0270   717          MOVZWL  CDRP$L_VAL2(R5),R4      ; Get process index
              00000000'GF    16  0274   718          JSB     G^EXE$DEANONPAGED       ; Deallocate the CDRP
                       FE2B    31  027A   719          BRW     WAKE_PROCESS            ; And wake process
                                027D   720
                                027D   721          .DSABL  LSB
                                027D   722
```

```
                                027D   724 ;*****************************************************************************
                                027D   725 ;
                                027D   726 ;               Build request message to get list of blocking locks
                                027D   727 ;
                                027D   728 ;*****************************************************************************
                                027D   729
                                027D   730         ; Action routine to build the actual request message
                                027D   731         ; Inputs are:
                                027D   732         ;       R2      Address of message buffer
                                027D   733         ;       R5      Address of CDRP
                                027D   734         ; All registers except R0 and R1 must be preserved
                                027D   735
                                027D   736         .ENABL  LSB
                                027D   737
                                027D   738         ASSUME  LIMSG$L_PRCLKID  EQ  LIMSG$L_MSTLKID+4
                                027D   739         ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
                                027D   740
                                027D   741 BLD_BLKINGMSGR:
        08 A2   0205 8F   B0    027D   742         MOVW    #LIMSG$K_BLKING@8-      ; Store function and facility codes
                                0283   743                 !CLSMSG$K_FAC_LKI,CLSMSG$B_FACILITY(R2)
                      0E   11   0283   744         BRB     10$                     ; Join common code
                                0285   745
                                0285   746 BLD_BLKBYMSGR:
        08 A2   0305 8F   B0    0285   747         MOVW    #LIMSG$K_BLKBY@8-       ; Store function and facility codes
                                028B   748                 !CLSMSG$K_FAC_LKI,CLSMSG$B_FACILITY(R2)
                      06   11   028B   749         BRB     10$                     ; Join common code
                                028D   750
                                028D   751 BLD_LOCKSMSGR:
        08 A2   0405 8F   B0    028D   752         MOVW    #LIMSG$K_LOCKS@8-       ; Store function and facility codes
                                0293   753                 !CLSMSG$K_FAC_LKI,CLSMSG$B_FACILITY(R2)
                                0293   754
          50    34 A5    D0    0293   755 10$:     MOVL    CDRP$L_VAL3(R5),R0      ; Get LKB address
    51    52    18   C1    0297   756         ADDL3   #LIMSG$L_MSTLKID,R2,R1  ; Point into message buffer
          81    54 A0    D0    029B   757         MOVL    LKB$L_REMLKID(R0),(R1)+ ; Store master lockid
          81    30 A0    D0    029F   758         MOVL    LKB$L_LKID(R0),(R1)+    ; Store process lockid
                38 B5    B0    02A3   759         MOVW    @CDRP$L_VAL4(R5),-      ; Store return buffer size
                22 A2          02A6   760                 LIMSG$W_LISTSIZE(R2)    ;
                         05    02A8   761         RSB
                               02A9   762
                               02A9   763         .DSABL  LSB
                               02A9   764
```

```
                        02A9   766          .SBTTL  LKI$RCV_BLKING - Receive a request for list of blocking locks
                        02A9   767          .SBTTL  LKI$RCV_BLKBY - Receive a request for list of blocked locks
                        02A9   768          .SBTTL  LKI$RCV_LOCKS - Receive a request for list of all locks
                        02A9   769
                        02A9   770 ;++
                        02A9   771 ; FUNCTIONAL DESCRIPTION:
                        02A9   772 ;
                        02A9   773 ;       This routine handles receiving a request for list of locks.
                        02A9   774 ;       The given lock id must be a valid non-zero lock id.
                        02A9   775 ;
                        02A9   776 ; CALLING SEQUENCE:
                        02A9   777 ;
                        02A9   778 ;       BSB/JSB LKI$RCV_BLKING (called by the CNX received message routine)
                        02A9   779 ;       BSB/JSB LKI$RCV_BLKBY (called by the CNX received message routine)
                        02A9   780 ;       BSB/JSB LKI$RCV_LOCKS (called by the CNX received message routine)
                        02A9   781 ;
                        02A9   782 ;       IPL must be at IPL$_SYNCH
                        02A9   783 ;
                        02A9   784 ; INPUT PARAMETERS:
                        02A9   785 ;
                        02A9   786 ;       R2      Address of message buffer
                        02A9   787 ;       R3      Address of CSB
                        02A9   788 ;       R5      Address of CDRP
                        02A9   789 ;
                        02A9   790 ; OUTPUT PARAMETERS:
                        02A9   791 ;
                        02A9   792 ;       None
                        02A9   793 ;
                        02A9   794 ; SIDE EFFECTS:
                        02A9   795 ;
                        02A9   796 ;       R0 - R5 destroyed
                        02A9   797 ;--
                        02A9   798          .ENABL  LSB
                        02A9   799
                        02A9   800          ; Format of allocated buffer return information
                        02A9   801
           00000000     02A9   802          RCV_W_STATUS = 0                ; Return status
           00000002     02A9   803          RCV_W_SIZE = 2                  ; Return size
           00000004     02A9   804          RCV_L_BLDRTN = 4                ; Completion build routine
           00000008     02A9   805          RCV_T_DATA = 8                  ; Return data
                        02A9   806
                        02A9   807 LKI$RCV_BLKING::
038C'CF    9E           02A9   808          MOVAB   W^RCVBLKINGREQ,-        ; Store address of action routine
  30 A5                 02AD   809                  CDRP$L_VAL2(R5)         ;   in CDRP
036B'CF    9E           02AF   810          MOVAB   W^BLD_BLKINGRSP,-       ; Store address of completion routine
  34 A5                 02B3   811                  CDRP$L_VAL3(R5)         ;   in CDRP
     1A    11           02B5   812          BRB     10$                     ; Join common code
                        02B7   813
                        02B7   814 LKI$RCV_BLKBY::
0396'CF    9E           02B7   815          MOVAB   W^RCVBLKBYREQ,-         ; Store address of action routine
  30 A5                 02BB   816                  CDRP$L_VAL2(R5)         ;   in CDRP
0373'CF    9E           02BD   817          MOVAB   W^BLD_BLKBYRSP,-        ; Store address of completion routine
  34 A5                 02C1   818                  CDRP$L_VAL3(R5)         ;   in CDRP
     0C    11           02C3   819          BRB     10$                     ; Join common code
                        02C5   820
                        02C5   821 LKI$RCV_LOCKS::
03A0'CF    9E           02C5   822          MOVAB   W^RCVLOCKSREQ,-         ; Store address of action routine
```

```
                30 A5        02C9   823                 CDRP$L_VAL2(R5)         ;   in CDRP
            037B'CF    9E    02CB   824         MOVAB   W^BLD_LOCKSRSP,-        ; Store address of completion routine
                34 A5        02CF   825                 CDRP$L_VAL3(R5)         ;   in CDRP
                             02D1   826
          51    22 A2    3C   02D1   827 10$:    MOVZWL  LIMSG$W_LISTSIZE(R2),R1 ; Get requested buffer size
             51    08    C0   02D5   828         ADDL    #RCV_T_DATA,R1          ; Make room for size field + return status
       54    0362'CF    9E   02D8   829         MOVAB   W^RCV_ERROR,R4          ; Set address of error processing routine
                FD20'    30   02DD   830         BSBW    CNX$PARTNER_INIT_CSB    ; Init partner's CSB
             07C0 8F    BB   02E0   831         PUSHR   #^M<R6,R7,R8,R9,R10>    ; Save registers
       58    22 A2    3C   02E4   832         MOVZWL  LIMSG$W_LISTSIZE(R2),R8 ; Get requested buffer size
    57 52    18    C1   02E8   833         ADDL3   #LIMSG$C_MSTLKID,R2,R7  ; Point R7 into message buffer
                02    90   02EC   834         MOVB    #LIMSG$K_RSPIVLKID,-    ; Assume error on lock id
                64         02EE   835                 RCV_W_STATUS(R4)
    5A    02 A4    9E   02EF   836         MOVAB   RCV_W_SIZE(R4),R10      ; Point R10 to length word
             34 A5    D0   02F3   837         MOVL    CDRP$C_VAL3(R5),-       ; Save completion routine address
             04 A4        02F6   838                 RCV_L_BLDRTN(R4)
                6A    B4   02F8   839         CLRW    (R10)                   ; Assume no data to return
    52    08 A4    9E   02FA   840         MOVAB   RCV_T_DATA(R4),R2       ; Point R2 to start of data area
          51    87    D0   02FE   841         MOVL    (R7)+,R1                ; Get master lockid (id on this system)
          50    87    D0   0301   842         MOVL    (R7)+,R0                ; Get process lockid (for verify)
             0133    30   0304   843         BSBW    VERIFYREMLKID           ; Convert to LKB address
             43 50    E9   0307   844         BLBC    R0,RCV_IVLKID           ; Invalid lock id
                             030A   845
                             030A   846         ; Have LKB address in R6, return buffer in R2. Setup other registers.
                             030A   847
          59    56   D0   030A   848         MOVL    R6,R9                   ; Copy LKB address
          56    58   D0   030D   849         MOVL    R8,R6                   ; Copy size of return data buffer
    58    50 A9    D0   0310   850         MOVL    LKB$L_RSB(R9),R8        ; Get RSB address
             30 B5    16   0314   851         JSB     @CDRP$L_VAL2(R5)        ; Call action routine
                             0317   852
                             0317   853         ; Send data back to requestor
                             0317   854
44 A5    52    FE00 8F    AB   0317   855         BICW3   #^C<VA$M_BYTE>,R2,-     ; Get byte offset
                15    09   EF   031E   856                 CDRP$W_CNXBOFF(R5)      ;  to start of buffer
                51    52        0321   857         EXTZV   #VA$V_VPN,#VA$S_VPN,-   ; Get virtual page number
                                      858                 R2,R1                   ;  of buffer
     50    00000000'GF    D0   0323   859         MOVL    G^MMG$GL_SPTBASE,R0     ; Get the base address of the SPT
        40 A5    6041    DE   032A   860         MOVAL   (R0)[R1],CDRP$L_CNXSVAPTE(R5) ; Set address of the first PTE
           46 A5    6A    3C   032F   861         MOVZWL  (R10),CDRP$L_CNXBCNT(R5); Set size of system buffer
              4A A5    94   0333   862         CLRB    CDRP$B_CNXRMOD(R5)      ; In kernel mode.
              30 A5    D4   0336   863         CLRL    CDRP$L_LBOFF(R5)        ; Start at beginning of buffer
              38 A5    D4   0339   864         CLRL    CDRP$L_RBOFF(R5)        ;  on both sides
              46 A5    D0   033C   865         MOVL    CDRP$L_CNXBCNT(R5),-    ; Set size of system buffer
              3C A5        033F   866                 CDRP$L_XCT_LEN(R5)
             07C0 8F    BA   0341   867         POPR    #^M<R6,R7,R8,R9,R10>    ; Restore registers, in case of fork
                FCB8'    30   0345   868         BSBW    CNX$BLOCK_WRITE         ; Send the data
                01    90   0348   869         MOVB    #LIMSG$K_RSPSUCCESS,-   ; Store success status indicator
                64         034A   870                 RCV_W_STATUS(R4)
                             034B   871
                             034B   872         ; Send response message.  CDRP address is in R5, allocated buffer in R4.
                             034B   873
                04    11   034B   874         BRB     70$                     ; Continue
                             034D   875
                             034D   876 RCV_IVLKID:
             07C0 8F    BA   034D   877         POPR    #^M<R6,R7,R8,R9,R10>    ; Restore registers
               02 A4    B0   0351   878 70$:    MOVW    RCV_W_SIZE(R4),-        ; Save return size
               32 A5        0354   879                 CDRP$C_VAL2+2(R5)       ;  in CDRP
```

```
                64   90  0356   880          MOVB    RCV_W_STATUS(R4),-      ; Save return status
            30  A5       0358   881                  CDRP$C_VAL2(R5)         ;  in CDRP
            04  A4   D0   035A   882          MOVL    RCV_L_BLDRTN(R4),-      ; Store address of response message
            4C  A5       035D   883                  CDRP$C_MSGBLD(R5)       ;  build routine
              FC9E'  31   035F   884          BRW     CNX$PARTNER_FINISH      ; Send response and close connection
                         0362   885
                         0362   886          .DSABL  LSB
                         0362   887
                         0362   888          ; Error processing routine
                         0362   889
                         0362   890 RCV_ERROR:                               ; Error processing
              50    55   D0   0362   891          MOVL    R5,R0               ; Copy CDRP address
        00000000'GF      17   0365   892          JMP     G^EXE$DEANONPAGED   ; Deallocate the CDRP
                         036B   893
```

```
                        036B    895 ;*****************************************************************************
                        036B    896 ;
                        036B    897 ;               Action routines to build response messages
                        036B    898 ;
                        036B    899 ;*****************************************************************************
                        036B    900
                        036B    901         .ENABL  LSB
                        036B    902
                        036B    903         ; These routines build the response messages for received requests
                        036B    904         ; Inputs:
                        036B    905         ;       R2      Address of message buffer
                        036B    906         ;       R5      Address of CDRP
                        036B    907         ; All registers except R0 and R1 must be preserved
                        036B    908
                        036B    909         ASSUME  CLSMSG$B_FUNC  EQ  CLSMSG$B_FACILITY+1
                        036B    910
                        036B    911 BLD_BLKINGRSP:
08 A2   0285 8F   B0    036B    912         MOVW    #LIMSG$K_BLKING@8-      ; Store function and facility codes
                        0371    913                 !CLSMSG$M_RESPMSG-
                        0371    914                 !CLSMSG$K_FAC_LKI,CLSMSG$B_FACILITY(R2)
            0E    11    0371    915         BRB     10$                     ; Join common code
                        0373    916
                        0373    917 BLD_BLKBYRSP:
08 A2   0385 8F   B0    0373    918         MOVW    #LIMSG$K_BLKBY@8-       ; Store function and facility codes
                        0379    919                 !CLSMSG$M_RESPMSG-
                        0379    920                 !CLSMSG$K_FAC_LKI,CLSMSG$B_FACILITY(R2)
            06    11    0379    921         BRB     10$                     ; Join common code
                        037B    922
                        037B    923 BLD_LOCKSRSP:
08 A2   0485 8F   B0    037B    924         MOVW    #LIMSG$K_LOCKS@8-       ; Store function and facility codes
                        0381    925                 !CLSMSG$M_RESPMSG-
                        0381    926                 !CLSMSG$K_FAC_LKI,CLSMSG$B_FACILITY(R2)
                        0381    927
      30 A5     90    0381    928 10$:    MOVB    CDRP$L_VAL2(R5),-       ; Store response status
      20 A2           0384    929                 LIMSG$B_STATUS(R2)
      32 A5     B0    0386    930         MOVW    CDRP$L_VAL2+2(R5),-     ; Store response length
      22 A2           0389    931                 LIMSG$W_LISTSIZE(R2)
            05    038B    932         RSB
                        038C    933
                        038C    934
                        038C    935         .DSABL  LSB
                        038C    936
```

```
                     038C   938 ;*****************************************************************************
                     038C   939 ;
                     038C   940 ;                       Build list of blocking locks
                     038C   941 ;
                     038C   942 ;*****************************************************************************
                     038C   943
                     038C   944         ; Action routine to build the data transfer message
                     038C   945         ; Inputs are:
                     038C   946         ;       R2      Address of return buffer
                     038C   947         ;       R6      Size of return buffer
                     038C   948         ;       R8      Address of RSB
                     038C   949         ;       R9      Address of LKB
                     038C   950         ;       R10     Address of size word for return data
                     038C   951         ; R0-R4,R7 may be destroyed
                     038C   952
                     038C   953 RCVBLKINGREQ:
          58    20 A8  9E  038C   954         MOVAB   RSB$L_WTQFL(R8),R8      ; Point R8 to wait queue listhead
00000000'GF     17   0390   955         JMP     G^LKI$SEARCH_BLOCKING   ; Search for all blocking locks,
                     0396   956                                         ;  and return
                     0396   957
                     0396   958
                     0396   959 ;*****************************************************************************
                     0396   960 ;
                     0396   961 ;                       Build list of blocked locks
                     0396   962 ;
                     0396   963 ;*****************************************************************************
                     0396   964
                     0396   965         ; Action routine to build the block transfer message
                     0396   966         ; Inputs are:
                     0396   967         ;       R2      Address of return buffer
                     0396   968         ;       R6      Size of return buffer
                     0396   969         ;       R8      Address of RSB
                     0396   970         ;       R9      Address of LKB
                     0396   971         ;       R10     Address of size word for return data
                     0396   972         ; R0-R4,R7 may be destroyed
                     0396   973
                     0396   974 RCVBLKBYREQ:
          58    10 A8  9E  0396   975         MOVAB   RSB$L_GRQFL(R8),R8      ; Point R8 to granted queue listhead
00000000'GF     17   039A   976         JMP     G^LKI$SEARCH_BLOCKEDBY  ; Search for all blocked locks,
                     03A0   977                                         ;  and return
                     03A0   978
                     03A0   979
                     03A0   980 ;*****************************************************************************
                     03A0   981 ;
                     03A0   982 ;                       Build list of all locks
                     03A0   983 ;
                     03A0   984 ;*****************************************************************************
                     03A0   985
                     03A0   986         ; Action routine to build the block transfer message
                     03A0   987         ; Inputs are:
                     03A0   988         ;       R2      Address of return buffer
                     03A0   989         ;       R6      Size of return buffer
                     03A0   990         ;       R8      Address of RSB
                     03A0   991         ;       R7      Scratch
                     03A0   992         ;       R9      Address of LKB
                     03A0   993         ;       R10     Address of size word for return data
                     03A0   994         ; R0-R4,R7 may be destroyed
```

```
                03A0   995
                03A0   996 RCVLOCKSREQ:
        52   DD 03A0   997          PUSHL   R2                              ; Save address of return buffer
58    10 A8   9E 03A2   998          MOVAB   RSB$L_GRQFL(R8),R8              ; Point R8 to granted queue listhead
   51   56   D0 03A6   999          MOVL    R6,R1                           ; Get size of buffer
                03A9  1000          ASSUME  RSB$L_CVTQFL EQ RSB$L_GRQFL+8
                03A9  1001          ASSUME  RSB$L_WTQFL EQ RSB$L_CVTQFL+8
   53   03   9A 03A9  1002          MOVZBL  #3,R3                           ; Initialize number of queues to search
   57   58   D0 03AC  1003 30$:     MOVL    R8,R7                           ; Copy listhead address, again
   58   67   D1 03AF  1004 50$:     CMPL    (R7),R8                         ; Back at listhead again?
        14   13 03B2  1005          BEQL    60$                             ; Br if yes
   51   18   C2 03B4  1006          SUBL    #LKI$C_LENGTH,R1                ; Any room left in buffer?
        15   19 03B7  1007          BLSS    90$                             ; Br if not
   57   67   D0 03B9  1008          MOVL    (R7),R7                         ; Else, move down list
57    C8 A7   9E 03BC  1009          MOVAB   -LKB$L_SQFL(R7),R7              ; Point to start of LKB
        10   10 03C0  1010          BSBB    LOCK_INFO                       ; Get the lock information
57    38 A7   9E 03C2  1011          MOVAB   LKB$L_SQFL(R7),R7               ; Point back to state queue
        E7   11 03C6  1012          BRB     50$                             ; Look for more
                03C8  1013 60$:     ASSUME  RSB$L_CVTQFL EQ RSB$L_GRQFL+8
                03C8  1014          ASSUME  RSB$L_WTQFL EQ RSB$L_CVTQFL+8
   58   08   C0 03C8  1015          ADDL    #8,R8                           ; Skip to next queue
     DE 53   F5 03CB  1016          SOBGTR  R3,30$                          ; Loop if more queues to search
        52 8ED0 03CE  1017 90$:     POPL    R2                              ; Restore address of return buffer
           05   03D1  1018          RSB                                     ; Return to caller
                03D2  1019
```

```
                     03D2  1021 ;+
                     03D2  1022 ; Return Lock Information
                     03D2  1023 ;
                     03D2  1024 ;       This routine will return the following lock information:
                     03D2  1025 ;
                     03D2  1026 ;               LKI$_LOCKID     - the lock's lock id
                     03D2  1027 ;               LKI$_PID        - the lock's PID
                     03D2  1028 ;               LKI$_SYSTEM     - the resource's system id (CSID)
                     03D2  1029 ;               LKI$_STATE      - the locks current state
                     03D2  1030 ;               LKI$_REMLKID    - the lock's remote id
                     03D2  1031 ;               LKI$_REMSYSTEM  - the lock's remote system id (CSID)
                     03D2  1032 ;
                     03D2  1033 ; Inputs:
                     03D2  1034 ;       R2 = Output buffer address
                     03D2  1035 ;       R7 = LKB address
                     03D2  1036 ;       R10 = Address of size word for return data
                     03D2  1037 ;
                     03D2  1038 ; Outputs:
                     03D2  1039 ;       None
                     03D2  1040 ;
                     03D2  1041 ; Side Effects:
                     03D2  1042 ;       R0 is destroyed
                     03D2  1043 ;       (R10) is increased by lock return size
                     03D2  1044 ;-
                     03D2  1045 LOCK_INFO:
    6A    18    A0   03D2  1046         ADDW    #LKI$C_LENGTH,(R10)     ; Tally return size
82    30    A7    D0 03D5  1047         MOVL    LKB$L_LKID(R7),(R2)+    ; Return the LOCKID
                     03D9  1048         ;
                     03D9  1049         ;  The EPID in the LKB is valid only for a master lock block.
                     03D9  1050         ;
50    14    A7    D0 03D9  1051         MOVL    LKB$L_EPID(R7),R0       ; Get the EPID
            04    E0 03DD  1052         BBS     #LKB$V_MSTCPY,-         ; Br if master copy
      0A    2A    A7       03DF  1053                 LKB$W_STATUS(R7),10$    ; ...EPID is valid
50    0C    A7    D0 03E2  1054         MOVL    LKB$L_PID(R7),R0        ; Else, get PID and
00000000'GF       16 03E6  1055         JSB     G^EXE$IPID_TO_EPID      ; Convert to EPID
      82          50 03EC  1056 10$:    MOVL    R0,(R2)+                ; Return the EPID
50    50    A7    D0 03EF  1057         MOVL    LKB$L_RSB(R7),R0        ; Get RSB address
82    38    A0    D0 03F3  1058         MOVL    RSB$L_CSID(R0),(R2)+    ; Return the SYSTEM ID
            0E    12 03F7  1059         BNEQ    30$                     ; Br if non-zero - okay
50    00000000'GF       D0 03F9  1060         MOVL    G^CLU$GL_CLUB,R0        ; Else, get CLUB address
            05    13 0400  1061         BEQL    30$                     ; Br if no cluster
   FC A2    60    A0   D0 0402  1062         MOVL    CLUB$L_LOCAL_CSID(R0),-4(R2) ; Return real CSID
                     0407  1063 30$:    ASSUME  LKB$B_GRMODE EQ LKB$B_RQMODE+1
82    34    A7    B0 0407  1064         MOVW    LKB$B_RQMODE(R7),(R2)+  ; Copy modes
82    36    A7    9B 040B  1065         MOVZBW  LKB$B_STATE(R7),(R2)+   ; Copy current state, zero byte
            05    18 040F  1066         BGEQ    40$                     ; Br if state is okay
   FE A2    FF    8F 0411  1067         MOVB    #LKI$C_WAITING,-2(R2)   ; Else, map waiting states to same code
                     0416  1068 40$:    ;
                     0416  1069         ;  The remote CSID and REMLKID are only valid in master
                     0416  1070         ;  copy lock blocks.
                     0416  1071         ;
82    54    A7    D0 0416  1072         MOVL    LKB$L_REMLKID(R7),(R2)+ ; Copy the REMLKID
82    58    A7    D0 041A  1073         MOVL    LKB$L_CSID(R7),(R2)+    ; Get the remote CSID
            04    E0 041E  1074         BBS     #LKB$V_MSTCPY,-         ; Br if master copy
      16    2A    A7       0420  1075                 LKB$W_STATUS(R7),90$    ; ...CSID, REMLKID are valid
   F8 A2    50    A7   D0 0423  1076         MOVL    LKB$L_LKID(R7),-8(R2)   ; Else, return the LOCKID as REMLKID
50    00000000'GF       D0 0428  1077         MOVL    G^CLU$GL_CLUB,R0        ; Get CLUB address
```

```
                                                H 12
                     04   13  042F  1078          BEQL    70$                             ; Br if no cluster, zero CSID
          50    60 A0   D0  0431  1079          MOVL    CLUB$L_LOCAL_CSID(R0),R0        ; Else, get real CSID
          FC A2   50   D0  0435  1080 70$:      MOVL    R0,-4(R2)                       ; Return real CSID
                       05  0439  1081 90$:      RSB
                           043A  1082
```

```
                     043A  1084          .SBTTL  VERIFYREMLKID - Verify remote lock id
                     043A  1085
                     043A  1086 ;++
                     043A  1087 ; FUNCTIONAL DESCRIPTION:
                     043A  1088 ;
                     043A  1089 ;       This routine verifies a lock id sent by another system
                     043A  1090 ;       and converts it into a LKB address.
                     043A  1091 ;
                     043A  1092 ; CALLING SEQUENCE:
                     043A  1093 ;
                     043A  1094 ;       BSBW    VERIFYREMLKID
                     043A  1095 ;       IPL must be at IPL$_SYNCH
                     043A  1096 ;
                     043A  1097 ; INPUT PARAMETERS:
                     043A  1098 ;
                     043A  1099 ;       R0      Lock id on remote system
                     043A  1100 ;       R1      Lock id on this system
                     043A  1101 ;
                     043A  1102 ; OUTPUT PARAMETERS:
                     043A  1103 ;
                     043A  1104 ;       R0      Completion code
                     043A  1105 ;       R6      Address of LKB
                     043A  1106 ;
                     043A  1107 ; COMPLETION CODES:
                     043A  1108 ;
                     043A  1109 ;       SS$_NORMAL      Lock id was valid can converted to LKB address
                     043A  1110 ;       SS$_IVLOCKID    Invalid lock id
                     043A  1111 ;
                     043A  1112 ; SIDE EFFECTS:
                     043A  1113 ;
                     043A  1114 ;       None.
                     043A  1115 ;
                     043A  1116 ; NOTE:
                     043A  1117 ;
                     043A  1118 ;       This routine does two consistency checks.  The first is that
                     043A  1119 ;       it verifies the lock id is valid via the sequence number check.
                     043A  1120 ;       If the lock id fails this check, then an error is returned to
                     043A  1121 ;       the caller as this is allowed in some cases and is fatal in others.
                     043A  1122 ;       However, if the lock id passes this check, then another check
                     043A  1123 ;       is made that compares the remote lock id as sent by the remote
                     043A  1124 ;       system with the remote lock id stored here in the LKB.  If this
                     043A  1125 ;       check fails then it is immediately fatal as the first check should
                     043A  1126 ;       catch all races that cause lock ids to not match across systems.
                     043A  1127 ;       Also note that this second check is not perfect in that it should
                     043A  1128 ;       also check CSB addresses.  This is considered unnecessary as the
                     043A  1129 ;       additional protection that check offers is small.
                     043A  1130 ;--
                     043A  1131
                     043A  1132 VERIFYREMLKID:
            54   DD  043A  1133          PUSHL   R4                      ; Save R4
          56 51  3C  043C  1134          MOVZWL  R1,R6                   ; Put lockid index in R6
00000000'GF 56  D1  043F  1135          CMPL    R6,G^LCK$GL_MAXID       ; Is the lock id too big?
            20  1A  0446  1136          BGTRU   40$                     ; Yes
54  00000000'GF  D0  0448  1137          MOVL    G^LCK$GL_IDTBL,R4       ; *** May combine with next instr.
       56 6446  D0  044F  1138          MOVL    (R4)[R6],R6             ; Get LKB address
            13  18  0453  1139          BGEQ    40$                     ; Unallocated id
      30 A6 51  D1  0455  1140          CMPL    R1,LKB$L_LKID(R6)       ; Check sequence number
```

```
                 0D   12  0459  1141          BNEQ    40$                             ; Not valid
          54 A6  50   D1  045B  1142          CMPL    R0,LKB$L_REMLKID(R6)            ; Check remote lock id
                 10   12  045F  1143          BNEQ    50$                             ; Doesn't match
             50  00'  9A  0461  1144          MOVZBL  S^#SS$_NORMAL,R0                ; Success
                 54 8ED0  0464  1145          POPL    R4                              ; Restore R4
                    05  0467  1146          RSB
                        0468  1147
        50  0000'8F   3C  0468  1148 40$:    MOVZWL  #SS$_IVLOCKID,R0                ; Invalid lock id
                 54 8ED0  046D  1149          POPL    R4                              ; Restore R4
                    05  0470  1150          RSB
                        0471  1151
                        0471  1152 50$:    BUG_CHECK       INVLOCKID,FATAL; Invalid remote lockid
                        0475  1153
                        0475  1154
                        0475  1155
                        0475  1156          .END
```

```
$$BASE                          = 00000001
$$DISPL                         = 00000003
$$GENSW                         = 00000001
$$HIGH                          = 00000002
$$LIMIT                         = 00000001
$$LOW                           = 00000001
$$MNSW                          = 00000001
$$MXSW                          = 00000001
BLD_BLKBYMSGR                     00000285 R      02
BLD_BLKBYRSP                      00000373 R      02
BLD_BLKINGMSGR                    0000027D R      02
BLD_BLKINGRSP                     0000036B R      02
BLD_LOCKSMSGR                     0000028D R      02
BLD_LOCKSRSP                      0000037B R      02
BLD_STDMSGR                       000000CF R      02
BLD_STDRSP                        00000136 R      02
BUG$_INVLOCKID                    ********   X    02
BUG$_LOCKMGRERR                   ********   X    02
CDRP$B_CNXRMOD                  = 0000004A
CDRP$L_CNXBCNT                  = 00000046
CDRP$L_CNXSVAPTE                = 00000040
CDRP$L_LBOFF                    = 00000030
CDRP$L_MSGBLD                   = 0000004C
CDRP$L_RBOFF                    = 00000038
CDRP$L_VAL1                     = 0000002C
CDRP$L_VAL2                     = 00000030
CDRP$L_VAL3                     = 00000034
CDRP$L_VAL4                     = 00000038
CDRP$L_VAL5                     = 0000003C
CDRP$L_XCT_LEN                  = 0000003C
CDRP$W_CNXBOFF                  = 00000044
CLSMSG$B_FACILITY               = 00000008
CLSMSG$B_FUNC                   = 00000009
CLSMSG$K_FAC_LKI                = 00000005
CLSMSG$M_RESPMSG                = 00000080
CLU$GL_CLUB                       ********   X    02
CLUB$L_LOCAL_CSID               = 00000060
CNX$ALLOC_WARMCDRP                ********   X    02
CNX$BLOCK_WRITE                   ********   X    02
CNX$BLOCK_XFER                    ********   X    02
CNX$DEALL_WARMCDRP_CSB            ********   X    02
CNX$INIT_CDRP                     ********   X    02
CNX$PARTNER_FINISH                ********   X    02
CNX$PARTNER_INIT_CSB              ********   X    02
CNX$SEND_MSG_CSB                  ********   X    02
CNX$SEND_MSG_RESP                 ********   X    02
CSB$L_CSID                      = 0000004C
DEALL_WARMCDRP                    000000A1 R      02
DYN$C_BUFIO                     = 00000013
EVT$_AST                          ********   X    02
EXE$ALONONPAGED                   ********   X    02
EXE$DEANONPAGED                   ********   X    02
EXE$IPID_TO_EPID                  ********   X    02
FKB$B_TYPE                      = 0000000A
FKB$W_SIZE                      = 00000008
IPL$_ASTDEL                     = 00000002
IPL$_SCS                        = 00000008
IPL$_SYNCH                      = 00000008
LCK$GL_IDTBL                      ********   X    02
LCK$GL_MAXID                      ********   X    02
LIMSG$B_STATUS                  = 00000020
LIMSG$K_BLKBY                   = 00000003
LIMSG$K_BLKING                  = 00000002
LIMSG$K_LOCKS                   = 00000004
LIMSG$K_RSPIVLKID               = 00000002
LIMSG$K_RSPSUCCESS              = 00000001
LIMSG$K_STDINFO                 = 00000001
LIMSG$K_STDINFO_LEN             = 00000040
LIMSG$L_LCKCOUNT                = 0000002C
LIMSG$L_MSTLKID                 = 00000018
LIMSG$L_PRCLKID                 = 0000001C
LIMSG$L_RSBREFCNT               = 00000028
LIMSG$L_STATE                   = 00000024
LIMSG$Q_VALBLK                  = 00000030
LIMSG$W_LISTSIZE                = 00000022
LKB$B_GRMODE                    = 00000035
LKB$B_RQMODE                    = 00000034
LKB$B_STATE                     = 00000036
LKB$L_CSID                      = 00000058
LKB$L_EPID                      = 00000014
LKB$L_LKID                      = 00000030
LKB$L_PID                       = 0000000C
LKB$L_REMLKID                   = 00000054
LKB$L_RSB                       = 00000050
LKB$L_SQFL                      = 00000038
LKB$V_MSTCPY                    = 00000004
LKB$W_STATUS                    = 0000002A
LKI$C_LENGTH                    = 00000018
LKI$C_WAITING                   = FFFFFFFF
LKI$DISPATCH                      00000000 RG     02
LKI$RCV_BLKBY                     000002B7 RG     02
LKI$RCV_BLKING                    000002A9 RG     02
LKI$RCV_LOCKS                     000002C5 RG     02
LKI$RCV_STDINFO                   000000E6 RG     02
LKI$SEARCH_BLOCKEDBY              ********    X   02
LKI$SEARCH_BLOCKING               ********    X   02
LKI$SND_BLKBY                     000001A3 RG     02
LKI$SND_BLKING                    0000019C RG     02
LKI$SND_LOCKS                     000001AA RG     02
LKI$SND_STDREQ                    00000011 RG     02
LOCK_INFO                         000003D2 R      02
MMG$GL_SPTBASE                    ********    X   02
PCB$L_PID                       = 00000060
PR$_IPL                           ********    X   02
PSL$V_IPL                       = 00000010
RCVBLKBYREQ                       00000396 R      02
RCVBLKINGREQ                      0000038C R      02
RCVLOCKSREQ                       000003A0 R      02
RCV_ERROR                         00000362 R      02
RCV_IVLKID                        0000034D R      02
RCV_L_BLDRTN                    = 00000004
RCV_T_DATA                      = 00000008
RCV_W_SIZE                      = 00000002
RCV_W_STATUS                    = 00000000
```

```
RSB$L_CSID                      = 00000038
RSB$L_CVTQFL                    = 00000018
RSB$L_GRQFL                     = 00000010
RSB$L_WTQFL                     = 00000020
RSB$Q_VALBLK                    = 00000028
RSB$W_REFCNT                    = 00000040
RSN$_SCS                        = 0000000D
SCH$GL_CURPCB                     ********    X   02
SCH$GL_PCBVEC                     ********    X   02
SCH$RSE                           ********    X   02
SCH$RWAIT                         ********    X   02
SENDBLKBYREQ                      0000020E R      02
SENDBLKINGREQ                     00000206 R      02
SENDLOCKSREQ                      00000216 R      02
SENDSTDREQ                        00000070 R      02
SS$_INSFMEM                       ********    X   02
SS$_IVLOCKID                      ********    X   02
SS$_NORMAL                        ********    X   02
SS$_NOSUCHNODE                    ********    X   02
STD_IVLKID                        0000010E R      02
VA$M_BYTE                       = 000001FF
VA$S_VPN                        = 00000015
VA$V_VPN                        = 00000009
VERIFYREMLKID                     0000043A R      02
WAKE_PROCESS                      000000A8 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	00000475 (1141.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.03	00:00:01.77
Command processing	108	00:00:00.44	00:00:03.15
Pass 1	426	00:00:11.19	00:00:40.11
Symbol table sort	0	00:00:01.53	00:00:06.91
Pass 2	205	00:00:02.60	00:00:12.32
Symbol table output	17	00:00:00.09	00:00:00.09
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	789	00:00:15.90	00:01:04.37

The working set limit was 1950 pages.
91816 bytes (180 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1476 non-local and 40 local symbols.
1156 source lines were read in Pass 1, producing 18 object records in Pass 2.
33 pages of virtual memory were used to define 31 macros.

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

Macro library name                 Macros defined
------------------                 --------------
_$255$DUA28:[SYSLOA.OBJ]CLUSTER.MLB;1          1
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                17
_$255$DUA28:[SYSLIB]STARLET.MLB;2              7
TOTALS (all libraries)                        25

1596 GETS were required to define 25 macros.

There were no errors, warnings or information messages.

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CSPOPCOM.LIS

CSPWAIT.LIS

CSPRCPCAC.LIS

CSPCJFRES.LIS

CSPQUORUM.LIS

DISTRLKI.LIS

CSPMOUNT.LIS

CSPVECTOR.LIS

CSPCLIENT.LIS

DSTRLOCK.LIS

DSTRLOCK.LIS