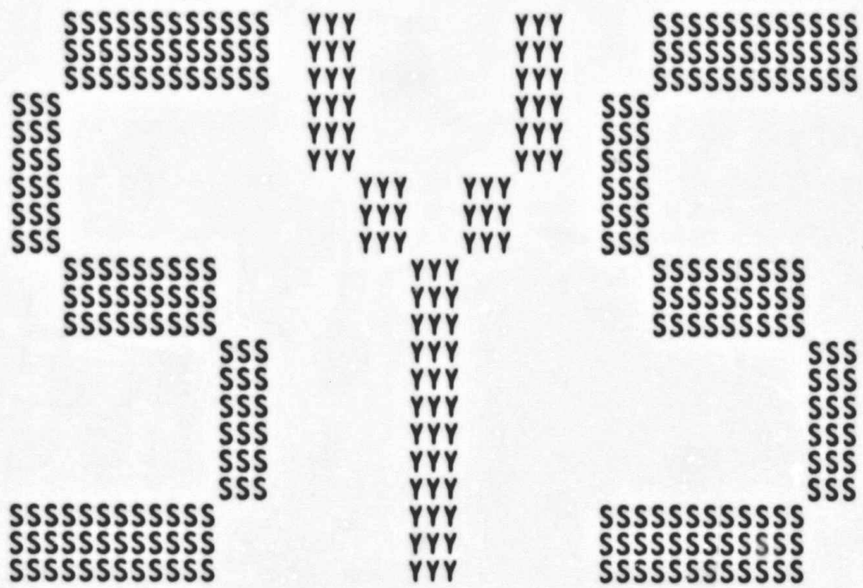
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSSSSSSSSSS YYY YYY SSSSSSSSSSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSS YYY YYY SSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSSSSSSSS YYY SSSSSSSSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSS YYY SSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS
SSSSSSSSSSS YYY SSSSSSSSSSS

```
DDDDDDDD    EEEEEEEEEE    AAAAAA    DDDDDDDD    LL          OOOOOO      CCCCCCCC  KK      KK
DDDDDDDD    EEEEEEEEEE    AAAAAA    DDDDDDDD    LL          OOOOOO      CCCCCCCC  KK      KK
DD      DD  EE            AA  AA    DD      DD  LL        OO      OO  CC          KK      KK
DD      DD  EE            AA  AA    DD      DD  LL        OO      OO  CC          KK      KK
DD      DD  EE            AA  AA    DD      DD  LL        OO      OO  CC          KK    KK
DD      DD  EE            AA  AA    DD      DD  LL        OO      OO  CC          KK    KK
DD      DD  EEEEEEEE      AA  AA    DD      DD  LL        OO      OO  CC          KKKKKK
DD      DD  EEEEEEEE      AA  AA    DD      DD  LL        OO      OO  CC          KKKKKK
DD      DD  EE            AAAAAAAAAA  DD      DD  LL        OO      OO  CC          KK    KK
DD      DD  EE            AAAAAAAAAA  DD      DD  LL        OO      OO  CC          KK    KK
DD      DD  EE            AA  AA    DD      DD  LL        OO      OO  CC          KK      KK    ....
DD      DD  EE            AA  AA    DD      DD  LL        OO      OO  CC          KK      KK    ....
DDDDDDDD    EEEEEEEEEE    AA  AA    DDDDDDDD    LLLLLLLLLL    OOOOOO      CCCCCCCC  KK      KK    ....
DDDDDDDD    EEEEEEEEEE    AA  AA    DDDDDDDD    LLLLLLLLLL    OOOOOO      CCCCCCCC  KK      KK    ....

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

Table of contents

```
0000     1          .TITLE  DEADLOCK - DEADLOCK DETECTION AND RESOLUTION
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;*****************************************************************************
0000     5  ;*                                                                           *
0000     6  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8  ;*  ALL RIGHTS RESERVED.                                                     *
0000     9  ;*                                                                           *
0000    10  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    11  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    12  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    13  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    14  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    15  ;*  TRANSFERRED.                                                             *
0000    16  ;*                                                                           *
0000    17  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    18  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    19  ;*  CORPORATION.                                                             *
0000    20  ;*                                                                           *
0000    21  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    22  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23  ;*                                                                           *
0000    24  ;*                                                                           *
0000    25  ;*****************************************************************************
0000    26
0000    27  ;++
0000    28  ; FACILITY: EXECUTIVE, SYSTEM SERVICES
0000    29  ;
0000    30  ; ABSTRACT:
0000    31  ;       This module implements deadlock detection (and resolution)
0000    32  ;       for the VMS lock manager system services ($ENQ and $DEQ).
0000    33  ;
0000    34  ; ENVIRONMENT: VAX/VMS
0000    35  ;
0000    36  ; AUTHOR: Steve Beckhardt,       CREATION DATE: 15-Jul-1981
0000    37  ;
0000    38  ; MODIFIED BY:
0000    39  ;
0000    40  ;       V03-013 SRB0150         Steve Beckhardt         21-Aug-1984
0000    41  ;               Cleared R9 prior to checking for conversion deadlocks.
0000    42  ;
0000    43  ;       V03-012 SRB0130         Steve Beckhardt         18-May-1984
0000    44  ;               Added support for LCK$M_NODLCKBLK flag and fixed bug
0000    45  ;               with LCK$M_NODLCKWT flag.
0000    46  ;
0000    47  ;       V03-011 SRB0122         Steve Beckhardt         30-Apr-1984
0000    48  ;               Fixed bug where local dequeue counter was going negative.
0000    49  ;               Fixed bug where deadlock searches were started during
0000    50  ;               state changes.
0000    51  ;
0000    52  ;       V03-010 SRB0117         Steve Beckhardt         9-Apr-1984
0000    53  ;               Added support for LCK$M_NODLCKWT flag.
0000    54  ;               Fixed bug where R9 was not preserved in LCK$DEQLOCK.
0000    55  ;               Added support for threads waiting for pool inserting
0000    56  ;               structures on the time out queue.
0000    57  ;
```

```
0000     58 ;       V03-009 SRB0115         Steve Beckhardt         5-Mar-1984
0000     59 ;               Added support for distributed deadlock detection.
0000     60 ;
0000     61 ;       V03-008 LY00B1          Larry Yetto             10-FEB-1984 09:52
0000     62 ;               Fix truncation errors
0000     63 ;
0000     64 ;       V03-007 SRB0102         Steve Beckhardt         9-Sep-1983
0000     65 ;               Fixed bug in SRB0100.
0000     66 ;
0000     67 ;       V03-006 SRB0100         Steve Beckhardt         15-Jul-1983
0000     68 ;               Added code to allow local deadlock detection to work
0000     69 ;               in a cluster.
0000     70 ;
0000     71 ;       V03-005 SRB0080         Steve Beckhardt         5-Apr-1983
0000     72 ;               Changed code for handling dequeuing deadlock victim
0000     73 ;               to use new support in LCK$DEQLOCK.
0000     74 ;
0000     75 ;       V03-004 SRB0073         Steve Beckhardt         25-Mar-1983
0000     76 ;               Added code to clear a register before calling LCK$DEQLOCK
0000     77 ;               as a result of changes to support cancelling lock requests.
0000     78 ;
0000     79 ;       V03-003 ROW0168         Ralph O. Weber          3-MAR-1983
0000     80 ;               Change external data references to G^.
0000     81 ;
0000     82 ;       V03-002 DWT0055         David Thiel             18-Jul-1982
0000     83 ;               Use L^ mode for external references to
0000     84 ;               SCH$GL_PCBVEC.
0000     85 ;
0000     86 ;       V03-001 KDM0002         Kathleen D. Morse       28-Jun-1982
0000     87 ;               Added $SSDEF.
0000     88 ;--
```

```
                     0000     90          .SBTTL  DECLARATIONS
                     0000     91 ;
                     0000     92 ; INCLUDE FILES:
                     0000     93 ;
                     0000     94
                     0000     95 ;
                     0000     96 ; EXTERNAL SYMBOLS:
                     0000     97 ;
                     0000     98
                     0000     99          $CADEF                          ; Conditional assembly switches
                     0000    100          $CLSMSGDEF                      ; Cluster message offsets
                     0000    101          $CLUBDEF                        ; CLUB offsets
                     0000    102          $DYNDEF                         ; Structure type code definitions
                     0000    103          $LCKDEF                         ; LCK definitions
                     0000    104          $LKBDEF                         ; LKB offsets
                     0000    105          $PCBDEF                         ; PCB offsets
                     0000    106          $RSBDEF                         ; RSB offsets
                     0000    107          $SSDEF                          ; System status code definitions
                     0000    108
                     0000    109 ;
                     0000    110 ; MACROS:
                     0000    111 ;
                     0000    112
                     0000    113 ;
                     0000    114 ; EQUATED SYMBOLS:
                     0000    115 ;
                     0000    116
00000018             0000    117 LOCKFRAME = 24                           ; Number of bytes pushed onto
                     0000    118                                          ; stack for each recursive call
                     0000    119                                          ; of LCK$SRCH_RESDLCK  (5 registers
                     0000    120                                          ; plus return address).  This
                     0000    121                                          ; cannot be charges without making
                     0000    122                                          ; corresponding coding changes
                     0000    123
                     0000    124 ;
                     0000    125 ; OWN STORAGE:
                     0000    126 ;
                     0000    127
                00000000     128          .PSECT  LOCKMGR
```

```
                     0000   130          .SBTTL  LCK$SEARCHDLCK - Search and break deadlocks
                     0000   131
                     0000   132 ;++
                     0000   133 ; FUNCTIONAL DESCRIPTION:
                     0000   134 ;
                     0000   135 ;       This routine is the top level routine for identifying and resolving
                     0000   136 ;       deadlocks.  Identifying deadlocks is performed by two separate
                     0000   137 ;       routines.  One identifies conversion deadlocks (is only called
                     0000   138 ;       if this request is a conversion) and the other identifies multiple
                     0000   139 ;       resource deadlocks.  When a deadlock is found, one of the locks
                     0000   140 ;       forming the deadlock is selected as the "victim".  This lock
                     0000   141 ;       receives the status SS$_DEADLOCK in its LKSB and the lock request
                     0000   142 ;       is denied.  Multiple deadlocks are handled in the following way.
                     0000   143 ;       This routine quits after it finds and breaks one deadlock.  However,
                     0000   144 ;       in this case, if the original lock (R6) is not the victim, then it is
                     0000   145 ;       not removed from the lock timeout queue.  The next time that the
                     0000   146 ;       timeout queue is examined this lock will again be searched for
                     0000   147 ;       deadlock.  This is repeated until either no deadlock is found for this
                     0000   148 ;       lock or it is taken off the timer queue for another reason (for
                     0000   149 ;       example, it gets granted).
                     0000   150 ;
                     0000   151 ;       This routine also must handle several instances where structures
                     0000   152 ;       having nothing to do with deadlock searching have been placed
                     0000   153 ;       on the time out queue.  These structures represent instances
                     0000   154 ;       in the distributed lock manager where a message needed to be
                     0000   155 ;       sent but pool could not be allocated.  Since, in general, the
                     0000   156 ;       structures could not accomodate a fork block, they are instead
                     0000   157 ;       inserting on the time out queue where here we resume the thread
                     0000   158 ;       of execution.
                     0000   159 ;
                     0000   160 ; CALLING SEQUENCE:
                     0000   161 ;
                     0000   162 ;       BSBW    LCK$SEARCHDLCK
                     0000   163 ;
                     0000   164 ; INPUT PARAMETERS:
                     0000   165 ;
                     0000   166 ;       R6      Address of LKB to determine if in deadlock cycle
                     0000   167 ;               This should either be a local or master copy lock.
                     0000   168 ;               This may also be a RSB waiting for pool to send a message.
                     0000   169 ;
                     0000   170 ; OUTPUT PARAMETERS:
                     0000   171 ;
                     0000   172 ;       None
                     0000   173 ;
                     0000   174 ; SIDE EFFECTS:
                     0000   175 ;
                     0000   176 ;       R0 - R4 are destroyed
                     0000   177 ;--
                     0000   178
                     0000   179 LCK$SEARCHDLCK::
                0FE0 8F   BB 0000   180          PUSHR   #^M<R5,R6,R7,R8,R9,R10,R11>
00000000'          GF   95 0004   181          TSTB    G^LCK$GB_STALLREQS      ; Don't start a search if we are
                     34   19 000A   182          BLSS    8$                      ; in the middle of a state change
                          000C   183
                          000C   184 5$:       ; Handle structures that need to resume threads waiting for pool.
                          000C   185
                     57   56 D0   000C   186          MOVL    R6,R7                   ; Save address of structure
```

```
                OA A6   91   000F   187          CMPB    LKB$B_TYPE(R6),-        ; Is this a RSB?
                    36        0012   188                  #DYN$C_RSB
                    0B   12   0013   189          BNEQ    6$                      ; No
                58  66   0F   0015   190          REMQUE  (R6),R8                 ; Yes remove it from the timeout queue
                                0018   191                                          ; and put RSB address in R8
          00000000'GF   16   0018   192          JSB     G^LCK$SND_RMVDIR        ; Send a remove dir. entry message
                    14   11   001E   193          BRB     7$
                36 A6   95   0020   194  6$:     TSTB    LKB$B_STATE(R6)         ; Is the lock granted?
                    22   15   0023   195          BLEQ    10$                     ; No
                    04   E1   0025   196          BBC     #LKB$V_MSTCPY,-         ; Yes, lock must be a master copy
                19 2A A6         0027   197                  LKB$W_STATUS(R6),9$
             58 50 A6   D0   002A   198          MOVL    LKB$L_RSB(R6),R8        ; Get RSB address
          00000000'GF   16   002E   199          JSB     G^LCK$SND_GRANTED       ; Send a lock granted message
       57 00000000'GF   D1   0034   200  7$:     CMPL    G^LCK$GL_TIMOUTQ,R7     ; Is the same structure back on the queue?
                    03   13   003B   201          BEQL    8$                      ; Yes, exit
                  0098   31   003D   202          BRW     60$                     ; No, try next structure
                  00B1   31   0040   203  8$:     BRW     LCK$DLCKEXIT
                                0043   204
                                0043   205  9$:     BUG_CHECK       LOCKMGRERR,FATAL; Granted lock is not master copy
                                0047   206
                                0047   207  10$:    ; Have a master or local copy lock.  The lock is still on the
                                0047   208          ; timeout queue.
                                0047   209
                    00000002    0047   210          .IF NE  CA$_MEASURE
          00000000'EF   D6   0047   211          INCL    L^PMS$GL_DLCKSRCH
                                004D   212          .ENDC
                                004D   213
                                004D   214          ASSUME  LKB$K_GRANTED  EQ  1
                                004D   215          ASSUME  LKB$K_CONVERT  EQ  0
                                004D   216          ASSUME  LKB$K_WAITING  EQ  -1
                                004D   217
                    59   D4   004D   218          CLRL    R9                      ; Indicate no timestamp assigned
                36 A6   95   004F   219          TSTB    LKB$B_STATE(R6)         ; Is this lock on the conversion queue?
                    09   12   0052   220          BNEQ    11$                     ; No, must be on wait queue
                  00A2   30   0054   221          BSBW    SEARCH_CVTDLCK          ; Yes, search for conversion deadlocks
                    50   D5   0057   222          TSTL    R0                      ; Was a deadlock found?
                    70   19   0059   223          BLSS    50$                     ; Yes, and we must exit for now
                    7B   14   005B   224          BGTR    60$                     ; Yes, but we can search again
                                005D   225
                                005D   226  11$:    ; We didn't have a conversion deadlock so now we have to search
                                005D   227          ; for multiple resource deadlocks.  Set up registers and determine
                                005D   228          ; if bitmap is available for use.  Note that normally references
                                005D   229          ; to EXE$GQ_SYSTIME should be at IPL$_HWCLK.  However, we can tolerate
                                005D   230          ; the race condition of referencing it at IPL$_SYNCH here.  The
                                005D   231          ; result would be to incorrectly conclude that the bitmap is in use
                                005D   232          ; which would cause us to retry later.
                                005D   233
                54  0C A6   3C   005D   234          MOVZWL  LKB$L_PID(R6),R4        ; Get process index
                    13   13   0061   235          BEQL    12$                     ; Must be a master copy
          54 00000000'FF44   D0   0063   236          MOVL    @L^SCH$GL_PCBVEC[R4],R4 ; Convert to PCB address
                58  64 A4   D0   006B   237          MOVL    PCB$L_EPID(R4),R8       ; Get EPID
                54  0104 C4   DE   006F   238          MOVAL   PCB$L_LOCKQFL(R4),R4    ; Make R4 point to lock queue in PCB
                    04   11   0074   239          BRB     14$
                58  14 A6   D0   0076   240  12$:    MOVL    LKB$L_EPID(R6),R8       ; Get EPID
             57 00000000'GF   D0   007A   241  14$:    MOVL    G^LCK$GL_PRCMAP,R7      ; Get address of process bitmap
                    5A   5E   D0   0081   242          MOVL    SP,R10                  ; Save current stack pointer
                00000000'GF   C1   0084   243          ADDL3   G^LCK$GL_EXTRASTK,-     ; Compute upper bound for stack
```

```
                5B   00000000'GF        008A   244                  G^EXE$GL_INTSTKLM,R11   ; (allow LCK$GL_EXTRASTK plus one
                           5B   18   C0  0090   245          ADDL    #LOCKFRAME,R11          ; lock frame)
                                        0093   246
                50   00000000'GF   7E   0093   247          MOVAQ   G^LCK$GQ_BITMAP_EXP,R0  ; Get address of bitmap expiration
           00000004'GF    0C   A0   D1   009A   248          CMPL    12(R0),G^EXE$GQ_SYSTIME+4; Compare with local expiration time
                                        00A2   249
                               14   1F   00A2   250          BLSSU   20$                     ; Bitmap is available
                               0A   1A   00A4   251          BGTRU   15$                     ; Bitmap may be in use
           00000000'GF    08   A0   D1   00A6   252          CMPL    8(R0),G^EXE$GQ_SYSTIME  ; Compare low order parts
                               08   1B   00AE   253          BLEQU   20$                     ; Bitmap is available
                                        00B0   254
                                        00B0   255 15$:      ; Bitmap may be in use;  need to send a message to get a timestamp.
                                        00B0   256           ; Note that if we really send a message that we won't return here
                                        00B0   257           ; but will exit deadlock detection for now.
                                        00B0   258
                     00000000'GF   16   00B0   259          JSB     G^LCK$SND_TIMESTAMP_RQST
                               0E   11   00B6   260          BRB     40$                     ; In case we do return with a timestamp
                                        00B8   261
                               60   7C   00B8   262 20$:     CLRQ    (R0)                    ; Indicate bitmap has been reused
                               54   DD   00BA   263          PUSHL   R4
     67    F8 A7    00    67    00   2C   00BC   264          MOVC5   #0,(R7),#0,-8(R7),(R7)  ; and clear it
                               54 8ED0   00C3   265          POPL    R4
                                        00C6   266
                                        00C6   267 40$:      ; Register usage at this point:
                                        00C6   268           ;
                                        00C6   269           ;       R4      Address of PCB+PCB$L_LOCKQFL (except master copies)
                                        00C6   270           ;       R6      Address of original LKB to perform search for
                                        00C6   271           ;       R7      Address of process bitmap
                                        00C6   272           ;       R8      EPID of process we are doing search for
                                        00C6   273           ;       R9      Indicates if we have a timestamp
                                        00C6   274           ;       R10     Current stack pointer
                                        00C6   275           ;       R11     Top of useable stack (there is some extra space)
                                        00C6   276
                     0099   30   00C6   277          BSBW    LCK$SRCH_RESDLCK        ; Search for multiple resource deadlocks
                       50   D5   00C9   278          TSTL    R0                      ; Was a deadlock found?
                       27   19   00CB   279 50$:     BLSS    LCK$DLCKEXIT            ; Yes, and we must exit for now
                       09   14   00CD   280          BGTR    60$                     ; Yes, but we can search again
                                        00CF   281
                                        00CF   282           ; No deadlock was found.  Remove this lock from the timeout queue.
                                        00CF   283
                     50   66   0F   00CF   284          REMQUE  LKB$L_ASTQFL(R6),R0     ; Remove from queue
                     0040 8F   AA   00D2   285          BICW    #LKB$M_TIMOUTQ,-        ; Clear status bit indicating
                       2A A6        00D6   286                  LKB$W_STATUS(R6)        ; lock was on timeout queue
                                        00D8   287
                                        00D8   288 60$:      ; See if we need to do another search (the same lock may still
                                        00D8   289           ; be at the head of the timeout queue or another lock may have
                                        00D8   290           ; also timed out).  We do this here instead of in TIMESCHDL because
                                        00D8   291           ; there are other exits from this routine that leave a timed out
                                        00D8   292           ; lock at the head of the queue so that a search can be restarted
                                        00D8   293           ; a second from now.
                                        00D8   294
          55   00000000'EF   DE   00D8   295          MOVAL   L^LCK$GL_TIMOUTQ,R5     ; Get address of list head
                       56   65   D0   00DF   296          MOVL    (R5),R6                 ; Get first entry on list
                       56   55   D1   00E2   297          CMPL    R5,R6                   ; Is list empty?
                            0D   13   00E5   298          BEQL    LCK$DLCKEXIT            ; Yes
                         18 A6   D1   00E7   299          CMPL    LKB$L_DUETIME(R6),-     ; No, has this one timed out?
                     00000000'EF        00EA   300                  L^EXE$GL_ABSTIM
```

```
                03   1A  00EF   301          BGTRU   LCK$DLCKEXIT            ; No, exit
              FF18   31  00F1   302          BRW     5$                      ; Yes, do another deadlock search
                         00F4   303
                         00F4   304 LCK$DLCKEXIT::
          0FE0 8F    BA  00F4   305          POPR    #^M<R5,R6,R7,R8,R9,R10,R11>
                     05  00F8   306          RSB
                         00F9   307
```

```
                     00F9   309          .SBTTL  SEARCH_CVTDLCK - Search for conversion deadlocks
                     00F9   310
                     00F9   311 ;++
                     00F9   312 ; FUNCTIONAL DESCRIPTION:
                     00F9   313 ;
                     00F9   314 ;       This routine searches for conversion deadlocks and selects a victim
                     00F9   315 ;       if one is found.  A conversion deadlock is one in which a conversion
                     00F9   316 ;       request has a granted mode incompatible with the requested mode
                     00F9   317 ;       of another conversion request ahead of it in the conversion
                     00F9   318 ;       queue.  For exampie, assume there are two PR
                     00F9   319 ;       locks on a resource.  One PR lock tries to convert to EX and
                     00F9   320 ;       therefore must wait.  Then the second PR lock tries to convert to
                     00F9   321 ;       EX and it too must wait.  However, the first will never get granted
                     00F9   322 ;       since its requested mode (EX) is incompatible with the second's
                     00F9   323 ;       granted mode (PR).  The second will never get granted since
                     00F9   324 ;       it's waiting behind the first.
                     00F9   325 ;       To find conversion deadlocks it is sufficient to check all locks
                     00F9   326 ;       ahead of this lock on the conversion queue to see if their
                     00F9   327 ;       requested modes are incompatible with this lock's granted mode.
                     00F9   328 ;
                     00F9   329 ; CALLING SEQUENCE:
                     00F9   330 ;
                     00F9   331 ;       BSBW    SEARCH_CVTDLCK
                     00F9   332 ;
                     00F9   333 ; INPUT PARAMETERS:
                     00F9   334 ;
                     00F9   335 ;       R6      Address of LKB to search for conversion deadlocks
                     00F9   336 ;       R9      Contains 0 indicating no message buffer
                     00F9   337 ;
                     00F9   338 ; OUTPUT PARAMETERS:
                     00F9   339 ;
                     00F9   340 ;       R0      Completion code:
                     00F9   341 ;                       0  = No deadlock found
                     00F9   342 ;                       1  = Deadlock found and another search may be performed
                     00F9   343 ;                       -1 = Deadlock may or may not have been found but don't
                     00F9   344 ;                            perform another search immediately.  Typical
                     00F9   345 ;                            reasons are master copy was on this system
                     00F9   346 ;                            so another deadlock search cannot be repeated
                     00F9   347 ;                            immediately (or we will find the same one again)
                     00F9   348 ;                            or we needed to allocate a CDRP but failed to
                     00F9   349 ;                            allocate pool.
                     00F9   350 ;
                     00F9   351 ; SIDE EFFECTS:
                     00F9   352 ;
                     00F9   353 ;       R0 - R2 and R5 are destroyed if a deadlock is not found
                     00F9   354 ;       R0 - R8 are destroyed if a deadlock is found
                     00F9   355 ;--
                     00F9   356
                     00F9   357 SEARCH_CVTDLCK:
          50 A6      C1  00F9   358          ADDL3   LKB$L_RSB(R6),-         ; Point to head of conversion queue
       55    18          00FC   359                  #RSB$L_CVTQFL,R5
    52    35 A6      9A  00FE   360          MOVZBL  LKB$B_GRMODE(R6),R2     ; Get granted mode of current lock
       51    56      D0  0102   361          MOVL    R6,R1                   ; Address of current lock
    51    3C A1      D0  0105   362 10$:     MOVL    LKB$L_SQBL(R1),R1       ; Get previous lock in queue
       55    51      D1  0109   363          CMPL    R1,R5                   ; Reached the queue head yet?
             4D      13  010C   364          BEQL    80$                     ; Yes
       51    38      C2  010E   365          SUBL    #LKB$L_SQFL,R1          ; Back up to start of LKB
```

```
                50   34 A1   9A  0111    366          MOVZBL  LKB$B_RQMODE(R1),R0      ; Get requested mode
E9 0000'CF42    50   E0  0115    367          BBS     R0,W^[CK$COMPAT_TBL[R2],10$ ; Branch if compatible
                             011C    368
                             011C    369          ; Have a conversion deadlock.  The victim is the one with the lower
                             011C    370          ; deadlock priority.  R1 and R6 contain the two LKB addresses.
                             011C    371          ; Either one of these locks could be a master copy;  get the two
                             011C    372          ; deadlock priorities out of either the PCB of the LKB.
                             011C    373
 55   00000000'EF    D0  011C    374          MOVL    L^SCH$GL_PCBVEC,R5      ; Get address of PCB vector
       50    0C A1    3C  0123    375          MOVZWL  LKB$L_PID(R1),R0        ; Get process index
                0B    13  0127    376          BEQL    20$                     ; Master copy
          50   6540    D0  0129    377          MOVL    (R5)[R0],R0             ; Get PCB address
      52    010C C0    D0  012D    378          MOVL    PCB$L_DLCKPRI(R0),R2    ; R2 has pri. for lock in R1
                04    11  0132    379          BRB     30$
       52    24 A1    D0  0134    380 20$:     MOVL    LKB$L_DLCKPRI(R1),R2    ; R2 has pri. for lock in R1
       50    0C A6    3C  0138    381 30$:     MOVZWL  LKB$L_PID(R6),R0        ; Get process index
                0B    13  013C    382          BEQL    40$                     ; Master copy
          50   6540    D0  013E    383          MOVL    (R5)[R0],R0             ; Get PCB address
      53    010C C0    D0  0142    384          MOVL    PCB$L_DLCKPRI(R0),R3    ; R3 has pri. for lock in R6
                04    11  0147    385          BRB     50$
       53    24 A6    D0  0149    386 40$:     MOVL    LKB$L_DLCKPRI(R6),R3    ; R3 has pri. for lock in R6
                             014D    387
           53   52    D1  014D    388 50$:     CMPL    R2,R3                   ; Compare the deadlock priorities
                03    1E  0150    389          BGEQU   60$                     ; Branch if orig. lock is victim
           56   51    D0  0152    390          MOVL    R1,R6                   ; Other lock is victim
                53    D4  0155    391 60$:     CLRL    R3                      ; Indicates R6 has LKB address
              01F5    30  0157    392          BSBW    LCK$BREAK_DEADLOCK      ; Break deadlock; returns status in R0
                05    015A    393          RSB
                             015B    394
                50    D4  015B    395 80$:     CLRL    R0                      ; No deadlock found
                05    015D    396          RSB
```

```
                        .SBTTL  LCK$SRCH_RESDLCK - Search for resource deadlocks

;++
; FUNCTIONAL DESCRIPTION:
;
;       This routine searches for multiple resource deadlocks and selects
;       a victim if one is found.  A multiple resource deadlock is one
;       in which a circular list of processes are each waiting for one
;       another on two or more resources.  For example, assume process A
;       locks resource 1, process B locks resource 2, then process A
;       locks resource 2 (and waits), and finally process B locks
;       resource 1 (and waits).  A and B are each waiting for the other on
;       different resources.  This type of deadlock must involve two or
;       more resources unless one process locks the same resource twice.
;       (Normally, it is senseless for one process to lock the
;       same resource twice but this does make sense if the process
;       is multi-threaded).
;       To find multiple resource deadlocks a recursive algorithm is used.
;       The basis of this algorithm is for each process with a lock on
;       the current resource blocking the current lock, find any waiting
;       locks that process has and recursively see what processes are
;       blocking those locks.  As we do this, see if we can find a path
;       back to the current process.  In other words, we are travelling
;       a graph of waiting processes searching for a path back to our
;       starting point.  If we find one, then the stack consists of a
;       list of waiting processes and locks forming a deadlock.  The lock
;       with the minimum deadlock priority is selected as a victim
;       and we return.  Multiple deadlocks are handled by calling this
;       routine again.
;       To prevent this algorithm from looping on a deadlock cycle that
;       doesn't include the original process (R8), a bitmap representing
;       each process in the system is used.  Whenever a particular
;       process is visited, the corresponding bit is set.  If the bit
;       is already set, then we won't visit that process after all.  Note
;       that when we leave a process, the corresponding bit is NOT cleared.
;       The result of this is that deadlock cycles not involving the original
;       process are not found (yet).  Instead, they are ignored by this
;       deadlock search, but will be found later when a lock in that
;       cycle times out.  The reason for not clearing the bitmap is that
;       this dramatically improves the worst-case  behavior of the
;       algorithm by not visiting a process if it has been visited before.
;
; CALLING SEQUENCE:
;
;       BSBW    LCK$SRCH_RESDLCK
;
; INPUT PARAMETERS:
;
;       R4      Address of PCB + PCB$L_LOCKQFL (to determine who is blocking)
;               (only if R6 is not a master copy)
;       R6      Address of LKB (to determine who is blocking)
;       R7      Address of process bitmap (one bit for each process in system)
;       R8      EPID of process that initiated search (our starting point)
;       R9      Address of input message or zero
;       R10     Bottom of deadlock stack
;       R11     Top of useable stack (so that we don't overflow the stack)
;
```

```
                     455 ; OUTPUT PARAMETERS:
                     456 ;
                     457 ;       R0      Completion code:
                     458 ;                       0  = No deadlock found
                     459 ;                       1  = Deadlock found (normal)
                     460 ;                       -1 = Deadlock found; master copy was on this system
                     461 ;                            so another deadlock search cannot be repeated
                     462 ;                            immediately (or we will find the same one again)
                     463 ;
                     464 ;
                     465 ; SIDE EFFECTS:
                     466 ;
                     467 ;       R1 is destroyed if a deadlock is not found
                     468 ;       R0 - R8 are destroyed if a deadlock is found
                     469 ;--
                     470
                     471 ; Note:  The following are the register conventions used by this routine.
                     472 ;       R0 and R1 may be used as scratch registers.  Each time this
                     473 ;       routine is called (recursively) R2 - R6 are saved on the stack.
                     474 ;       R7 - R11  remain constant during the recursive calls.  Registers
                     475 ;       are used as follows:
                     476 ;
                     477 ;               R2      Maximum lock mode computed so far
                     478 ;               R3      Address of queue header in RSB
                     479 ;               R4      Address of PCB + PCB$L_LOCKQFL (address of queue header)
                     480 ;               R5      Address of LKB blocking LKB in R6
                     481 ;               R6      Address of LKB to determine who is blocking
                     482 ;               R7      Address of process bitmap
                     483 ;               R8      Ultimate EPID we are searching for
                     484 ;               R9      Address of input message or zero
                     485 ;               R10     Bottom of deadlock stack
                     486 ;               R11     Top of useable stack
                     487 ;
                     488 ;       Note that there are several assumptions made in the code about
                     489 ;       what registers are used for what and the order in which they
                     490 ;       are saved on the stack.  Specifically, the loop that selects
                     491 ;       the deadlock victim assumes both the number of resisters saved
                     492 ;       and their relative positions on the stack.  See also the
                     493 ;       definition of the symbol LOCKFRAME at the beginning of this module.
                     494
                     495 STATE_ERROR:
                     496         BUG_CHECK       LOCKMGRERR,FATAL
                     497
                     498 LCK$SRCH_RESDLCK::
 007C 8F    BB       499         PUSHR   #^M<R2,R3,R4,R5,R6>     ; Can't change this without also
                     500                                         ; changing value of LOCKFRAME and
                     501                                         ; deadlock resolution code
                     502
                     503         ; First run through all locks waiting ahead of this lock
                     504         ; maximizing the requested modes and checking all locks
                     505         ; incompatible with the current "maxmode".  If this lock is
                     506         ; on the wait queue then we do the wait queue first and
                     507         ; the converison queue next.  If this lock is on the
                     508         ; conversion queue then we do only the conversion queue.
                     509         ; Later we'll do all the granted locks.
                     510
                     511         ASSUME  RSB$L_CVTQFL  EQ  RSB$L_GRQFL+8
```

```
                     0166   512          ASSUME  RSB$L_WTQFL  EQ  RSB$L_CVTQFL+8
                     0166   513
52   34 A6   9A      0166   514          MOVZBL  LKB$B_RQMODE(R6),R2     ; R2 = this lock's requested mode
      18 C1          016A   515          ADDL3   #RSB$C_CVTQFL,-        ; R3 = Addr. of cvt. queue header
53   50 A6          016C   516                  LKB$L_RSB(R6),R3
     20 A3   D5      016F   517          TSTL    RSB$L_CSID-RSB$L_CVTQFL(R3) ; Verify resource is mastered here
        EA   12      0172   518          BNEQ    STATE_ERROR
                     0174   519          DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
                     0174   520                  <-
                     0174   521                  <CONVERT,10$>,-
                     0174   522                  <WAITING,5$>-
                     0174   523                  >
        DE   11      017E   524          BRB     STATE_ERROR            ; Shouldn't have locks in other states
   53   08 C0        0180   525  5$:     ADDL    #8,R3                  ; Wait queue - point to wait queue hdr
   55   56 D0        0183   526  10$:    MOVL    R6,R5                  ; R5 will point to other LKB's
                     0186   527                                         ; in front of the one pointed to by R6
                     0186   528
55   3C A5   D0      0186   529  20$:    MOVL    LKB$L_SQBL(R5),R5      ; Get previous lock on state queue
   53   55 D1        018A   530          CMPL    R5,R3                  ; Reached head of queue yet?
        03 12        018D   531          BNEQ    15$                    ; No
      00BA 31        018F   532          BRW     50$                    ; Yes
   55   38 C2        0192   533  15$:    SUBL    #LKB$L_SQFL,R5         ; Back up to point to start of LKB
     36 A5 91        0195   534          CMPB    LKB$B_STATE(R5),-      ; Is lock in an SCS state?
     FF 8F           0198   535                  #LKB$K_WAITING
        EA 19        019A   536          BLSS    20$                    ; Yes, ignore
50   34 A5   9A      019C   537          MOVZBL  LKB$B_RQMODE(R5),R0    ; R0 = requested mode
   51   52 D0        01A0   538          MOVL    R2,R1                  ; Save old maxmode
                     01A3   539
                     01A3   540          ; Maximize lock modes (in R0 and R2) and see if this lock (R5) is
                     01A3   541          ; incompatible with (the previous) maxmode.  The maximization function
                     01A3   542          ; is a simple arithmetic maximum except if the two modes are CW and PR.
                     01A3   543          ; In that case the maximum of CW and PR is PW.  PW is incompatible
                     01A3   544          ; with everything either CW or PR is incompatible with.
                     01A3   545
     52   50 91      01A3   546          CMPB    R0,R2                  ; Current mode greater than maxmode?
        20 13        01A6   547          BEQL    25$                    ; No, they're equal
        0C 1A        01A8   548          BGTRU   21$                    ; Yes, compute new maxmode
     02   50 91      01AA   549          CMPB    R0,#LCK$K_CWMODE       ; No, is current mode CW?
        19 12        01AD   550          BNEQ    25$                    ; No, maxmode = R2
     03   52 91      01AF   551          CMPB    R2,#LCK$K_PRMODE       ; Yes, is maxmode PR?
        14 12        01B2   552          BNEQ    25$                    ; No, maxmode = R2
        0A 11        01B4   553          BRB     22$                    ; Yes, new maxmode is PW
     02   52 91      01B6   554  21$:    CMPB    R2,#LCK$K_CWMODE       ; Is maxmode CW?
        0A 12        01B9   555          BNEQ    23$                    ; No, maxmode = R0
     03   50 91      01BB   556          CMPB    R0,#LCK$K_PRMODE       ; Yes, is current mode PR?
        05 12        01BE   557          BNEQ    23$                    ; No, maxmode = R0
     52   04 90      01C0   558  22$:    MOVB    #LCK$K_PWMODE,R2       ; Have CW and PR; maxmode = PW
        03 11        01C3   559          BRB     25$
     52   50 90      01C5   560  23$:    MOVB    R0,R2                  ; Maxmode = R0
                     01C8   561
0000'CF41   50 E0    01C8   562  25$:    BBS     R0,-                   ; Branch if compatible with
            B7       01CE   563                  W^LCK$COMPAT_TBL[R1],20$; saved maxmode
        0A E0        01CF   564          BBS     #LCK$V_NODLCKBLK,-     ; Branch if this lock should not be
      B2 28 A5       01D1   565                  LKB$W_FLAGS(R5),20$    ; considered as blocking other locks
                     01D4   566
                     01D4   567          ; Have a lock incompatible with maxmode.  First see
                     01D4   568          ; if the process owning the lock (in R5) is the process we
```

```
                                01D4   569          ; started with (in R8).  If it is, then we have deadlock.  If not,
                                01D4   570          ; then see if the process has any other waiting locks.  If it
                                01D4   571          ; does then we have to recurse down a level.  If it doesn't then
                                01D4   572          ; we can continue at this level.
                                01D4   573
                          04    E1    01D4   574          BBC     #LKB$V_MSTCPY,-          ; Branch if not master copy lock
                    11 2A A5          01D6   575                  LKB$W_STATUS(R5),28$
                 58   14 A5    D1    01D9   576          CMPL    LKB$L_EPID(R5),R8        ; Have a master copy; deadlock found?
                          6A    13    01DD   577          BEQL    45$                      ; Yes
                       56  55    D0    01DF   578          MOVL    R5,R6                    ; No
                00000000'GF    16    01E2   579          JSB     G^LCK$SND_SRCHDLCK       ; Send a message to keep looking
                          9C    11    01E8   580          BRB     20$                      ; Continue on this RSB
                   50   0C A5    3C    01EA   581  28$:     MOVZWL  LKB$L_PID(R5),R0         ; Get process index
              54   00000000'FF40    D0    01EE   582          MOVL    @L^SCH$GL_PCBVEC[R0],R4  ; Convert to PCB address
                 58   64 A4    D1    01F6   583          CMPL    PCB$L_EPID(R4),R8        ; Is this the original process?
                          4D    13    01FA   584          BEQL    45$                      ; Yes, have a deadlock
                    86 67   50    E2    01FC   585          BBSS    R0,(R7),20$              ; Br. if we've already done this process
                 54   0104 C4    DE    0200   586          MOVAL   PCB$L_LOCKQFL(R4),R4     ; Point to lock queue header
                   56   04 A4    D0    0205   587          MOVL    4(R4),R6                 ; Get last lock in list
                   56   C0 A6    DE    0209   588  30$:     MOVAL   -LKB$L_OWNQFL(R6),R6     ; Point to start of LKB
                      56  55    D1    020D   589          CMPL    R5,R6                    ; Is this the one we have in R5?
                          2B    13    0210   590          BEQL    35$                      ; Yes, move on to next one
                                0212   591          DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
                                0212   592                  <-
                                0212   593                  <CONVERT,32$>,-
                                0212   594                  <WAITING,32$>-
                                0212   595                  >
                       FF67     31    021C   596          BRW     20$
                          09    E0    021F   597  32$:     BBS     #LCK$V_NODLCKWT,-        ; Branch if this lock should not be
                    19 28 A6          0221   598                  LKB$W_FLAGS(R6),35$      ; considered as waiting for other locks
                 50   50 A6    D0    0224   599          MOVL    LKB$L_RSB(R6),R0         ; Get RSB for this lock
                   38 A0    D5    0228   600          TSTL    RSB$L_CSID(R0)           ; Is it managed elsewhere?
                          08    13    022B   601          BEQL    34$                      ; No, recurse here
                00000000'GF    16    022D   602          JSB     G^LCK$SND_SRCHDLCK       ; Yes, send a message to keep searching
                          08    11    0233   603          BRB     35$                      ; Continue with this PCB
                      5B  5E    D1    0235   604  34$:     CMPL    SP,R11                   ; Is there enough stack to recurse?
                          0F    1F    0238   605          BLSSU   45$                      ; No, have to assume deadlock
                       FF25     30    023A   606          BSBW    LCK$SRCH_RESDLCK         ; Yes, recursively search
                 56   44 A6    D0    023D   607  35$:     MOVL    LKB$L_OWNQBL(R6),R6      ; Get previous lock
                      54  56    D1    0241   608          CMPL    R6,R4                    ; Reached end of list?
                          C3    12    0244   609          BNEQ    30$                      ; No, get next lock in PCB (inner loop)
                       FF3D     31    0246   610  40$:     BRW     20$                      ; Yes, get next lock in RSB (outer loop)
                                0249   611
                       00A3     31    0249   612  45$:     BRW     DEADLOCK_FOUND
                                024C   613
                                024C   614  50$:     ; Reached the queue header.  Back up R3 to point to the previous
                                024C   615          ; queue header in the RSB.  If R3 is pointing to the granted
                                024C   616          ; queue, then we are done with this loop and we continue with
                                024C   617          ; the granted queue.  Otherwise, we repeat this loop for the
                                024C   618          ; conversion queue.
                                024C   619
                      53  08    C2    024C   620          SUBL    #8,R3                    ; Back up R3 one queue header
                55   C8 A3    9E    024F   621          MOVAB   -LKB$L_SQFL(R3),R5       ; Prepare to process that queue
                56   10 AE    D0    0253   622          MOVL    16(SP),R6                ; Restore R6
                          10    C1    0257   623          ADDL3   #RSB$L_GRQFL,-           ; R0 = address of granted queue
                50   50 A6          0259   624                  LKB$L_RSB(R6),R0
                      50  53    D1    025C   625          CMPL    R3,R0                    ; Have we reached the granted queue?
```

```
                E5   12  025F   626          BNEQ    40$                     ; No, repeat for conversion queue
                             0261   627
                             0261   628      ; Now repeat a similar procedure for all locks on the granted
                             0261   629      ; queue whose granted mode is incompatible with the maxmode
                             0261   630      ; in R2.
                             0261   631
          55    38 A5   D0   0261   632 60$: MOVL    LKB$L_SQFL(R5),R5       ; Get next lock in granted queue
            53    55   D1   0265   633      CMPL    R5,R3                   ; Reached end of queue?
                  7E   13   0268   634      BEQL    90$                     ; Yes, no deadlock
               55    38   C2   026A   635      SUBL    #LKB$L_SQFL,R5          ; Back up to point to start of LKB
            50    35 A5   9A   026D   636      MOVZBL  LKB$B_GRMODE(R5),R0     ; Get granted mode
E9 0000'CF42    50   E0   0271   637      BBS     R0,W^LCK$COMPAT_TBL[R2],60$ ; Branch if compatible
                  0A   E0   0278   638      BBS     #LCK$V_NODLCKBLR,-      ; Branch if this lock should not be
                E4 28 A5        027A   639              LKB$W_FLAGS(R5),60$     ; considered as blocking other locks
                             027D   640
                             027D   641      ; Have an incompatible lock on the granted queue.  First see
                             027D   642      ; if the process owning the lock (in R5) is the process we
                             027D   643      ; started with (in R8).  If it is, then we have deadlock.  If not,
                             027D   644      ; then see if the process has any waiting locks.  If it
                             027D   645      ; does then we have to recurse down a level.  If it doesn't then
                             027D   646      ; we can continue at this level.
                             027D   647
                  04   E1   027D   648      BBC     #LKB$V_MSTCPY,-         ; Branch if not master copy lock
              11 2A A5        027F   649              LKB$W_STATUS(R5),63$
            58    14 A5   D1   0282   650      CMPL    LKB$L_EPID(R5),R8       ; Have a master copy; deadlock found?
                  67   13   0286   651      BEQL    DEADLOCK_FOUND          ; Yes
                56    55   D0   0288   652      MOVL    R5,R6                   ; No
          00000000'GF    16   028B   653      JSB     G^LCK$SND_SRCHDLCK      ; Send a message to keep looking
                  CE   11   0291   654      BRB     60$                     ; Continue on this RSB
             50    0C A5   3C   0293   655 63$: MOVZWL  LKB$L_PID(R5),R0        ; Get process index
                  C8   13   0297   656      BEQL    60$                     ; Ignore system owned locks
54    00000000'FF40   D0   0299   657      MOVL    @L^SCH$GL_PCBVEC[R0],R4 ; Convert to PCB address
            58    64 A4   D1   02A1   658      CMPL    PCB$L_EPID(R4),R8       ; Is this the original process?
                  48   13   02A5   659 65$: BEQL    DEADLOCK_FOUND          ; Yes, have a deadlock
             B6 67    50   E2   02A7   660      BBSS    R0,(R7),60$             ; Br. if we've already done this process
          54    0104 C4   DE   02AB   661      MOVAL   PCB$L_LOCKQFL(R4),R4    ; Point to lock queue header
            56    04 A4   D0   02B0   662      MOVL    4(R4),R6                ; Get last lock in list
            56    C0 A6   DE   02B4   663 70$: MOVAL   -LKB$L_OWNQFL(R6),R6    ; Back up to start of LKB
                             02B8   664      DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
                             02B8   665              <-
                             02B8   666              <CONVERT,71$>,-
                             02B8   667              <WAITING,71$>-
                             02B8   668              >
                  9D   11   02C2   669      BRB     60$                     ; Done with this PCB
                  09   E0   02C4   670 71$: BBS     #LCK$V_NODLCKWT,-       ; Branch if this lock should not be
              19 28 A6        02C6   671              LKB$W_FLAGS(R6),75$     ; considered as waiting for other locks
            50    50 A6   D0   02C9   672      MOVL    LKB$L_RSB(R6),R0        ; Get RSB for this lock
                  38 A0   D5   02CD   673      TSTL    RSB$L_CSID(R0)          ; Is it managed elsewhere?
                  08   13   02D0   674      BEQL    72$                     ; No, recurse here
          00000000'GF    16   02D2   675      JSB     G^LCK$SND_SRCHDLCK      ; Yes, send a message to keep searching
                  08   11   02D8   676      BRB     75$                     ; Continue with this PCB
               5B    5E   D1   02DA   677 72$: CMPL    SP,R11                  ; Is there enough stack to recurse?
                  10   1F   02DD   678 73$: BLSSU   DEADLOCK_FOUND          ; No, have to assume deadlock
                FE80   30   02DF   679      BSBW    LCK$SRCH_RESDLCK        ; Yes, recursively search
            56    44 A6   D0   02E2   680 75$: MOVL    LKB$L_OWNQBL(R6),R6     ; Get previous lock
                  CC   11   02E6   681      BRB     70$                     ; Repeat inner loop - Note we don't
                             02E8   682                                      ; check for end of queue since there
```

```
                     02E8   683
                     02E8   684
          50   D4    02E8   685 90$:     CLRL    R0                      ; No deadlock found
                     02EA   686
                     02EA   687 SEARCH_EXIT:
     007C 8F   BA    02EA   688          POPR    #^M<R2,R3,R4,R5,R6>
               05    02EE   689          RSB
```

```
02EF    691 DEADLOCK_FOUND:
02EF    692 ;           Come here if we found a deadlock.  The stack consists of
02EF    693 ;           a series of stack frames, one for each lock involved in
02EF    694 ;           the deadlock.  Each stack frame consists of the 5 saved
02EF    695 ;           registers (R2 - R6) and a return address.  Note that in
02EF    696 ;           each stack frame the saved R6 points to the lock and the
02EF    697 ;           saved R4 points to the respective PCB lock queue (if the lock
02EF    698 ;           is not a master copy.  In principal, only the first and last
02EF    699 ;           frames could represent master copy locks (the frame that started
02EF    700 ;           this search and the frame that ended it).
02EF    701 ;           The stack frames are bounded by R10 and the current SP.
02EF    702 ;           The following diagram shows the stack with three frames.
02EF    703 ;
02EF    704 ;
02EF    705 ;            SP -> +---------------+           .  +-----------------------+
02EF    706 ;                  |---------------|        .     |       Saved R2        |
02EF    707 ;                  |---------------|     .        +-----------------------+
02EF    708 ;                  |---------------|  .           |       Saved R3        |
02EF    709 ;                  |---------------|  .           +-----------------------+
02EF    710 ;                  |---------------|  .           |       Saved R4        |
02EF    711 ;                  +---------------+.             | (PCB + PCB$L_LOCKQFL) |
02EF    712 ;                  |---------------|              +-----------------------+
02EF    713 ;                  |---------------|              |       Saved R5        |
02EF    714 ;                  |---------------|              +-----------------------+
02EF    715 ;                  |---------------|.             |       Saved R6        |
02EF    716 ;                  +---------------+  .           |   (Address of LKB)    |
02EF    717 ;                  |---------------|     .        +-----------------------+
02EF    718 ;                  |---------------|        .     |     Return Address    |
02EF    719 ;                  |---------------|           .  +-----------------------+
02EF    720 ;                  |---------------|
02EF    721 ;                  |---------------|
02EF    722 ;                  |---------------|
02EF    723 ;                  +---------------+
02EF    724 ;           R10 ->
02EF    725 ;
02EF    726 ;           We will now search the frames looking for the process with
02EF    727 ;           the smallest deadlock priority.  When found, the respective
02EF    728 ;           deadlock priority will be compared with that in the input message
02EF    729 ;           (if any).  The objective is to find the best candidate for a deadlock
02EF    730 ;           victim.  After the deadlock is broken the stack will be trimmed
02EF    731 ;           back so that we will return to
02EF    732 ;           the original caller.  Note that a deadlock priority of zero
02EF    733 ;           causes an immediate exit from the loop.  Register usage will be:
02EF    734 ;
02EF    735 ;                   R0      Current deadlock priority
02EF    736 ;                   R1      Current lock frame pointer
02EF    737 ;                   R2      Minimum deadlock priority, so far
02EF    738 ;                   R3      Best victim frame, so far
02EF    739 ;                   R4      Address of PCB lock queue (current frame)
02EF    740 ;                   R9      Address of input message or zero
02EF    741 ;                   R10     Bottom of stack (start search here)
02EF    742 ;                   SP      Top of stack (end search here)
02EF    743 ;
02EF    744 ;           Note that the following code makes a number of assumptions
02EF    745 ;           regarding the order of registers saved on the stack and their
02EF    746 ;           contents.
02EF    747
```

```
                51    5A    18    C3   02EF   748          SUBL3   #LOCKFRAME,R10,R1       ; Initialize current frame pointer
                      53    51    D0   02F3   749          MOVL    R1,R3                   ; Initialize "best" frame pointer
                      52    01    CE   02F6   750          MNEGL   #1,R2                   ; Initialize "best" deadlock priority
                50    10    A1    D0   02F9   751 20$:     MOVL    16(R1),R0               ; Get LKB address
                            04    E1   02FD   752          BBC     #LKB$V_MSTCPY,-         ; Branch if not master copy
                      06    2A    A0   02FF   753                  LKB$W_STATUS(R0),25$
                50    24    A0    D0   0302   754          MOVL    LKB$L_DLCKPRI(R0),R0    ; Get deadlock priority from master copy
                            08    11   0306   755          BRB     28$
                54    08    A1    D0   0308   756 25$:     MOVL    8(R1),R4                ; Get pointer to PCB lock queue
                50    08    A4    D0   030C   757          MOVL    PCB$L_DLCKPRI-PCB$L_LOCKQFL(R4),R0 ; Get current deadlock pri.
                            12    13   0310   758 28$:     BEQL    35$                     ; Branch if zero - have best victim
                      52    50    D1   0312   759          CMPL    R0,R2                   ; Compare current priority with
                            03    1E   0315   760          BGEQU   30$                     ; previous minimum.
                      52    50    7D   0317   761          MOVQ    R0,R2                   ; This frame becomes "best so far"
                      51    18    C2   031A   762 30$:     SUBL    #LOCKFRAME,R1           ; Move to next frame
                      5E    51    D1   031D   763          CMPL    R1,SP                   ; Reached top of stack yet?
                            D7    1E   0320   764          BGEQU   20$                     ; No, repeat for next frame
                            03    11   0322   765          BRB     40$
                      52    50    7D   0324   766 35$:     MOVQ    R0,R2                   ; Move priority and frame pointer
                                       0327   767
                                       0327   768 40$:     ; Compare lowest deadlock priority so far (R2) with that in the
                                       0327   769          ; input message (if any) and select the lower.  R3 points to "best"
                                       0327   770          ; stack frame.
                                       0327   771
                            59    D5   0327   772          TSTL    R9                      ; Any message?
                            06    13   0329   773          BEQL    45$                     ; No
                24    A9    52    D1   032B   774          CMPL    R2,LKMSG$L_VCTMPRI(R9)  ; Compare priorities
                            0E    1A   032F   775          BGTRU   50$                     ; The one in the message was lower
                                       0331   776
                                       0331   777 45$:     ; The one on the stack was lower;  R3 points to the relavant frame.
                                       0331   778
                56    10    A3    D0   0331   779          MOVL    16(R3),R6               ; Get address of LKB
                      0A    A6    91   0335   780          CMPB    LKB$B_TYPE(R6),-        ; Make sure it's a LKB
                            35         0338   781                  #DYN$C_LKB
                            10    12   0339   782          BNEQ    90$                     ; Bugcheck
                            53    D4   033B   783          CLRL    R3                      ; Indicate we have an LKB address
                            04    11   033D   784          BRB     60$
                                       033F   785
                                       033F   786 50$:     ; The one in the message was lower
                                       033F   787
                52    28    A9    7D   033F   788          MOVQ    LKMSG$L_VCTMLKID(R9),R2 ; Get victim lockid (R2) and CSID (R3)
                                       0343   789
                                       0343   790 60$:     ; Break the deadlock
                                       0343   791
                            0A    10   0343   792          BSBB    LCK$BREAK_DEADLOCK      ; Returns status in R0
                                       0345   793
                5E    5A    18    C3   0345   794          SUBL3   #LOCKFRAME,R10,SP       ; Remove all frames but one from stack
                            9F    11   0349   795          BRB     SEARCH_EXIT             ; Return to original caller
                                       034B   796
                                       034B   797 90$:     BUG_CHECK       NOTLKB,FATAL
```

```
                                                .SBTTL  LCK$BREAK_DEADLOCK - Break a deadlock

                                        ;++
                                        ; FUNCTIONAL DESCRIPTION:
                                        ;
                                        ;       This routine is called to break a deadlock.  The victim lock
                                        ;       has already been selected and may be passed to this routine
                                        ;       as either an LKB address or a lockid.  Note that the specified lock
                                        ;       may not even exist on this system (as either a master or process copy).
                                        ;       Specifically, the following cases are handled:
                                        ;
                                        ;       o  The victim lock is a local copy on this system.  In this
                                        ;          case it is cancelled locally.
                                        ;       o  The victim lock is a process copy on this system.  It is
                                        ;          cancelled locally, but a message is sent to the master system.
                                        ;       o  Any other case sends a message to the process system for that
                                        ;          lock.
                                        ;
                                        ;       If the lock is cancelled here, then we also determine if it
                                        ;       is necessary to send a message to redo the original search.
                                        ;
                                        ; CALLING SEQUENCE:
                                        ;
                                        ;       BSBW    LCK$BREAK_DEADLOCK
                                        ;
                                        ; INPUT PARAMETERS:
                                        ;
                                        ;       R2      Lockid of process copy of lock (only if R3 is not 0)
                                        ;       R3      CSID of process copy of lock (or 0 indicating R6 has
                                        ;               an LKB address)
                                        ;       R6      Address of LKB (only if R3 is 0)
                                        ;       R9      Address of message buffer (or 0 indicating no message)
                                        ;
                                        ; OUTPUT PARAMETERS:
                                        ;
                                        ;       R0      Completion code:
                                        ;                       1  = Deadlock found (normal)
                                        ;                       -1 = Deadlock found; master copy was on this system
                                        ;                            so another deadlock search cannot be repeated
                                        ;                            immediately (or we will find the same one again)
                                        ;
                                        ; SIDE EFFECTS:
                                        ;
                                        ;       R0 - R8 are not preserved
                                        ;--
                                        ;
                                        ;
                                        LCK$BREAK_DEADLOCK::
                53   D5   034F    847          TSTL    R3                      ; Do we have a lockid or LKB address?
                2E   13   0351    848          BEQL    20$                     ; LKB address
50    00000000'GF   D0   0353    849          MOVL    G^CLU$GL_CLUB,R0        ; Get CLUB address
              60 A0   53   D1   0358    850          CMPL    R3,CLUB$L_LOCAL_CSID(R0); Is it the CSID of this system?
                2E   12   035E    851          BNEQ    30$                     ; No
              54   52   D0   0360    852          MOVL    R2,R4                   ; Yes, move lockid
        00000000'GF   16   0363    853          JSB     G^LCK$CVT_ID_TO_LKB     ; and convert to LKB address
                0F   50   E9   0369    854          BLBC    R0,5$                   ; No LKB to cancel; still redo search
                04   E0   036C    855          BBS     #LKB$V_MSTCPY,-         ; Verify not master copy
```

```
                     036E   856                     LKB$W_STATUS(R6),10$
                     0371   857             DISPATCH        LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
                     0371   858                             <-
                     0371   859                             <CONVERT,60$>,-
                     0371   860                             <WAITING,60$>-
                     0371   861                             >
           74  11    037B   862 5$:         BRB     75$                     ; Lock is not waiting; still redo search
                     037D   863
                     037D   864 10$:        BUG_CHECK       LOCKMGRERR,FATAL; Victim lock is master copy
                     0381   865
                     0381   866 20$:        ; Have a LKB address.  See if it's a master copy
                     0381   867
           04  E1    0381   868             BBC     #LKB$V_MSTCPY,-         ; Branch if not master copy
      12 2A A6       0383   869                     LKB$W_STATUS(R6),60$
   52  54 A6  D0     0386   870             MOVL    LKB$L_REMLKID(R6),R2    ; Get process lockid
   53  58 A6  D0     038A   871             MOVL    LKB$L_CSID(R6),R3       ; and CSID
                     038E   872
                     038E   873 30$:        ; Send a message to the process system informing it that it
                     038E   874             ; has a deadlock victim
                     038E   875
00000000'GF  16      038E   876             JSB     G^LCK$SND_DLCKFND       ; Send message
   50  01  CE        0394   877             MNEGL   #1,R0                   ; Set status
               05    0397   878             RSB
                     0398   879
                     0398   880 60$:        ; Here is where we actually break the deadlock.  If the lock was
                     0398   881             ; a new lock request, then it is dequeued.  If the lock was a
                     0398   882             ; conversion, then it is regranted at its old lock mode.  In either case
                     0398   883             ; the completion status (in the LKSB) is SS$_DEADLOCK.
                     0398   884             ; Note that the lock database may change as a result
                     0398   885             ; of the victim lock being dequeued (or regranted).  For example,
                     0398   886             ; when a lock is dequeued, it is possible for other locks to
                     0398   887             ; be granted (possibly the original lock that started the deadlock
                     0398   888             ; search).
                     0398   889             ; The victim lock (R6) may be either a local or process copy lock on
                     0398   890             ; this system.  Get master lockid and CSID and save for later
                     0398   891             ; in order to decide if the original search must be repeated.
                     0398   892
           00000002  0398   893             .IF NE  CA$_MEASURE
00000000'EF  D6      0398   894             INCL    L^PMS$GL_DLCKFND
00000000'EF  D6      039E   895             INCL    L^PMS$GL_DEQ_LOC
                     03A4   896             .ENDC
                     03A4   897
           59  DD    03A4   898             PUSHL   R9                      ; Save R9
        30 A6  DD    03A6   899             PUSHL   LKB$L_LKID(R6)          ; Save lockid
   50   50 A6  D0     03A9   900             MOVL    LKB$L_RSB(R6),R0        ; Get RSB address
        38 A0  DD     03AD   901             PUSHL   RSB$L_CSID(R0)          ; Save CSID of system mastering lock
           05  13     03B0   902             BEQL    65$                     ; It's this system
04 AE   54 A6  D0     03B2   903             MOVL    LKB$L_REMLKID(R6),4(SP) ; Save remote lockid instead
      54   02  D0     03B7   904 65$:        MOVL    S^#LCK$M_CANCEL,R4      ; Set CANCEL flag
57    0E0A 8F  3C     03BA   905             MOVZWL  #SS$_DEADLOCK,R7        ; Set error status
        FC3E' 30      03BF   906             BSBW    LCK$DEQLOCK             ; Cancel lock request
      0230 8F  BA     03C2   907             POPR    #^M<R4,R5,R9>           ; Restore CSID (R4) and LKID (R5) and R9
0124  8F   50  B1     03C6   908             CMPW    R0,#SS$_INSFMEM         ; Were we unable to allocate a LDRP?
           24  13     03CB   909             BEQL    75$                     ; Yes, redo search
        30 50  E9     03CD   910             BLBC    R0,DEQ_ERROR            ; Error - bugcheck
                     03D0   911
                     03D0   912 70$:        ; If this was a purely local search (R9=0), then we are done.
```

```
                                                C 1
                                03D0    913          ; If the original lock that started the search was the victim,
                                03D0    914          ; then it has been removed from the timeout queue.  Otherwise, it
                                03D0    915          ; is still on the timeout queue and we will start another deadlock
                                03D0    916          ; search for it.
                                03D0    917          ; If this was a distributed search (R9<>0), then it is necessary
                                03D0    918          ; to redo the original search unless the original lock was the victim.
                                03D0    919          ; The lockid and CSID of the original lock is in the message.
                                03D0    920          ; R4 and R5 contain the lockid and CSID of the lock chosen as
                                03D0    921          ; victim.  Note that in both cases we are referring to the master
                                03D0    922          ; lockid and CSID.
                                03D0    923
                          59    D5    03D0    924          TSTL    R9                              ; Was this a local search?
                          28    13    03D2    925          BEQL    80$                             ; Yes, exit
                    52    14  A9    7D    03D4    926          MOVQ    LKMSG$L_ORIGLKID(R9),R2 ; Get original lockid (R2) and CSID (R3)
               50   00000000'GF    D0    03D8    927          MOVL    G^CLU$GL_CLUB,R0                ; Get address of CLUB
                    60  A0    53    D1    03DF    928          CMPL    R3,CLUB$L_LOCAL_CSID(R0); Is the CSID this system?
                          02    12    03E3    929          BNEQ    72$                             ; No
                          53    D4    03E5    930          CLRL    R3                              ; Yes, use zero for local CSID
                    54    53    D1    03E7    931 72$:     CMPL    R3,R4                           ; Do CSIDs match?
                          05    12    03EA    932          BNEQ    75$                             ; No
                    55    52    D1    03EC    933          CMPL    R2,R5                           ; Do lockids match?
                          0B    13    03EF    934          BEQL    80$                             ; Yes, victim was original lock
                                      03F1    935
                                03F1    936 75$:     ; Must redo the original search (as long as we have a message (R9)
                                03F1    937          ; with the original CSID and lockid)
                                03F1    938
                    52    59    D0    03F1    939          MOVL    R9,R2                           ; Move address of message
                          06    13    03F4    940          BEQL    80$
               00000000'GF        16    03F6    941          JSB     G^LCK$SND_REDO_SRCH             ; Redo the search
                    50    01    D0    03FC    942 80$:     MOVL    #1,R0
                          05    03FF    943          RSB
                                0400    944
                                0400    945
                                0400    946 DEQ_ERROR:
                                0400    947          BUG_CHECK       LOCKMGRERR,FATAL ; Lock was granted or other dequeue
                                0404    948                                          ; error
                                0404    949
                                0404    950
                                0404    951
                                0404    952
                                0404    953          .END
```

```
$$BASE                          = FFFFFFFF
$$DISPL                         = 00000001
$$GENSW                         = 00000001
$$HIGH                          = 00000000
$$LIMIT                         = 00000001
$$LOW                           = FFFFFFFF
$$MNSW                          = 00000001
$$MXSW                          = 00000001
BUG$_LOCKMGRERR                   ********    X   02
BUG$_NOTLKB                       ********    X   02
CA$_MEASURE                     = 00000002
CLU$GL_CLUB                       ********    X   02
CLUB$L_LOCAL_CSID               = 00000060
DEADLOCK_FOUND                    000002EF R      02
DEQ_ERROR                         00000400 R      02
DYN$C_LKB                       = 00000035
DYN$C_RSB                       = 00000036
EXE$GL_ABSTIM                     ********    X   02
EXE$GL_INTSTKLM                   ********    X   02
EXE$GQ_SYSTIME                    ********    X   02
LCK$BREAK_DEADLOCK                0000034F RG     02
LCK$COMPAT_TBL                    ********    X   02
LCK$CVT_ID_TO_LKB                 ********    X   02
LCK$DEQLOCK                       ********    X   02
LCK$DLCKEXIT                      000000F4 RG     02
LCK$GB_STALLREQS                  ********    X   02
LCK$GL_EXTRASTK                   ********    X   02
LCK$GL_PRCMAP                     ********    X   02
LCK$GL_TIMOUTQ                    ********    X   02
LCK$GQ_BITMAP_EXP                 ********    X   02
LCK$K_CWMODE                    = 00000002
LCK$K_PRMODE                    = 00000003
LCK$K_PWMODE                    = 00000004
LCK$M_CANCEL                    = 00000002
LCK$SEARCHDLCK                    00000000 RG     02
LCK$SND_DLCKFND                   ********    X   02
LCK$SND_GRANTED                   ********    X   02
LCK$SND_REDO_SRCH                 ********    X   02
LCK$SND_RMVDIR                    ********    X   02
LCK$SND_SRCHDLCK                  ********    X   02
LCK$SND_TIMESTAMP_RQST            ********    X   02
LCK$SRCH_RESDLCK                  00000162 RG     02
LCK$V_NODLCKBLK                 = 0000000A
LCK$V_NODLCKWT                  = 00000009
LKB$B_GRMODE                    = 00000035
LKB$B_RQMODE                    = 00000034
LKB$B_STATE                     = 00000036
LKB$B_TYPE                      = 0000000A
LKB$K_CONVERT                   = 00000000
LKB$K_GRANTED                   = 00000001
LKB$K_WAITING                   = FFFFFFFF
LKB$L_ASTQFL                    = 00000000
LKB$L_CSID                      = 00000058
LKB$L_DLCKPRI                   = 00000024
LKB$L_DUETIME                   = 00000018
LKB$L_EPID                      = 00000014
LKB$L_LKID                      = 00000030
LKB$L_OWNQBL                    = 00000044
LKB$L_OWNQFL                    = 00000040
LKB$L_PID                       = 0000000C
LKB$L_REMLKID                   = 00000054
LKB$L_RSB                       = 00000050
LKB$L_SQBL                      = 0000003C
LKB$L_SQFL                      = 00000038
LKB$M_TIMOUTQ                   = 00000040
LKB$V_MSTCPY                    = 00000004
LKB$W_FLAGS                     = 00000028
LKB$W_STATUS                    = 0000002A
LKMSG$L_ORIGLKID                = 00000014
LKMSG$L_VCTMLKID                = 00000028
LKMSG$L_VCTMPRI                 = 00000024
LOCKFRAME                       = 00000018
PCB$L_DLCKPRI                   = 0000010C
PCB$L_EPID                      = 00000064
PCB$L_LOCKQFL                   = 00000104
PMS$GL_DEQ_LOC                    ********    X   02
PMS$GL_DLCKFND                    ********    X   02
PMS$GL_DLCKSRCH                   ********    X   02
RSB$L_CSID                      = 00000038
RSB$L_CVTQFL                    = 00000018
RSB$L_GRQFL                     = 00000010
RSB$L_WTQFL                     = 00000020
SCH$GL_PCBVEC                     ********    X   02
SEARCH_CVTDLCK                    000000F9 R      02
SEARCH_EXIT                       000002EA R      02
SS$_DEADLOCK                    = 00000E0A
SS$_INSFMEM                     = 00000124
STATE_ERROR                       0000015E 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
LOCKMGR	00000404 (1028.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.09	00:00:01.10
Command processing	120	00:00:00.70	00:00:05.42
Pass 1	407	00:00:14.80	00:00:53.94
Symbol table sort	0	00:00:02.22	00:00:07.64
Pass 2	185	00:00:03.42	00:00:12.82
Symbol table output	11	00:00:00.10	00:00:00.10
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	762	00:00:21.36	00:01:21.05

The working set limit was 1800 pages.
84198 bytes (165 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1437 non-local and 70 local symbols.
953 source lines were read in Pass 1, producing 17 object records in Pass 2.
23 pages of virtual memory were used to define 21 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]CLUSTER.MLB;1	1
_255DUA28:[SYS.OBJ]LIB.MLB;1	8
_255DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	15

1544 GETS were required to define 15 macros.

There were no errors, warnings or information messages.

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

0373 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CMODSSDSP
LIS

BUGCHECK
LIS

COMDRVSUB
LIS

CLUSTRVEC
LIS

BUFFERCTL
LIS

DEADLOCK
LIS

CVTFILNAM
LIS

BOOPARAM
LIS

CJFSYSVEC
LIS

CVTATB
LIS

BUGCHKMSG
LIS

CONSOLIO
LIS

FILEREAD
LIS

EXCEPTMSG
LIS

DISMOUNT
LIS

DEBUGDATA
LIS

ERRORLOG
LIS

DEVICEDAT
LIS

EXCEPTION
LIS

FILERWIO
LIS

EXSUBRÔUT
LIS