


```

DDDDDDDD  SSSSSSSS  TTTTTTTTTT  RRRRRRRR  DDDDDDDD  LL  CCCCCCCC  KK  KK
DDDDDDDD  SSSSSSSS  TTTTTTTTTT  RRRRRRRR  DDDDDDDD  LL  CCCCCCCC  KK  KK
DD DD  SS  TT  RR RR  DD DD  LL  CC  CC  KK  KK
DD DD  SS  TT  RR RR  DD DD  LL  CC  CC  KK  KK
DD DD  SS  TT  RR RR  DD DD  LL  CC  CC  KK  KK
DD DD  SS  TT  RRRRRRRR  DD DD  LL  CC  CC  KK  KK
DD DD  SS  TT  RRRRRRRR  DD DD  LL  CC  CC  KKKKKK  KK
DD DD  SS  TT  RR RR  DD DD  LL  CC  CC  KKKKKK  KK
DD DD  SS  TT  RR RR  DD DD  LL  CC  CC  KK  KK
DD DD  SS  TT  RR RR  DD DD  LL  CC  CC  KK  KK
DD DD  SS  TT  RR RR  DD DD  LL  CC  CC  KK  KK
DDDDDDDD  SSSSSSSS  TT  RR RR  DDDDDDDD  LLLLLLLLLL  CCCCCCCC  KK  KK
DDDDDDDD  SSSSSSSS  TT  RR RR  DDDDDDDD  LLLLLLLLLL  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
LLLLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLLLL  IIIIII  SSSSSSSS

```

(2)	70	DECLARATIONS
(3)	157	LCK\$SND_TIMESTAMP_RQST
(4)	286	GET_TIMESTAMP - Get a bitmap time stamp
(5)	356	LCK\$RCV_TIMESTAMP_RQST
(6)	437	CHECK_TIMESTAMP - Check bitmap timestamp
(7)	503	LCK\$SND_SRCHDLCK - Send deadlock search message
(8)	728	LCK\$RCV_SRCHDLCK - Receive search deadlock message
(9)	846	LCK\$SND_DLCKFND - Send deadlock found message
(10)	943	LCK\$RCV_DLCKFND - Receive deadlock found message
(11)	981	LCK\$SND_REDO_SRCH - Send a redo deadlock search message
(12)	1046	LCK\$RCV_REDO_SRCH
(13)	1077	REDO_SRCH - Redo deadlock search
(14)	1163	LCK\$CVT_ID_TO_LKB - Convert a lockid to LKB address
(15)	1206	SEND_DLCK_MSG - Send any deadlock detection message
(16)	1243	RCV_DLCK_MSG - Receive a deadlock message
(17)	1287	DEACL_DLCK_MSG - Deallocate deadlock message buffer
(18)	1321	LCK\$ACLOC_CONGCDRP - Allocate a long CDRP
(19)	1365	WAIT_FOR_POOL - Wait for pool

```

0000 1 .TITLE DSTRDLCK - DISTRIBUTED 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 distributed deadlock detection (and resolution)
0000 32 : for the VMS lock manager system services ($ENQ and $DEQ) when
0000 33 : operating in a VAXcluster environment.
0000 34
0000 35 : ENVIRONMENT: VAX/VMS, VAXcluster loadable code
0000 36
0000 37 : AUTHOR: Steve Beckhardt, CREATION DATE: 28-Feb-1984
0000 38
0000 39 : MODIFIED BY:
0000 40
0000 41 : V03-006 SRB0143 Steve Beckhardt 9-Jul-1984
0000 42 : Changed handling of repeated failures to complete a deadlock
0000 43 : search. Instead of calling it a deadlock, the lock is now
0000 44 : placed back on the end of the time out queue with a fresh
0000 45 : wait time and retry count. This should eliminate the
0000 46 : occasional false deadlocks. As a result, the maximum
0000 47 : timestamp lifetime was reduced to 5 (1.6 secs.)
0000 48
0000 49 : V03-005 SRB0137 Steve Beckhardt 9-Jul-1984
0000 50 : Fixed bugs in timestamp lifetime code. Increased
0000 51 : maximum timestamp lifetime to 6 (3.2 secs.)
0000 52
0000 53 : V03-004 SRB0134 Steve Beckhardt 22-Jun-1984
0000 54 : Fixed bug in stack handling in insufficient pool code.
0000 55
0000 56 : V03-003 SRB0130 Steve Beckhardt 18-May-1984
0000 57 : Fixed bug involving location of test for NODLCKWT flag.

```

0000	58	:			
0000	59	:	V03-002	SRB0125	Steve Beckhardt
0000	60	:			2-May-1984
0000	61	:			Fixed bug involving race between process level handling
0000	62	:			of lock granted message (getting lock on PCB queue) and
0000	63	:			receiving deadlock search message.
0000	64	:	V03-001	SRB0119	Steve Beckhardt
0000	65	:			10-Mar-1984
0000	66	:			Fixed bugs, added support for LCKSM_NODLCKWT flag.
0000	67	:			Added support for waiting for pool.
0000	68	:			--

```

0000 70          .SBTTL  DECLARATIONS
0000 71          :
0000 72          : INCLUDE FILES:
0000 73          :
0000 74          :
0000 75          :
0000 76          : EXTERNAL SYMBOLS:
0000 77          :
0000 78          :
0000 79          $CADEF          : Conditional assembly switches
0000 80          $CDRPDEF       : CDRP offsets
0000 81          $CLMSGDEF      : Cluster message offsets
0000 82          $CLUBDEF       : CLUB offsets
0000 83          $CSBDEF        : CSB offsets
0000 84          $DYNDEF        : Structure type code definitions
0000 85          $FKBDEF        : Fork block offsets
0000 86          $IPLDEF        : IPL definitions
0000 87          $LCKDEF        : LCK definitions
0000 88          $LKBDEF        : LKB offsets
0000 89          $PCBDEF        : PCB offsets
0000 90          $RSBDEF        : RSB offsets
0000 91          $SSDEF         : System status code definitions
0000 92          :
0000 93          :
0000 94          : MACROS:
0000 95          :
0000 96          :
0000 97          :
0000 98          : EQUATED SYMBOLS:
0000 99          :
0000 100         :
0007A120 0000 101 TSLT_UNITS = 50*1000*10      : Time stamp lifetime units (50 ms.)
00000005 0000 102 MAX_TSLT = 5                 : Maximum timestamp lifetime shift count
0000 103         : (represents 1.6 secs.)
0000 104         : This also represents the interval
0000 105         : that must elapse before local searches
0000 106         : can proceed without timestamps
0000 107         :
00000018 0000 108 LOCKFRAME = 24                : Number of bytes pushed onto
0000 109         : stack for each recursive call
0000 110         : of SEARCH_RESDLCK (5 registers
0000 111         : plus return address). This
0000 112         : cannot be changes without making
0000 113         : corresponding coding changes
0000 114         : This must also agree with the
0000 115         : symbol of the same name in
0000 116         : DEADLOCK.MAR.
0000 117         :
0000 118         :
0000 119         : OWN STORAGE:
0000 120         :
0000 121         :
00000000 0000 122         .PSECT $$$040, LONG
0000 123         :
0000 124         .ALIGN LONG
0000 125         :
0000 126         :

```

```

0000 127 : NOTE: The fork block and lock message buffer must be adjacent
0000 128 :
0000 129 LKMSG_FKB:
00000000 00000000 0000 130 .QUAD 0 ; Queue Links
0018 0008 131 .WORD FKBSK_LENGTH ; Size
08 000A 132 .BYTE DYN$C_FRK ; Type
08 000B 133 .BYTE IPL$ SYNCH ; Fork IPL
00000018 000C 134 .BLKB FKBSK_LENGTH 12 ; Remainder of fork block
0018 135 LKMSG_BFR: ; Buffer to use for lock messages
0000004C 0018 136 .BLKB LKMSG$K_DLM_LENGTH
004C 137
004C 138 :*****
004C 139 :
004C 140 : NOTE: The following assumptions are in effect for this entire module
004C 141 :
004C 142 :*****
004C 143 :
004C 144 ASSUME LKMSG$B_TSLT EQ 2+LKMSG$W_MEMSEQ
004C 145 ASSUME LKMSG$L_ORIGEPI EQ 2+LKMSG$B_TSLT
004C 146 ASSUME LKMSG$L_ORIGLKID EQ 4+LKMSG$L_ORIGEPI
004C 147 ASSUME LKMSG$L_ORIGCSID EQ 4+LKMSG$L_ORIGLKID
004C 148 ASSUME LKMSG$Q_BITMAP_EXP EQ 4+LKMSG$L_ORIGCSID
004C 149 ASSUME LKMSG$L_VCTMPRI EQ 8+LKMSG$Q_BITMAP_EXP
004C 150 ASSUME LKMSG$L_VCTMLKID EQ 4+LKMSG$L_VCTMPRI
004C 151 ASSUME LKMSG$L_VCTMCSID EQ 4+LKMSG$L_VCTMLKID
004C 152 ASSUME LKMSG$L_NEXTLKID EQ 4+LKMSG$L_VCTMCSID
004C 153
004C 154
00000000 155 .PSECT $$$020

```

```

0000 157      .SBTTL LCK$SND_TIMESTAMP_RQST
0000 158
0000 159      :++
0000 160      : FUNCTIONAL DESCRIPTION:
0000 161      :
0000 162      : This routine sends a timestamp request to the system issuing
0000 163      : timestamps unless this system is issuing timestamps. If a message
0000 164      : is actually sent, then this routine does not return to its caller.
0000 165      : Instead, the stack is unwound and we exit deadlock detection.
0000 166      : This routine only returns to its caller if this system
0000 167      : is issuing timestamps and we successfully get one.
0000 168
0000 169      : CALLING SEQUENCE:
0000 170
0000 171      BSBW LCK$SND_TIMESTAMP_RQST
0000 172      Note: This routine only returns to its caller if a timestamp
0000 173      is issued locally. In all other cases, the stack is
0000 174      reset and we exit from deadlock detection.
0000 175
0000 176      : INPUT PARAMETERS:
0000 177
0000 178      R8 EPID of original process
0000 179      R10 Stack position to unwind to
0000 180
0000 181      : IMPLICIT INPUTS:
0000 182
0000 183      It is assumed that the lock that started the deadlock search is
0000 184      still at the head of the timeout queue.
0000 185
0000 186      : OUTPUT PARAMETERS:
0000 187
0000 188      R9 Address of a message buffer template to be used instead
0000 189      of a real message buffer if a timestamp is assigned locally
0000 190
0000 191      : SIDE EFFECTS:
0000 192
0000 193      The bitmap is cleared if we issue a time stamp
0000 194      :--
0000 195
0000 196
0000 197 LCK$SND_TIMESTAMP_RQST::
54 00000000'GF DO 0000 198      MOVL G*LCK$GL_TIMEOUT,R4 ; Get original lock (from head of queue)
53 00000000'EF DO 0007 199      MOVL LCK$GL_TS_CSID,R3 ; Get CSID of system issuing timestamps
      29 13 000E 200      BEQL 40$ ; It's us
      049A 30 0010 201      BSBW LCK$ALLOC_LONGCDRP ; Allocate a CDRP
      51 50 E9 0013 202      BLBC R0,70$ ; Error
      0016 203
      0016 204      ; Store necessary info. in CDRP to be able to start a deadlock
      0016 205      ; search later.
      0016 206
      52 2C A5 9E 0016 207      MOVAB CDRP$L_VAL1(R5),R2 ; Point into CDRP data area
      54 10 001A 208      BSBB 80$ ; Store data in CDRP
      5C A5 50 DO 001C 209      MOVL R0,CDRP$L_VAL9(R5) ; Store victim CSID
      30 A4 DO 0020 210      MOVL LK$SL_LKID(R4),- ; Store next lockid
      60 A5 0023 211      CDRP$L_VAL10(R5)
      0270'CF 9E 0025 212      MOVAB W^BLD_TIMESTAMP_RQST,- ; Store address of message build routine
      4C A5 0029 213      CDRP$L_MSGBLD(R5)

```



```

002B 214
002B 215 ; Remove lock from timeout queue, reset the stack, and send the message.
002B 216
50 64 OF 002B 217 REMQUE LKB$L_ASTQFL(R4),R0 ; Remove LKB from timeout queue
0040 8F AA 002E 218 BICW #LKB$M_TIMEOUTQ,- ; Clear corresponding status bit
2A A4 0032 219 LKB$W_STATUS(R4)
0443 30 0034 220 BSBW SEND_DLCK_MSG ; Send the message
2E 11 0037 221 BRB 70$
0039 222
0039 223 40$: ; We are issuing timestamps
0039 224
54 55 54 DO 0039 225 MOVL R4,R5 ; Save LKB address
4E A4 9A 003C 226 MOVZBL LKB$B_TSLT(R4),R4 ; Get timestamp lifetime
6D 10 0040 227 BSBW GET_TIMESTAMP
22 54 E9 0042 228 BLBC R4,70$ ; Bitmap in use
54 55 DO 0045 229 MOVL R5,R4 ; Restore LKB address
0018 CF 9E 0048 230 MOVAB W^LKMSG BFR,R9 ; Point to internal message buffer
E8 A9 D5 004D 231 TSTL -FKB$K_LENGTH(R9) ; Is it in use?
15 12 0050 232 BNEQ 70$ ; Yes
08 A9 0A02 8F B0 0052 233 MOVW #LKMSG$K_SRCHDLCK@8- ; Store facility and function codes
0058 234 !CLMSG$R_FAC_LCK,CLMSG$B_FACILITY(R9)
52 0C A9 9E 0058 235 MOVAB LKMSG$W_MEMSEQ(R9),R2 ; Point to data area
12 10 005C 236 BSBW 80$ ; Fill in fields
51 30 A4 DO 005E 237 MOVL LKB$L_LKID(R4),R1 ; Get next lockid to search (this one)
2C A9 50 7D 0062 238 MOVQ R0,LKMSG$L_VCTMCSID(R9) ; Store victim CSID and next lockid
05 0066 239 RSB ; Return to caller
0067 240
0067 241 70$: ; This exit unwinds the stack and exits deadlock detection.
0067 242 ; If we sent a message then the original lock has been removed
0067 243 ; from the timeout queue. We want to exit deadlock detection rather
0067 244 ; than trying to search for another deadlock because we will be
0067 245 ; unable to get another timestamp. If we were unable to allocate a CDRP
0067 246 ; or the bitmap was in use then we leave the lock on the timeout queue
0067 247 ; so that we will retry this operation 1 second from now.
0067 248
5E 5A DO 0067 249 MOVL R10,SP ; Reset stack
00000000 GF 17 006A 250 JMP G^LCK$DLCKEXIT ; Return
0070 251
0070 252 ;
0070 253 ; Local subroutine to store message data in either CDRP or internal
0070 254 ; message template
0070 255 ;
0070 256 ; Inputs: R0,R1 Timestamp expiration (if assigned locally)
0070 257 ; R2 Address of data area in CDRP or internal message bfr
0070 258 ; R4 Address of original LKB
0070 259 ; R8 Original EPID
0070 260 ;
0070 261 ; Outputs:
0070 262 ; R0 Victim CSID (not stored in data area because
0070 263 ; CDRP$L_VAL9 is not contiguous with CDRP$L_VAL8.)
0070 264 ;
0070 265 ;
55 00000000 GF 55 DD 0070 266 80$: PUSHL R5
82 00AC C5 B0 0072 267 MOVL G^CLUS$GL CLUB,R5 ; Get address of CLUB
82 4E A4 98 0079 268 MOVW CLUB$W_MEMSEQ(R5),(R2)+ ; Store memseq
82 58 DO 007E 269 MOVZBW LKB$B_TSLT(R4),(R2)+ ; Store timestamp lifetime
0082 270 MOVL R8,(R2)+ ; Store original EPID

```

```

82 30 A4 DO 0085 271      MOVL LKBSL_LKID(R4),(R2)+ ; Store original lockid
82 60 A5 DO 0089 272      MOVL CLUB$C_LOCAL_CSID(R5),(R2)+ ; Store CSID of this system
      82 50 7D 008D 273      MOVQ R0,(R2)+ ; Store timestamp
      82 01 CE 0090 274      MNEGL #1,(R2)+ ; Initialize victim priority
51 30 A4 DO 0093 275      MOVL LKBSL_LKID(R4),R1 ; Get local lockid
50 60 A5 DO 0097 276      MOVL CLUB$C_LOCAL_CSID(R5),R0 ; and local CSID
      04 E1 009B 277      BBC #LKBSV_MSTCPY,- ; Branch if not master copy
      08 2A A4 DO 009D 278      LKBSW_STATUS(R4),85$
51 54 A4 DO 00A0 279      MOVL LKBSL_REMLKID(R4),R1 ; Get remote lockid instead
50 58 A4 DO 00A4 280      MOVL LKBSL_CSID(R4),R0 ; and remote CSID
      82 51 DO 00A8 281 85$: MOVL R1,(R2)+ ; Store victim lockid; return victim
      05 B EDO 00AB 282      POPL R5 ; CSID in R0
      05 00AE 283
      00AF 284

```

```

00AF 286          .SBTTL GET_TIMESTAMP - Get a bitmap time stamp
00AF 287
00AF 288      :++
00AF 289      : FUNCTIONAL DESCRIPTION:
00AF 290      :
00AF 291      : This routine returns a bitmap timestamp with a specified lifetime.
00AF 292      : The timestamp lifetime is encoded as a shift count that represents
00AF 293      : the number of bits the basic lifetime should be shifted.
00AF 294      : For example, if the basic lifetime unit is 50 ms. (TSLT_UNITS) then
00AF 295      : a specified lifetime (R4) of 2 would return a timestamp with
00AF 296      : an expiration time 200 ms. from now.
00AF 297
00AF 298      : CALLING SEQUENCE:
00AF 299
00AF 300          BSBW  GET_TIMESTAMP
00AF 301
00AF 302      : INPUT PARAMETERS:
00AF 303
00AF 304          R4    Timestamp lifetime (encoded as a shift count)
00AF 305
00AF 306      : OUTPUT PARAMETERS:
00AF 307
00AF 308          R0,R1  Quadword expiration time (success only)
00AF 309          R4    Completion code: 0 = failure
00AF 310
00AF 311
00AF 311          1 = success
00AF 312      : SIDE EFFECTS:
00AF 313
00AF 314          On success, bitmap is cleared, new expiration time is
00AF 315          stored as both local and exact expiration time
00AF 316      :--
00AF 317
00AF 318 GET_TIMESTAMP:
00AF 319      : Determine if the previous timestamp has expired yet.
00AF 320      : Note that normally, this test should be performed at the IPL
00AF 321      : of the hardware clock interrupt (IPL$HWCLK). However, we can
00AF 322      : tolerate the race condition here. The result would be to think
00AF 323      : that the bitmap is in use when it really wasn't. If this occurs,
00AF 324      : we will simply try again later.
00AF 325
00AF 326          PUSHR  #^M<R2,R3,R5>
52 00000000'GF 2C BB 00AF 327          MOVAQ  G^LCK$GQ_BITMAP_EXP,R2 ; Get address of expiration time
50 00000000'GF 7E 00B1 328          MOVAQ  G^EX$GQ_SYSTIME,R0 ; Get address of system time
04 A0 04 A2 D1 00BF 329          CMPL   4(R2),4(R0) ; Compare low order time
0C 1F 00C4 330          BLSSU  20$ ; Bitmap is available
05 1A 00C6 331          BGTRU  10$ ; Bitmap is in use
60 62 D1 00C8 332          CMPL   (R2),(R0) ; Compare high order time
05 1B 00CB 333          BLEQU  20$ ; Bitmap is available
00CD 334
00CD 335 10$: ; Bitmap is in use. Return failure.
00CD 336
05 64 D4 00CD 337          CLRL   R4
2C BA 00CF 338          POPR  #^M<R2,R3,R5>
05 00D1 339          RSB
00D2 340
00D2 341 20$: ; Bitmap is available. Compute new expiration times and clear bitmap.
00D2 342

```

54	0007A120	8F	54	78	00D2	343	ASHL	R4,#TSLT_UNITS,R4	; Compute time stamp lifetime
		50	60	7D	00DA	344	MOVQ	(R0),R0	; Get current time
		50	54	C0	00DD	345	ADDL	R4,R0	; Compute expiration time
		51	00	D8	00E0	346	ADWC	#0,R1	
		62	50	7D	00E3	347	MOVQ	R0,(R2)	; Store expiration time
	08	A2	50	7D	00E6	348	MOVQ	R0,8(R2)	; Store local expiration time
		7E	50	7D	00EA	349	MOVQ	R0,-(SP)	; Save timestamp for return to caller
60	50	00000000	'GF	D0	00ED	350	MOVL	G^LCK\$GL_PRCMAP,R0	; Get address of bitmap
F8	A0	00	60	2C	00F4	351	MOVCS	#0,(R0),#0,-8(R0),(R0)	; Clear it
		54	01	D0	00FB	352	MOVL	#1,R4	; Return success
			2F	BA	00FE	353	POPR	#^M<R0,R1,R2,R3,R5>	
				05	0100	354	RSB		

```

0101 356          .SBTTL LCK$RCV_TIMESTAMP_RQST
0101 357
0101 358 :++
0101 359 : FUNCTIONAL DESCRIPTION:
0101 360 :
0101 361 :     This routine is called by the received message dispatcher when
0101 362 :     we receive a request for a timestamp.  If we can assign a timestamp
0101 363 :     then we send a message that starts the deadlock search.  If we
0101 364 :     cannot assign a timestamp (because the previous one has not
0101 365 :     expired yet), then we send a message that will cause the original
0101 366 :     lock to be requeued to the timeout queue.
0101 367 :
0101 368 : CALLING SEQUENCE:
0101 369 :
0101 370 :     JSB      LCK$RCV_TIMESTAMP_RQST  (called by received message dispatcher)
0101 371 :
0101 372 : INPUT PARAMETERS:
0101 373 :
0101 374 :     R2      Address of message buffer
0101 375 :     R3      Address of CSB
0101 376 :
0101 377 : OUTPUT PARAMETERS:
0101 378 :
0101 379 :     None
0101 380 :
0101 381 : SIDE EFFECTS:
0101 382 :
0101 383 :     R0 - R5 not preserved
0101 384 :
0101 385 :     If a timestamp is assigned, the bitmap is cleared and the new
0101 386 :     expiration time is stored as both the local and exact expiration time.
0101 387 :--
0101 388
0101 389 LCK$RCV_TIMESTAMP_RQST::
0388 30 0101 390      BSBW      REV_DLCK_MSG
          OC  BB 0104 391      PUSHR     #^M^R2,R3>
0000'CF  D5 0106 392      TSTL      W^LCK$GL_TS_CSID      ; Verify we are assigning timestamps
          46 12 010A 393      BNEQ      70$              ; Error!
          010C 394
          010C 395      ; Get a timestamp
          010C 396
54 0E A2 9A 010C 397      MOVZBL   LKMSG$B_TSLT(R2),R4      ; Get timestamp lifetime
          9D 10 0110 398      BSBB      GET_TIMESTAMP
          05 54 E8 0112 399      BLBS      R4,20$              ; Success
          02A3 30 0115 400      BSBW      LCK$SND_REDO_SRCH      ; Failure - redo deadlock search
          33 11 0118 401      BRB       50$
          011A 402
          011A 403 20$:      ; Have a timestamp.  Send a message that will initiate the deadlock
          011A 404      ; search.  Store all necessary fields in the CDRP.
          011A 405
          53 50 7D 011A 406      MOVQ      R0,R3              ; Move timestamp
          038D 30 011D 407      BSBW      LCK$ALLOC_LONGCDRP      ; Allocate a CDRP
          05 50 E8 0120 408      BLBS      R0,30$              ; Have one
          03A5 30 0123 409      RSBW      WAIT_FOR_POOL
          25 11 0126 410      BRB       50$
          0128 411
0278'CF 9E 0128 412 30$:  MOVAB     W^BLD_SRCHDLCK,-      ; Store address of message build routine

```

4C	A5		012C	413		CDRPSL MSGBLD(R5)		
0C	A2	7D	012E	414	MOVQ	LKMSG\$ MEMSEQ(R2),-	: Store memseq, timestamp lifetime,	
2C	A5		0131	415		CDRPSL VAL1(R5)	: and original EPID	
14	A2	7D	0133	416	MOVQ	LKMSG\$ ORIGLKID(R2),-	: Store original lockid and CSID	
34	A5		0136	417		CDRPSL VAL3(R5)		
3C	A5	53	7D	0138	418	MOVQ	R3,CDRPSL VAL5(R5)	: Store timestamp
	A2	7D	013C	419	MOVQ	LKMSG\$ VCTMPRI(R2),-	: Store deadlock victim priority	
44	A5		013F	420		CDRPSL VAL7(R5)	: and lockid	
2C	A2	7D	0141	421	MOVQ	LKMSG\$ VCTMCSID(R2),-	: Store deadlock victim CSID and	
5C	A5		0144	422		CDRPSL VAL9(R5)	: next lockid	
			0146	423				
			0146	424			: End the message	
			0146	425				
53	18	A2	D0	0146	426	MOVL	LKMSG\$ ORIGCSID(R2),R3 ; Get CSID of original system	
	032D		30	014A	427	BSBW	SEND_DLCK_MSG	
				014D	428			
				014D	429	50\$:	: Deallocate the message buffer and return	
				014D	430			
	0C	BA		014D	431	POPR	#^M<R2,R3>	
	034D	31		014F	432	BRW	DEALL_DLCK_MSG	
				0152	433			
				0152	434			
				0152	435	70\$:	BUG_CHECK LOCKMGRERR,FATAL; This system is not issuing timestamps	

0156 437 .SBTTL CHECK_TIMESTAMP - Check bitmap timestamp

0156 438

0156 439 :++

0156 440 : FUNCTIONAL DESCRIPTION:

0156 441

0156 442

0156 443

0156 444

0156 445

0156 446

0156 447

0156 448

0156 449

0156 450

0156 451

0156 452

0156 453

0156 454

0156 455

0156 456

0156 457

0156 458

0156 459

0156 460

0156 461

0156 462

0156 463

0156 464

0156 465

0156 466

0156 467

0156 468

0156 469

0156 470

0156 471

0156 472

0156 473

0156 474

0156 475

0156 476

0156 477

0156 478

0156 479

0156 480

0156 481

0156 482

0156 483

0175 484

0175 485

0175 486

0175 487

0175 488

0175 489

0179 490

0178 491

0182 492

0189 493

CALLING SEQUENCE:

BSBW CHECK_TIMESTAMP

INPUT PARAMETERS:

R2 Address of message buffer

OUTPUT PARAMETERS:

R0 Completion code: 0 = abort deadlock search
1 = use bitmap

R7 Address of bitmap

SIDE EFFECTS:

If the bitmap is cleared, then the local expiration time
is reset.

R1 is not preserved

CHECK_TIMESTAMP:

MOVL G^LCK\$GL_PRCMAP,R7 ; Get address of bitmap

MOVAQ G^LCK\$GQ_BITMAP_EXP,R1 ; Get address of bitmap timestamp

CPL 4(R1),LKMSG\$Q_BITMAP_EXP+4(R2) ; Compare high order times

BLSSU 10\$; Reuse bitmap

BGTFU 40\$; Bitmap has been preempted

CMP (R1),LKMSG\$Q_BITMAP_EXP(R2) ; Compare low order times

BGTRU 40\$; Bitmap has been preempted

BEQL 20\$; Continue using bitmap

10\$: ; Eitmap may be used after it is initialized. Store new timestamps.
; The expiration timestamp is the one in the message. The local
; timestamp is the current system time plus the maximum timestamp
; lifetime.

MOVQ LKMSG\$Q_BITMAP_EXP(R2),(R1) ; Store new expiration timestamp

PUSHR #^M<R2,R3,R4,R5>

MOVQ G^EXE\$GQ_SYSTIME,R2 ; Get this system's time

ADDL #<TSLT_UNITS@MAX_TSLT>,R2 ; Add maximum timestamp lifetime

ADWC #0,R3

57 00000000'GF D0
51 00000000'GF 7E
20 A2 04 A1 D1
0A 1F C169
30 1A C16B
1C A2 61 D1
2A 1A 0171
24 13 0173
0175
0175
0175
0175
0175
61 1C A2 7D
3C BB 0179
52 00000000'GF 7D
52 00F42400 8F C0
53 00 D8

```
67  FB A7  00  08 A1  52  7D 018C  494  MOVQ  R2,8(R1) ; Store local expiration time
      67  00  2C 0190  495  MOVCS #0,(R7),#0,-8(R7),(R7) ; Clear it
      3C  BA 0197  496  POPR  #^M<R2,R3,R4,R5>
      50  01  D0 0199  497 20$: MOVL  #1,R0
      05  019C  498  RSB
      05  019D  499
      50  D4 019D  500 40$: CLRL  R0 ; Bitmap was preempted
      05  019F  501  RSB
```



```

01A0 503      .SBTTL LCK$SND_SRCHDLCK - Send deadlock search message
01A0 504      :++
01A0 505      : FUNCTIONAL DESCRIPTION:
01A0 506      :
01A0 507      :     This routine sends a search for deadlock message when either
01A0 508      :     a master copy lock is blocking another lock or a waiting
01A0 509      :     lock is found that is mastered on another system. In effect,
01A0 510      :     this message serves to "follow an edge" of the "wait-for" graph.
01A0 511      :
01A0 512      : CALLING SEQUENCE:
01A0 513      :
01A0 514      :     BSBW  LCK$SND_SRCHDLCK
01A0 515      :     Note: This routine may not return to its caller if called
01A0 516      :           without a timestamp assigned (R9=0). In this case,
01A0 517      :           a message requesting a timestamp is sent and the stack
01A0 518      :           is unwound and we exit the deadlock search.
01A0 519      :           Also, if we fail to allocate a CDRP, we also unwind the
01A0 520      :           stack and exit the deadlock search.
01A0 521      :
01A0 522      : INPUT PARAMETERS:
01A0 523      :
01A0 524      :     R6    Address of LKB
01A0 525      :     R9    Address of Message buffer or 0 indicating none
01A0 526      :     R10   Bottom of stack
01A0 527      :
01A0 528      : IMPLICIT INPUTS:
01A0 529      :
01A0 530      :     The region of stack bounded by R10 and SP contains a series
01A0 531      :     of stack frames that describe that current "wait-for" cycle
01A0 532      :     (see description below)
01A0 533      :
01A0 534      : OUTPUT PARAMETERS:
01A0 535      :
01A0 536      :     R0    Completion code:
01A0 537      :           0 = exit normally
01A0 538      :           -1 = exit due to failure to allocate a CDRP; stack
01A0 539      :               has been unwound back to original caller.
01A0 540      :     R9    Address of message buffer if timestamp assigned
01A0 541      :
01A0 542      : SIDE EFFECTS:
01A0 543      :
01A0 544      :     R1 not preserved
01A0 545      : --
01A0 546      :
007C 8F  BB 01A0 547 LCK$SND_SRCHDLCK::
01A0 548      : PUSH  #^M<R2,R3,R4,R5,R6>      ; Can't change this without also
01A4 549      :                               ; changing value of LOCKFRAME and
01A4 550      :                               ; deadlock resolution code
01A4 551      :
01A4 552      : ; Determine if a timestamp has been assigned. R9 = 0 indicates
01A4 553      : ; none was assigned. R9 <> 0 indicates it points
01A4 554      : ; to a message buffer and therefore, a timestamp has been assigned.
01A4 555      :
01A4 556      : TSTL  R9      ; Is there a timestamp assigned?
01A6 557      : BNEQ  10$     ; Yes
FE55 30 01A8 558      : BSBW  LCK$SND_TIMESTAMP_RQST ; No, get one (may not return here)
01AB 559

```

```

02FF 30 01AB 560 10$:
OE 50 E8 01AB 561
52 59 D0 01AE 562
0314 30 01B1 563
SE 5A 04 C3 01B4 564
50 01 CE 01B7 565
05 01 05 01BB 566
01BF 567
01BF 568
01BF 569
01BF 570 15$:
01BF 571
01BF 572
01BF 573
01BF 574
01BF 575
01BF 576
01BF 577
01BF 578
01BF 579
01BF 580
01BF 581
01BF 582
01BF 583
01BF 584
01BF 585
01BF 586
01BF 587
01BF 588
01BF 589
01BF 590
01BF 591
01BF 592
01BF 593
01BF 594
01BF 595
01BF 596
01BF 597
01BF 598
01BF 599
01BF 600
01BF 601
01BF 602
01BF 603
01BF 604
01BF 605
01BF 606
01BF 607
01BF 608
01BF 609
01BF 610
01BF 611
01BF 612
01BF 613
01BF 614
01BF 615
01BF 616

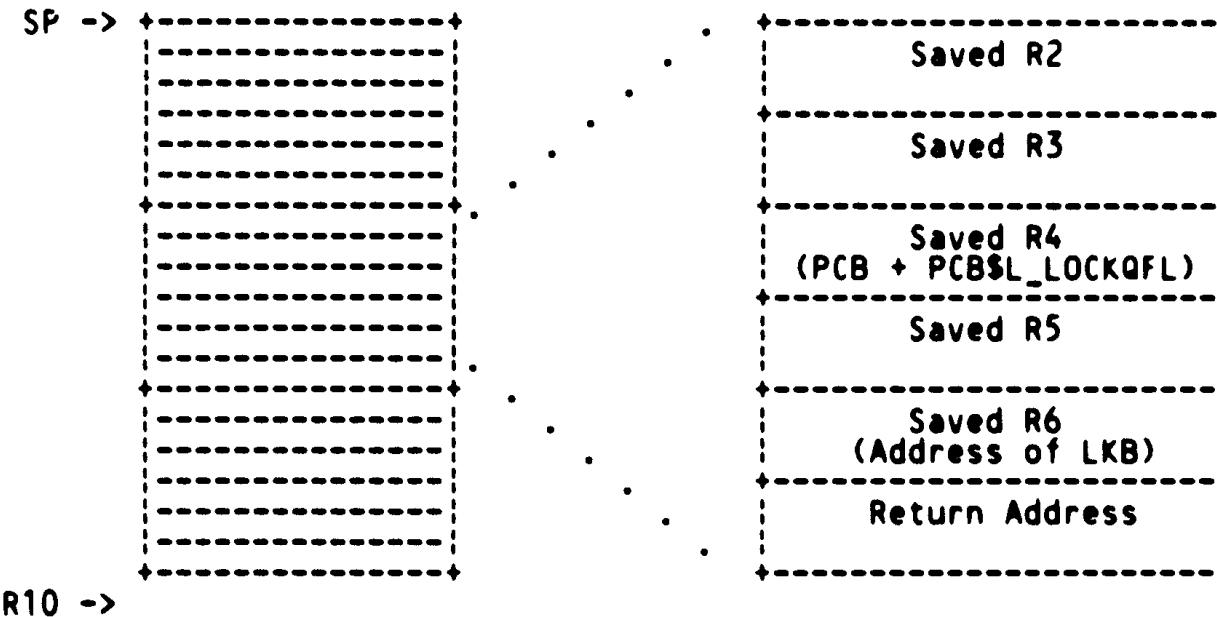
```

```

: Allocate a CDRP
BSBW LCK$ALLOC_LONGCDRP
BLBS R0,15$ : Have one
MOVL R9,R2 : Move address of message buffer
BSBW WAIT FOR POOL
SUBL3 #4,RTO,SP : Unwind stack
MNEGL #1,R0 : Set completion code
RSB : Return to original caller

```

The stack consists of a series of stack frames, one for each lock involved in the current wait-for cycle. Each stack frame consists of the 5 saved registers (R2 - R6) and a return address. Note that in each stack frame the saved R6 points to the lock and the saved R4 points to the respective PCB lock queue (if the lock is not master copy). Only the first and last frames should contain master copy locks.
The stack frames are bounded by R10 and the current SP.
The following diagram shows the stack with three frames.



We will now search the frames looking for the process with the smallest deadlock priority. When found, the respective deadlock priority will be compared with that in the input message. The objective is to find the best candidate for a deadlock victim if a deadlock is later found. This candidate will be included in the message we send to the other system. Note that a deadlock priority of zero causes an immediate exit from the loop. Register usage will be:

```

R0 Current deadlock priority
R1 Current lock frame pointer
R2 Minimum deadlock priority, so far
R3 Best victim frame, so far
R4 Address of PCB lock queue (current frame)

```

			01BF	617	:	R5	Address of CDRP	
			01BF	618	:	R9	Address of message buffer	
			01BF	619	:	R10	Bottom of stack (start search here)	
			01BF	620	:	SP	Top of stack (end search here)	
			01BF	621	:			
			01BF	622	:			
			01BF	623	:			
			01BF	624	:			
			01BF	625	:			
			01BF	626	:			
	24	A9	D5	01BF	626	TSTL	LKMSG\$\$_VCTMPRI(R9)	; Don't bother searching if the priority
		62	13	01C2	627	BEQL	50\$; in the message is zero
				01C4	628			
51	5A	18	C3	01C4	629	SUBL3	#LOCKFRAME,R10,R1	; Initialize current frame pointer
	53	51	DO	01C8	630	MOVL	R1,R3	; Initialize "best" frame pointer
	52	01	CE	01CB	631	MNEGL	#1,R2	; Initialize "best" deadlock priority
50	10	A1	DO	01CE	632	MOVL	16(R1),R0	; Get LKB address
		04	E1	01D2	633	BBC	#LKBSV_MSTCPY,-	; Branch if not master copy
	06	2A	A0	01D4	634		LKBSW_STATUS(R0),25\$	
50	24	A0	DO	01D7	635	MOVL	LKBSL_DLCKPRI(R0),R0	; Get deadlock priority from master copy
		08	11	01DB	636	BRB	28\$	
54	08	A1	DO	01DD	637	MOVL	8(R1),R4	; Get pointer to PCB lock queue
50	08	A4	DO	01E1	638	MOVL	PCBSL_DLCKPRI-PCBSL_LOCKQFL(R4),R0	; Get current deadlock pri.
		12	13	01E5	639	BEQL	35\$; Branch if zero - have best victim
	52	50	D1	01E7	640	CMPL	R0,R2	; Compare current priority with
		03	1E	01EA	641	BGEQU	30\$; previous minimum.
	52	50	7D	01EC	642	MOVQ	R0,R2	; This frame becomes "best so far"
	51	18	C2	01EF	643	SUBL	#LOCKFRAME,R1	; Move to next frame
	5E	51	D1	01F2	644	CMPL	R1,SP	; Reached top of stack yet?
		D7	1E	01F5	645	BGEQU	20\$; No, repeat for next frame
		03	11	01F7	646	BRB	40\$	
	52	50	7D	01F9	647	MOVQ	R0,R2	; Move priority and frame pointer
				01FC	648			
				01FC	649			
				01FC	650			
				01FC	651			
				01FC	652			
24	A9	52	D1	01FC	653	CMPL	R2,LKMSG\$\$_VCTMPRI(R9)	; Compare priorities
		24	1A	0200	654	BGTRU	50\$; The one in the message was lower
				0202	655			
				0202	656			; The one on the stack was lower. R3 points to relevant frame.
				0202	657			
51	10	A3	DO	0202	658	MOVL	16(R3),R1	; Get address of LKB
		04	E1	0206	659	BBC	#LKBSV_MSTCPY,-	; Branch if not master copy
	0A	2A	A1	0208	660		LKBSW_STATUS(R1),45\$	
53	54	A1	DO	020B	661	MOVL	LKBSL_REMLKID(R1),R3	; Get remote lockid
54	58	A1	DO	020F	662	MOVL	LKBSL_CSID(R1),R4	; and CSID
		19	11	0213	663	BRB	60\$	
50	53	30	A1	0215	664	MOVL	LKBSL_LKID(R1),R3	; Get lockid
	00000000	'GF	DO	0219	665	MOVL	G^CLUSGL CLUB,R0	; Get address of CLUB
	54	60	A0	0220	666	MOVL	CLUBS\$_LOCAL_CSID(R0),R4	; Get local CSID
		08	11	0224	667	BRB	60\$	
				0226	668			
				0226	669			; The one in the message was lower.
				0226	670			
52	24	A9	7D	0226	671	MOVQ	LKMSG\$\$_VCTMPRI(R9),R2	; Get victim priority and lockid
54	2C	A9	DO	022A	672	MOVL	LKMSG\$\$_VCTMCSID(R9),R4	; and CSID
				022E	673			

Note that the following code makes a number of assumptions regarding the order of registers saved on the stack and their contents.

```

        J22E 674 60$: ; Store info. in CDRP
        022E 675
44 A5 52 7D 022E 676      MOVQ   R2,CDRPSL_VAL7(R5)      ; Store victim priority and lockid
5C A5 54 DO 0232 677      MOVL   R4,CDRPSL_VAL9(R5)      ; and CSID
        54 A6 DO 0236 678      MOVL   LKBSL_REM[KID(R6),-    ; and next lockid to continue search
        60 A5 0239 679      CDRPSL_VAL10(R5)
        55 DD 023B 680      PUSHL  R5
0C A9 18 28 023D 681      MOVCS  #24,LKMSG$W_MEMSEQ(R9),- ; and other fields
        2C A5 0241 682      CDRPSL_VAL1(R5)
        55 B EDO 0243 683      POPL   R5
        78 AF 9E 0246 684      MOVAB  B^BLD_SRCHDLCK,-      ; Store address of message build routine
        4C A5 0249 685      CDRPSL_MSGBLD(R5)
        024B 686
        024B 687      ; Send the message
        024B 688
        06 2A 04 E1 024B 689      BBC    #LKBSV_MSTCPY,-      ; Branch if not master copy
        53 58 A6 DO 024D 690      LKBSW_STATUS(R6),70$
        08 11 0254 691      BRB    LKBSL_CSID(R6),R3    ; Get CSID
        50 50 A6 DO 0256 692      BRB    75$
53 38 A0 DO 025A 693 70$:  MOVL   LKBSL_RSB(R6),R0    ; Get RSB address
        0219 30 025E 694 70$:  MOVL   RSB$S_CSID(R0),R3    ; Get CSID
        007C 8F BA 0261 695 75$:  BSBW   SEND_DLCK_MSG
        50 D4 0265 696      POPR   #^M<R2,R3,R4,R5,R6>
        05 0267 697      CLRL  R0                    ; Set completion code
        0268 698      RSB
        0268 699      ;
        0268 700      ; Message build routine
        0268 701      ;
        0268 702      ; Inputs: R2 Address of message buffer
        0268 703      ; R5 Address of CDRP
        0268 704      ;
        0268 705      ;
08 A2 0C02 8F B0 0268 706 BLD_REDO_SRCH:
        0E 11 026E 707      MOVW   #LKMSG$K_REDO_SRCH@8- ; Store facility and function codes
        0270 708      !CLMSG$R_FAC_LCK,CLMSG$B_FACILITY(R2)
        0270 709      BRB    BLD_COMMON
        0270 710
08 A2 0902 8F B0 0270 711 BLD_TIMESTAMP_RQST:
        06 11 0276 712      MOVW   #LKMSG$K_TSQRST@8- ; Store facility and function codes
        0278 713      !CLMSG$R_FAC_LCK,CLMSG$B_FACILITY(R2)
        0278 714      BRB    BLD_COMMON
        0278 715
08 A2 0A02 8F B0 0278 716 BLD_SRCHDLCK:
        027E 717      MOVW   #LKMSG$K_SRCHDLCK@8- ; Store facility and function codes
        027E 718      !CLMSG$R_FAC_LCK,CLMSG$B_FACILITY(R2)
        027E 719 BLD_COMMON:
        2C A5 3C BB 027E 720      PUSHR #^M<R2,R3,R4,R5>
        20 28 0280 721      MOVCS #32,CDRPSL_VAL1(R5),- ; Move data from CDRP to message buffer
        0C A2 0284 722      LKMSG$W_MEMSEQ(R2)
        3C BA 0286 723      POPR  #^M<R2,R3,R4,R5>
        5C A5 7D 0288 724      MOVQ  CDRPSL_VAL9(R5),-
        2C A2 028B 725      LKMSG$[_VCTMCSID(R2)
        05 028D 726      RSB
    
```

```

028E 728      .SBTTL LCK$RCV_SRCHDLCK - Receive search deadlock message
028E 729
028E 730      :++
028E 731      : FUNCTIONAL DESCRIPTION:
028E 732      :
028E 733      : This routine is called when we receive a deadlock search message
028E 734      : for either a lock mastered on this system or a waiting lock owned
028E 735      : by this system. We continue searching using this lock as our
028E 736      : starting point.
028E 737
028E 738      : CALLING SEQUENCE:
028E 739
028E 740      : BSBW LCK$RCV_SRCHDLCK (called from input message dispatcher)
028E 741
028E 742      : INPUT PARAMETERS:
028E 743
028E 744      : R2 Address of message buffer
028E 745      : R3 Address of CSB
028E 746
028E 747      : OUTPUT PARAMETERS:
028E 748
028E 749      : None
028E 750
028E 751      : SIDE EFFECTS:
028E 752
028E 753      : Other deadlock search messages may be sent to other systems.
028E 754      : R0 and R1 are not preserved.
028E 755      :--
028E 756
028E 757 LCK$RCV_SRCHDLCK::
028E 758 BSBW RCV_DLCK_MSG
028E 759 PUSHR #*MZR2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
028E 760 MOVL R2,R9 ; Move address of input message
028E 761
028E 762 ; Get lockid of lock to start search with. Convert to LKB address.
028E 763
028E 764 MOVL LKMSG$SL_NEXTLKID(R2),R4 ; Get lockid
028E 765 BSBW LCK$CVT_ID_TO_LKB ; Convert to LKB address
028E 766 BLBC R0,10$ ; No LKB found; ignore message
028E 767
028E 768 ; Check bitmap expiration timestamp before we start using bitmap
028E 769
028E 770 BSBW CHECK_TIMESTAMP ; Returns address of bitmap in R7
028E 771 BLBS R0,20$ ; We can use bitmap
028E 772
028E 773 ; Bitmap has been preempted by a later deadlock search.
028E 774 ; Double the bitmap lifetime requested and send back a message to
028E 775 ; redo the original deadlock search.
028E 776
028E 777 INCB LKMSG$B_TSLT(R2) ; This will double the lifetime
028E 778 BSBW LCK$SND_REDO_SRCH
028E 779 BRW 70$
028E 780
028E 781 10$: ; Set up registers
028E 782
028E 783 20$:
028E 784 MOVL LKMSG$SL_ORIGEPID(R2),R8 ; Original EPID
028E 785 MOVL SP,R10 ; Current stack position

```

01FB 30
OFFC 8F BB
59 52 DO

54 30 A2 DO
01B5 30
OC 50 E9

FEB1 30
09 50 E8

0E A2 96
010D 30
008E 31

58 10 A2 DO
SA SE DO

```

5B 00000000'GF C1 02B8 785 ADDL3 G^LCK$GL_EXTRASTK,- ; Compute stack limit
00000000'GF C0 02BE 786 G^EXE$GL-INTSTKLM,R11
5B 18 C0 02C4 787 ADDL #LOCKFRAME,R11
52 0C A6 3C 02C7 788 MOVZWL LKB$S_PID(R6),R2 ; Get process index
0B 13 02CB 789 BEQL 25$ ; Branch if master copy or system owned
51 00000000'GF D0 02CD 790 MOVL G^SCH$GL_PCBVEC,R1 ; Convert to PCB address
54 6142 D0 02D4 791 MOVL (R1)[R2],R4
02D8 792
02D8 793 25$: ; The way in which we resume the deadlock search depends on
02D8 794 ; whether this lock is a master (or local) or process copy.
02D8 795
50 50 A6 D0 02D8 796 MOVL LKB$S_RSB(R6),R0 ; Get RSB address
38 A0 D5 02DC 797 TSTL RSB$S_CSID(R0) ; Is lock mastered here?
53 13 02DF 798 BEQL 60$ ; Yes
02E1 799
02E1 800 ; This is a process copy lock. For each lock this process
02E1 801 ; has in either CONVERT or WAITING state, see who is blocking
02E1 802 ; those locks.
02E1 803
55 56 D0 02E1 804 MOVL R6,R5 ; Move LKB address
52 D5 02E4 805 TSTL R2 ; If process index is 0 then lock is
57 13 02E6 806 BEQL 70$ ; system owned
53 67 52 E2 02E8 807 BBSS R2,(R7),70$ ; Br. if we've already done this process
54 0104 C4 DE 02EC 808 MOVAL PCB$S_LOCKQFL(R4),R4 ; Point to lock queue header
56 04 A4 D0 02F1 809 MOVL 4(R4),R6 ; Get last lock in list
54 56 D1 02F5 810 30$: ; Reached end of list?
45 13 02F8 811 BEQL 70$ ; Yes
56 C0 A6 DE 02FA 812 MOVAL -LKB$S_OWNOFL(R6),R6 ; Point to start of LKB
56 55 D1 02FE 813 CMPL R5,R6 ; Is this the one we have in R5?
2B 13 0301 814 BEQL 35$ ; Yes, move on to next one
0303 815 DISPATCH LKB$S_STATE(R6),TYPE=B,PREFIX=LKB$K,-
0303 816 <-
0303 817 <CONVERT,32$>,-
0303 818 <WAITING,32$>-
0303 819 >
30 11 030D 820 BRB 70$ ; Exit for all other states
09 E0 030F 821 32$: BBS #LCK$V_NODLCKWT,- ; Branch if this lock should not be
1A 28 A6 0311 822 ; considered as waiting for other locks
50 50 A6 D0 0314 823 MOVL LKB$S_RSB(R6),R0 ; Get RSB for this lock
38 A0 D5 0318 824 TSTL RSB$S_CSID(R0) ; Is it managed elsewhere?
08 13 031B 825 BEQL 33$ ; No
000001A0'GF 16 031D 826 JSB G^LCK$SND_SRCHDLCK ; Yes, send a message to keep looking
06 11 0323 827 BRB 34$ ; Continue on this PCB
00000000'GF 16 0325 828 33$: JSB G^LCK$SRCH_RESDLCK ; No, recursively search
11 50 E8 032B 829 34$: BLBS R0,70$ ; If LBS, exit search
56 44 A6 D0 032E 830 35$: MOVL LKB$S_OWNOBL(R6),R6 ; Get previous lock
C1 11 0332 831 BRB 30$ ; Repeat
0334 832
0334 833 60$: ; This lock is a local or master copy. Just determine who is blocking
0334 834 ; this lock after verifying that the lock is not granted.
0334 835
0334 836 ASSUME LKB$K_GRANTED GT 0
36 A6 95 0334 837 TSTB LKB$S_STATE(R6) ; Ignore message if lock is granted
06 14 0337 838 BGTR 70$
00000000'GF 16 0339 839 JSB G^LCK$SRCH_RESDLCK ; Search for deadlock
033F 840
033F 841 70$: ; Deallocate the original message buffer and exit
  
```

DSTRDLCK
V04-000

OFFC 8F	BA	033F	842	POPR	#^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
0159	31	0343	844	BRW	DEALL_DLCK_MSG ; Deallocate message buffer and return

DST
V04

```

0346 846          .SBTTL LCK$SND_DLCKFND - Send deadlock found message
0346 847
0346 848 :++
0346 849 : FUNCTIONAL DESCRIPTION:
0346 850 :
0346 851 :     This routine sends a message informing another system that a
0346 852 :     specified lock has been chosen as a deadlock victim. The lock
0346 853 :     on the destination system is either a local copy or a process
0346 854 :     because it is on that system the the dequeue/cancel function
0346 855 :     must be issued.
0346 856 :
0346 857 : CALLING SEQUENCE:
0346 858 :
0346 859 :     JSB     LCK$SND_DLCKFND
0346 860 :     IPL must be a IPL$SCS
0346 861 :     Note:   If we don't have an input message and we are unable
0346 862 :             to allocate a CDRP, then we will unwind the stack
0346 863 :             and exit from deadlock detection.
0346 864 :
0346 865 : INPUT PARAMETERS:
0346 866 :
0346 867 :     R2     Lock id. of victim lock
0346 868 :     R3     CSID of destination system
0346 869 :     R9     Address of input message or 0 indicating no message
0346 870 :     R10    Bottom of stack
0346 871 :
0346 872 : OUTPUT PARAMETERS:
0346 873 :
0346 874 :     R0     Completion code of -1 if we unwind the stack and exit
0346 875 :             from deadlock searching.
0346 876 :
0346 877 : SIDE EFFECTS:
0346 878 :
0346 879 :     R0 - R5 not preserved
0346 880 : --
0346 881
0346 882 LCK$SND_DLCKFND::
0346 883
0346 884         BSBW  LCK$ALLOC_LONGCDRP      ; Alloc. CDRP
0346 885         BLBC  R0,80$                 ; Unable to allocate one
0346 886
0346 887         ; Store necessary information to build message in CDRP.
0346 888         ; The original lockid and CSID are stored only if we have
0346 889         ; an input message. This will be used by the destination system
0346 890         ; to redo a deadlock search. If we don't have an input message
0346 891         ; then this system will automatically redo the search.
0346 892
0346 893         MOVL  G^CLUSGL CLUB,R0
0346 894         MOVW  CLUB$W_MEMSEQ(R0),-      ; Store memseq.
0346 895         CDRP$L_VAL1(R5)
0346 896
0346 897         TSTL  R9                      ; Is there an input message?
0346 898         BEQL  10$                     ; No
0346 899         MOVQ  LKMSG$L_ORIGLKID(R9),-   ; Yes, store original lockid and
0346 900         CDRP$L_VAL3(R5)               ; CSID in CDRP
0346 901         MOVB  LKMSG$B_TSLT(R9),-      ; and timestamp lifetime
0346 902         CDRP$L_VAL1+2(R5)
0346 903         BRB  20$

```

```

0164 30
2D 50 E9
50 00000000'GF D0
00AC C0 B0
2C A5
59 D5
0C 13
14 A9 7D
34 A5
0E A9 90
2E A5
03 11

```



```

3C A5 34 A5 7C 0369 903 10$: CLRQ CDRP$L_VAL3(R5) ; Indicate no original message
      52 7D 036C 904 20$: MOVQ R2,CDRP$L_VAL5(R5) ; Store victim lockid and CSID
      BA AF 9E 0370 905 MOVAB B^BLD_DLCKFND,- ; Store address of message build routine
      4C A5 0373 906 CDRP$E_MSGBLD(R5)
      0102 30 0375 907 BSBW SEND_DECK_MSG ; Send the message
      05 0378 908 RSB
      0379 909
52 59 D0 0379 910 80$: MOVL R9,R2 ; Move address of message
      04 13 037C 911 BEQL 90$ ; No message
      014A 30 037E 912 BSBW WAIT_FOR_POOL
      05 0381 913 RSB
5E 50 01 CE 0382 914 90$: MNEGL #1,R0 ; Set completion status
      5A 04 C1 0385 915 ADDL3 #4,R10,SP ; Unwind stack
      05 0389 917 RSB
      038A 918
      038A 919
      038A 920 ; Action routine to build deadlock found message. Inputs are:
      038A 921 :
      038A 922 : R2 Address of message buffer
      038A 923 : R5 Address of CDRP
      038A 924 : CDRP$L_VAL1 MEMSEQ and timestamp lifetime
      038A 925 : CDRP$L_VAL3 Original lockid (or 0)
      038A 926 : CDRP$L_VAL4 Original CSID (or 0)
      038A 927 : CDRP$L_VAL5 Lockid of victim lock
      038A 928 : CDRP$L_VAL6 CSID of victim lock
      038A 929 :
      038A 930
      038A 931 BLD_DLCKFND:
      038A 932 ASSUME CLSMG$B_FUNC EQ 1+CLSMG$B_FACILITY
      08 A2 0B02 8F B0 038A 933 MOVW #LKMSG$K_DLCKFND@8- ; Store function and facility codes
      0390 934 !CLSMG$R_FAC_LCK,CLSMG$B_FACILITY(R2)
      2C A5 D0 0390 935 MOVL CDRP$L_VAL1(R5),- ; Store MEMSEQ and timestamp lifetime
      0C A2 0393 936 LKMSG$Q MEMSEQ(R2)
      34 A5 7D 0395 937 MOVQ CDRP$L_VAL3(R5),- ; Store original lockid and CSID
      14 A2 0398 938 LKMSG$E_ORIGLKID(R2)
      3C A5 7D 039A 939 MOVQ CDRP$L_VAL5(R5),- ; Store victim lockid and CSID
      28 A2 039D 940 LKMSG$E_VCTMLKID(R2)
      05 039F 941 RSB

```



```

038B 981      .SBTTL LCK$$ND_REDO_SRCH - Send a redo deadlock search message
038B 982
038B 983
038B 984 :++
038B 985 : FUNCTIONAL DESCRIPTION:
038B 986 :
038B 987 : This routine is called when it is necessary to redo a deadlock
038B 988 : search. It sends a message to the system mastering the lock
038B 989 : unless that system is this system.
038B 990 : CALLING SEQUENCE:
038B 991 :
038B 992 :     BSBW  LCK$$ND_REDO_SRCH
038B 993 :
038B 994 : INPUT PARAMETERS:
038B 995 :
038B 996 :     R2    Address of message buffer
038B 997 :
038B 998 : IMPLICIT INPUTS:
038B 999 :
038B 1000 :     LKMSG$B_TSLT(R2)    Timestamp lifetime
038B 1001 :     LKMSG$L_ORIGLKID(R2) Lockid of lock to repeat deadlock search
038B 1002 :     LKMSG$L_ORIGCSID(R2) CSID of system mastering above lock
038B 1003 :                          (0 indicates original CSID and lockid
038B 1004 :                          are unknown, but it is not necessary
038B 1005 :                          to send the redo message - see LCK$$ND_DLCKFND)
038B 1006 :
038B 1007 : OUTPUT PARAMETERS:
038B 1008 :
038B 1009 :     None
038B 1010 :
038B 1011 : SIDE EFFECTS:
038B 1012 :
038B 1013 :     R0 - R5 are not preserved
038B 1014 :--
038B 1015
038B 1016 LCK$$ND_REDO_SRCH::
038B 1017 : Determine if the lock is mastered on this system
038B 1018
53 18 A2 D0 038B 1019 MOVL LKMSG$L_ORIGCSID(R2),R3 ; Get original CSID
10 13 038B 1020 BEQL 10$ ; Not present
50 00000000'GF D0 03C1 1021 MOVL G^CLUGL CLUB,R0 ; Get address of CLUB
60 A0 53 D1 03C8 1022 CMPL R3,CLUB$C_LOCAL_CSID(R0); Is it the CSID of this system?
04 12 03CC 1023 BNEQ 20$ ; No
03CE 1024
03CE 1025 : Lock is mastered on this system
03CE 1026
0027 30 03CE 1027 BSBW REDO_SRCH ; Requeue it to the timeout queue
05 03D1 1028 10$: RSB
03D2 1029
03D2 1030 20$: ; Lock is mastered elsewhere
03D2 1031
00D8 30 03D2 1032 BSBW LCK$$LOC_LONGCDRP ; Allocate CDRP
14 50 E9 03D5 1033 BLBC R0,80$ ; Unable to allocate one
0C A2 D0 03D8 1034 MOVL LKMSG$W_MEMSEQ(R2),- ; Store MEMSEQ and timestamp lifetime
2C A5 03DB 1035 CDRP$L_VAL1(R5)
14 A2 D0 03DD 1036 MOVL LKMSG$L_ORIGLKID(R2),- ; Store lockid
34 A5 03E0 1037 CDRP$L_VAL3(R5)

```

DS
SY
RS
SC
SEI
TSI
WA
PS
--
SA
SS
SS
Ph
--
In
Col
Pa
Syl
Pa
Syl
Psi
Cri
As
Th
99
Th
14
28
Ma
--
\$
-\$
-\$
TO
17
Th
MA

DSTRDLCK
V04-000

N 14

- DISTRIBUTED DEADLOCK DETECTION AND RES 16-SEP-1984 00:35:31 VAX/VMS Macro V04-00 Page 25
LCK\$SND_REDO_SRCH - Send a redo deadlock 5-SEP-1984 04:09:19 [SYSLOA.SRC]DSTRDLCK.MAR;1 (11)

FE82 CF	9E	03E2	1038	MOVAB	W^BLD_REDO_SRCH,-	; Store address of message build routine
4C AS		03E6	1039		CDRPS[C MSGBLD(R5)	
008F	30	03E8	1040	BSBW	SEND_DECK_MSG	; Send the message
	05	03EB	1041	RSB		
		03EC	1042			
00DC	30	03EC	1043	BSBW	WAIT_FOR_POOL	
	05	03EF	1044	RSB		

**

```

03FO 1046      .SBTTL LCK$RCV_REDO_SRCH
03FO 1047
03FO 1048      :++
03FO 1049      : FUNCTIONAL DESCRIPTION:
03FO 1050      :
03FO 1051      :     This routine is called when we receive a message to redo a deadlock
03FO 1052      :     search for a lock mastered on this system.
03FO 1053      :
03FO 1054      : CALLING SEQUENCE:
03FO 1055      :
03FO 1056      :     BSBW  LCK$RCV_REDO_SRCH (called from input message dispatcher)
03FO 1057      :
03FO 1058      : INPUT PARAMETERS:
03FO 1059      :
03FO 1060      :     R2    Address of message buffer
03FO 1061      :     R3    Address of CSB
03FO 1062      :
03FO 1063      : OUTPUT PARAMETERS:
03FO 1064      :
03FO 1065      :     None
03FO 1066      :
03FO 1067      : SIDE EFFECTS:
03FO 1068      :
03FO 1069      :     R0, R1 and R4 not preserved
03FO 1070      : --
03FO 1071      :
03FO 1072      LCK$RCV_REDO_SRCH::
0099 30 03FO 1073      BSBW  RCV_DLCK_MSG
03    10 03F3 1074      BSBB  REDO_SRCH      ; Do the work
00A7 31 03F5 1075      BRW   DEALL_DLCK_MSG ; Deallocate message buffer and return

```

```

03F8 1077      .SBTTL REDO_SRCH - Redo deadlock search
03F8 1078
03F8 1079      :++
03F8 1080      : FUNCTIONAL DESCRIPTION:
03F8 1081      :
03F8 1082      : This routine is called to requeue a lock back on the timeout
03F8 1083      : queue when a deadlock search must be repeated.  Deadlock searches
03F8 1084      : are repeated for reasons such as:
03F8 1085      :
03F8 1086      :     o One deadlock has already been found for this lock
03F8 1087      :     o A timestamp could not be issued
03F8 1088      :     o The deadlock search was incomplete for some reason
03F8 1089      :       (e.g. unable to allocate pool or our timestamp was
03F8 1090      :         superseded)
03F8 1091
03F8 1092      : CALLING SEQUENCE:
03F8 1093      :
03F8 1094      :     BSBW  REDO_SRCH
03F8 1095
03F8 1096      : INPUT PARAMETERS:
03F8 1097      :
03F8 1098      :     R2    Address of message buffer
03F8 1099
03F8 1100      : IMPLICIT INPUTS:
03F8 1101      :
03F8 1102      :     LKMSG$B_TSLT(R2)    Timestamp lifetime
03F8 1103      :     LKMSG$L_ORIGLKID(R2) Lockid of lock to repeat deadlock search
03F8 1104
03F8 1105      : OUTPUT PARAMETERS:
03F8 1106      :
03F8 1107      :     None
03F8 1108
03F8 1109      : SIDE EFFECTS:
03F8 1110      :
03F8 1111      :     R0, R1 and R4 not preserved
03F8 1112      :--
03F8 1113
03F8 1114      REDO_SRCH:
03F8 1115      PUSHL  R6                ; Save R6
03F8 1116      MOVL  LKMSG$L_ORIGLKID(R2),R4 ; Get lockid
03F8 1117      BSBW  LCK$CVT_ID_TO_LKB      ; Convert to LKB address
03F8 1118      BLBC  R0,60$              ; No LKB
03F8 1119      MOVL  LKB$L_RSB(R6),R0      ; Get RSB address
03F8 1120      TSTL  RSB$L_CSID(R0)        ; Verify it's mastered here
03F8 1121      BNEQ  90$                ; Error!
03F8 1122
03F8 1123      ; Only requeue lock if it's in either CONVERT or WAITING state
03F8 1124      ; and it's not already queued.
03F8 1125
03F8 1126      DISPATCH  LKB$B_STATE(R6),TYPE=B,PREFIX=LKB$K_,-
03F8 1127      <-
03F8 1128      <CONVERT,30$>,-
03F8 1129      <WAITING,30$>,-
03F8 1130      >
03F8 1131      BRB   60$                ; Ignore for other states
03F8 1132
03F8 1133      30$: ; If we haven't used up all the retries (MAX_TSLT) then the lock

```

```

54  56  DD  03F8 1115
    14 A2 DD  03FA 1116
    0053 30 03FE 1117
50  48 50  E9  0401 1118
    50 A6 D0  0404 1119
    38 A0 D5  0408 1120
    43 12  040B 1121
                    040D 1122
                    040D 1123
                    040D 1124
                    040D 1125
                    040D 1126
                    040D 1127
                    040D 1128
                    040D 1129
                    040D 1130
    33 11  0417 1131
                    0419 1132
                    0419 1133

```

```

0419 1134 ; is requested at the front of the timeout queue. If we have
0419 1135 ; used up all retries, then the retry count is cleared, and the
0419 1136 ; lock is queued at the back of the timeout queue with another
0419 1137 ; wait time applied. The result is to retry a deadlock search for
0419 1138 ; this lock later.
0419 1139
2E 2A 06 E2 0419 1140 BBSS #LKBSV TIMEOUT,- ; Branch if already on the queue;
00000000 GF D0 041B 1141 LKBSW STATUS(R6),60$ ; set bit otherwise
18 A6 041E 1142 MOVL G^EXE$GL ABSTIM,- ; Store immediate timeout time
50 00000000 GF DE 0424 1143 LKBSL DUETIME(R6)
OE A2 90 0426 1144 MOVAL G^LCK$GL TIMEOUT,R0 ; Get address of timeout queue
4E A6 042D 1145 MOVB LKMSG$B_TSLT(R2),- ; Store timestamp lifetime
4E A6 91 0430 1146 LKBSB_TSLT(R6)
05 0432 1147 CMPB LKBSB_TSLT(R6),- ; Have we exceeded the maximum
05 14 0435 1148 #MAX_TSLT ; number of retries?
60 66 OE 0436 1149 BGTR 40$ ; Yes
OF 11 0438 1150 INSQUE LKBSL_ASTQFL(R6),(R0) ; No, insert at the head of the queue
043D 1151 BRB 60$
4E A6 94 043D 1152 40$: CLRB LKBSB_TSLT(R6) ; Reset retry count
00000000 GF C0 0440 1154 ADDL G^LCK$GL WAITTIME,- ; Add another wait time to due time
18 A6 0446 1155 LKBSL_DUETIME(R6)
04 B0 66 OE 0448 1156 INSQUE LKBSL_ASTQFL(R6),a4(R0) ; Insert at the tail of the queue
044C 1157
56 8ED0 044C 1158 60$: POPL R6
05 044F 1159 RSB
0450 1160
0450 1161 90$: BUG_CHECK LOCKMGRERR,FATAL; Not mastered here

```

0454 1163 .SBTTL LCK\$CVT_ID_TO_LKB - Convert a lockid to LKB address

0454 1164
0454 1165 :++
0454 1166 : FUNCTIONAL DESCRIPTION:
0454 1167 :
0454 1168 : This routine converts a lockid to a LKB address, if possible.
0454 1169 : The verification check of comparing remote lockids is not
0454 1170 : performed as both lockids are not available.
0454 1171 :
0454 1172 : CALLING SEQUENCE:
0454 1173 :
0454 1174 : BSBW LCK\$CVT_ID_TO_LKB
0454 1175 :
0454 1176 : INPUT PARAMETERS:
0454 1177 :
0454 1178 : R4 Lockid
0454 1179 :
0454 1180 : OUTPUT PARAMETERS:
0454 1181 :
0454 1182 : R0 Completion code (0 = failure; 1 = success)
0454 1183 : R6 Address of LKB (success only)
0454 1184 :
0454 1185 : SIDE EFFECTS:
0454 1186 :
0454 1187 : None
0454 1188 :--
0454 1189 :
0454 1190 :

0454 1191	LCK\$CVT_ID_TO_LKB:				
0454 1192	MOVZWL	R4,R6			: Put lockid index in R6
0457 1193	CMPL	R6,G^LCK\$GL_MAXID			: Is the lock id too big?
045E 1194	BGTRU	60\$: Yes
0460 1195	MOVL	G^LCK\$GL_IDTBL,R0			: Get address of lockid table
0467 1196	MOVL	(R0)[R6],R6			: Get LKB address
0468 1197	BGEQ	60\$: Unallocated id
046D 1198	CMPL	R4,LKBSL_LKID(R6)			: Check sequence number
0471 1199	BNEQ	60\$: Not valid
0473 1200	MOVL	#1,R0			
0476 1201	RSB				
0477 1202					
0477 1203	60\$: CLRL	R0			
0479 1204	RSB				


```

047A 1206      .SBTTL SEND_DLCK_MSG - Send any deadlock detection message
047A 1207
047A 1208      :++
047A 1209      : FUNCTIONAL DESCRIPTION:
047A 1210      :
047A 1211      : This routine is called to send any message when the caller
047A 1212      : wants control returned to it as opposed to it's caller.
047A 1213      : After the message has been acknowledged, the CDRP is deallocated.
047A 1214      : Note that all errors are ignored.
047A 1215
047A 1216      : CALLING SEQUENCE:
047A 1217      :
047A 1218      : BSBW SEND_DLCK_MSG
047A 1219
047A 1220      : INPUT PARAMETERS:
047A 1221      :
047A 1222      : R3 CSID of destination system
047A 1223      : R5 Address of CDRP
047A 1224
047A 1225      : OUTPUT PARAMETERS:
047A 1226      :
047A 1227      : None
047A 1228
047A 1229      : SIDE EFFECTS:
047A 1230      :
047A 1231      : R0 - R2 and R4 are destroyed.
047A 1232      :--
047A 1233
047A 1234      SEND_DLCK_MSG:
047A 1235      .IF NE CAS MEASURE
047A 1236      INCL G^PMSSGL_DLCKMSGS_OUT
0480 1237      .ENDC
0480 1238
0480 1239      BSBW CNX$SEND_MSG
0483 1240      MOVL R5,R0 : Address of CDRP
0486 1241      JMP G^EXE$DEANONPAGED : Deallocate it and return

```

00000002
00000000'GF D6

FB7D' 30
50 55 D0
00000000'GF 17

```

048C 1243      .SBTTL RCV_DLCK_MSG - Receive a deadlock message
048C 1244
048C 1245      :++
048C 1246      : FUNCTIONAL DESCRIPTION
048C 1247      :
048C 1248      : This routine is called whenever we receive a deadlock message.
048C 1249      : Its purpose is to verify that the internal message buffer
048C 1250      : is available (i.e. not in use waiting for pool). If it is
048C 1251      : in use, then we reject this message and break the connection.
048C 1252
048C 1253      : CALLING SEQUENCE:
048C 1254
048C 1255      : BSBW RCV_DLCK_MSG
048C 1256      : NOTE: If we break the connection, then we return to our caller's
048C 1257      : caller, usually, the input message dispatcher.
048C 1258
048C 1259      : INPUT PARAMETERS:
048C 1260
048C 1261      : R2 Address of message buffer
048C 1262      : R3 Address of CSB
048C 1263
048C 1264      : OUTPUT PARAMETERS:
048C 1265
048C 1266      : None
048C 1267
048C 1268      : SIDE EFFECTS:
048C 1269
048C 1270      : None if we return to our caller.
048C 1271      : The message buffer is deallocated if we break the connection.
048C 1272      :--
048C 1273
048C 1274 RCV_DLCK_MSG:
0000'CF  D5 048C 1275 TSTL W^LKMSG_FKB ; Is fork block in use?
07 12 0490 1276 BNEQ 10$ ; Yes
0492 1277
00000002 0492 1278 .IF NE CAS MEASURE
00000000'GF D6 0492 1279 INCL G^PMSSGL_DLCKMSGS_IN
0498 1280 .ENDC
0498 1281
05 0498 1282 RSB
0499 1283
SE 04 C0 0499 1284 10$: ADDL #4,SP ; Pop caller's return address off stack
FB61' 31 049C 1285 BRW CNX$RCV_REJECT ; Reject message

```

```

049F 1287      .SBTTL  DEALL_DLCK_MSG - Deallocate deadlock message buffer
049F 1288
049F 1289      :++
049F 1290      : FUNCTIONAL DESCRIPTION:
049F 1291      :
049F 1292      :   This routine is called to deallocate received deadlock message
049F 1293      :   buffers. However, it distinguishes between real message buffers
049F 1294      :   and our internal buffer which is not deallocated.
049F 1295      :
049F 1296      : CALLING SEQUENCE:
049F 1297      :
049F 1298      :   BSBW  DEALL_DLCK_MSG
049F 1299      :
049F 1300      : INPUT PARAMETERS:
049F 1301      :
049F 1302      :   R2    Address of message buffer
049F 1303      :   R3    Address of CSB
049F 1304      :
049F 1305      : OUTPUT PARAMETERS:
049F 1306      :
049F 1307      :   None
049F 1308      :
049F 1309      : SIDE EFFECTS:
049F 1310      :
049F 1311      :   R0 - R2 destroyed
049F 1312      :--
049F 1313

```

```

50 0018'CF 9E 049F 1314 DEALL_DLCK_MSG:
    52 50  D1 04A4 1315      MOVAB  W^LKMSG_BFR,R0      ; Get address of internal buffer
    03 13 04A7 1316      CMPL   R0,R2          ; Is it our internal message buffer?
    FB54' 30 04A9 1317      BEQL   10$           ; Yes
    05 04AC 1318      BSBW   CNX$DEALL_MSG_BUF_CSBB ; No, deallocate real message buffer
    10$:  RSB

```

```

04AD 1321      .SBTTL LCK$ALLOC_LONGCDRP - Allocate a long CDRP
04AD 1322
04AD 1323      :++
04AD 1324      : FUNCTIONAL DESCRIPTION:
04AD 1325      :
04AD 1326      : This routine is used to allocate a longer CDRP than is normally
04AD 1327      : used for connection purposes. The reason is because deadlock
04AD 1328      : messages have more context than can fit into a regular sized
04AD 1329      : CDRP.
04AD 1330      :
04AD 1331      : CALLING SEQUENCE:
04AD 1332      :
04AD 1333      : BSBW LCK$ALLOC_LONGCDRP
04AD 1334      :
04AD 1335      : INPUT PARAMETERS:
04AD 1336      :
04AD 1337      : None
04AD 1338      :
04AD 1339      : OUTPUT PARAMETERS:
04AD 1340      :
04AD 1341      : R0 Completion code
04AD 1342      : R5 Address of CDRP
04AD 1343      :
04AD 1344      : COMPLETION CODES:
04AD 1345      :
04AD 1346      : SSS_NORMAL CDRP allocated
04AD 1347      : SSS_INSFEM Insufficient memory
04AD 1348      :
04AD 1349      : SIDE EFFECTS:
04AD 1350      :
04AD 1351      : R0 and R1 not preserved.
04AD 1352      :--
04AD 1353
04AD 1354 LCK$ALLOC_LONGCDRP:
04AD 1355      POSHL R2
51 0064 8F 3C 04AF 1356      MOVZWL #CDRPSK_CM_LONG_LENGTH,R1; Size of CDRP
00000000'GF 16 04B4 1357      JSB G*EXESA[ONONPAGED] ; Allocate pool
0A 50 E9 04BA 1358      BLBC R0,80$ ; Insufficient memory
55 52 D0 04BD 1359      MOVL R2,R5 ; Move address of CDRP
08 A5 51 B0 04C0 1360      MOVW R1,CDRPSW_CDRPSIZE(R5) ; Store size
FB39' 30 04C4 1361      BSBW CNX$INIT_CDRP ; Initialize CDRP
52 8ED0 04C7 1362      POPL R2
05 04CA 1363      RSB

```

04CB 1365 .SBTTL WAIT_FOR_POOL - Wait for pool

04CB 1366
04CB 1367
04CB 1368
04CB 1369
04CB 1370
04CB 1371
04CB 1372
04CB 1373
04CB 1374
04CB 1375
04CB 1376
04CB 1377
04CB 1378
04CB 1379
04CB 1380
04CB 1381
04CB 1382
04CB 1383
04CB 1384
04CB 1385
04CB 1386
04CB 1387
04CB 1388
04CB 1389
04CB 1390
04CB 1391
04CB 1392
04CB 1393
04CB 1394
04CB 1395
04CB 1396
04CB 1397
04CB 1398
04CB 1399

```

:++
: FUNCTIONAL DESCRIPTION:
:
: This routine copies an input message into the internal
: message buffer and does a FORK_WAIT. Upon resumption,
: we simulate receiving the same message again. If another
: input message is received while the fork block is queued,
: then we reject the message and break the connection (see
: RCV_DLCK_MSG).
:
: CALLING SEQUENCE:
:
: BSBW WAIT_FOR_POOL
: NOTE: If no input message is specified, then we unwind the
: stack and return to LCK$DLCKEXIT.
:
: INPUT PARAMETERS:
:
: R2 Address of input message (or 0 indicating no input message)
: R10 Address of stack position to unwind to if R2=0
:
: OUTPUT PARAMETERS:
:
: None
:
: SIDE EFFECTS:
:
: R0 and R1 not preserved.
: A fork block is queued that when resumed will call the input
: message dispatcher.
:--
    
```

```

WAIT_FOR_POOL:
52 D5 04CB 1400 TSTL R2 ; Do we have an input message?
09 12 04CD 1401 BNEQ 10$ ; Yes
SE SA D0 04CF 1402 MOVL R10,SP ; No, unwind stack
00000000'GF 17 04D2 1403 JMP G^LCK$DLCKEXIT ; Exit deadlock detection
04D8 1404
56 007C 8F BB 04D8 1405 10$: PUSH R2,R3,R4,R5,R6 ; Save registers
0000'CF 9E 04DC 1406 MOVAB W^LKMSG_FKB,R6 ; Get address of fork block
66 D5 04E1 1407 TSTL (R6) ; Verify it's not in use
33 12 04E3 1408 BNEQ 90$ ; Error
55 18 A6 9E 04E5 1409 MOVAB FKB$K_LENGTH(R6),R5 ; Get address of message buffer
65 62 34 28 04E9 1410 MOVC3 #LKMSG$K_DLM_LENGTH,(R2),(R5) ; Copy message
55 56 D0 04ED 1411 MOVL R6,R5 ; Move fork block address
007C 8F BA 04F2 1413 POPR #^M<R2,R3,R4,R5,R6> ; Queue fork block
05 04F6 1414 RSB
04F7 1415
04F7 1416 50$: FORK_WAIT ; Fork and wait
65 7C 04FD 1417 CLRQ (R5) ; Indicate fork block is not in use
52 18 A5 9E 04FF 1418 MOVAB FKB$K_LENGTH(R5),R2 ; Get address of message buffer
53 D4 0503 1419 CLRL R3 ; Indicate no CSB address
50 00000000'GF D0 0505 1420 MOVL G^CLUS$GL_CLUB,R0 ; Get address of CLUB
00AC C0 B1 050C 1421 CMPW CLUB$W_MEMSEQ(R0),- ; Has memseq changed?
    
```

DSTRDLCK
V04-000

- DISTRIBUTED DEADLOCK DETECTION AND RES 16-SEP-1984 00:35:31 VAX/VMS Macro V04-00
WAIT_FOR_POOL - Wait for pool 5-SEP-1984 04:09:19 [SYSLOA.SRC]DSTRDLCK.MAR;1

Page 35
(19)

DS
VO

```

OC A2      0510 1422      LKMSG$W_MEMSEQ(R2)
  03      12 0512 1423      60$      : Yes, ignore this message
FAE9'     30 0514 1424      BNEQ      : No, dispatch on this message
         05 0517 1425 60$:  BSBW      LCK$DISPATCH
         0518 1426      RSB
         0518 1427 90$:  BUG_CHECK      LOCKMGRERR,FATAL
         051C 1428
         051C 1429
         051C 1430
         051C 1431
         051C 1432      .END
```

```

SSBASE = FFFFFFFF
SSDISPL = 00000001
SSGENSW = 00000001
SSHIGH = 00000000
SSLIMIT = 00000001
SSLOW = FFFFFFFF
SSMNSW = 00000001
SSMXSW = 00000001
BLD_COMMON = 0000027E R 03
BLD_DLCKFND = 0000038A R R 03
BLD_REDO_SRCH = 00000268 R R 03
BLD_SRCHDLCK = 00000278 R R 03
BLD_TIMESTAMP_RQST = 00000270 R 03
BUGS_LOCKMGRERR = ***** X 03
CAS_MEASURE = 00000002
CDRPSK_CM_LONG_LENGTH = 00000064
CDRPSL_MSGBLD = 0000004C
CDRPSL_VAL1 = 0000002C
CDRPSL_VAL10 = 00000060
CDRPSL_VAL3 = 00000034
CDRPSL_VAL5 = 0000003C
CDRPSL_VAL7 = 00000044
CDRPSL_VAL9 = 0000005C
CDRPSW_CDRPSIZE = 00000008
CHECK_TIMESTAMP = 00000156 R 03
CLMSGSSB_FACILITY = 00000008
CLMSGSSB_FUNC = 00000009
CLMSGSSK_FAC_LCK = 00000002
CLUSGL_CCUB = ***** X 03
CLUBSL_LOCAL_CSID = 00000060
CLUBSW_MEMSEQ = 000000AC
CNX$DEALL_MSG_BUF_CSB = ***** X 03
CNX$INIT_CDRP = ***** X 03
CNX$RCV_REJECT = ***** X 03
CNX$SEND_MSG = ***** X 03
DEALL_DLCK_MSG = 0000049F R 03
DYN$C_FRK = 00000008
EXESA[ONONPAGED = ***** X 03
EXESDEANONPAGED = ***** X 03
EXESFORK_WAIT = ***** X 03
EXESGL_ABSTIM = ***** X 03
EXESGL_INTSTKLM = ***** X 03
EXESGL_SYSTIME = ***** X 03
FKBSK_LENGTH = 00000018
GET_TIMESTAMP = 000000AF R 03
IPL$ SYNCH = 00000008
LCK$ALLOC_LONGCDRP = 000004AD R 03
LCK$BREAK_DEADLOCK = ***** X 03
LCK$CVT_ID_TO_LKB = 00000454 RG 03
LCK$DISPATCH = ***** X 03
LCK$DLCKEXIT = ***** X 03
LCK$GL_EXTRASTK = ***** X 03
LCK$GL_IDTBL = ***** X 03
LCK$GL_MAXID = ***** X 03
LCK$GL_PRCMAP = ***** X 03
LCK$GL_TIMEOUTQ = ***** X 03
LCK$GL_TS_CSID = ***** X 03

```

```

LCK$GL_WAITTIME = ***** X 03
LCK$GQ_BITMAP_EXP = ***** X 03
LCK$RCV_DLCKFND = 000003A0 RG 03
LCK$RCV_REDO_SRCH = 000003F0 RG 03
LCK$RCV_SRCHDLCK = 0000028E RG 03
LCK$RCV_TIMESTAMP_RQST = 00000101 RG 03
LCK$SND_DLCKFND = 00000346 RG 03
LCK$SND_REDO_SRCH = 0000038B RG 03
LCK$SND_SRCHDLCK = 000001A0 RG 03
LCK$SND_TIMESTAMP_RQST = 00000000 RG 03
LCK$SRCH_RESDLCK = ***** X 03
LCK$V_NODLCKWT = 00000009
LKBSB_STATE = 00000036
LKBSB_TSLT = 0000004E
LKBSK_CONVERT = 00000000
LKBSK_GRANTED = 00000001
LKBSK_WAITING = FFFFFFFF
LKBSL_ASTQFL = 00000000
LKBSL_CSID = 00000058
LKBSL_DLCKPRI = 00000024
LKBSL_DUE TIME = 00000018
LKBSL_LKID = 00000030
LKBSL_OWNOBL = 00000044
LKBSL_OWNOFL = 00000040
LKBSL_PID = 0000000C
LKBSL_REMLKID = 00000054
LKBSL_RSB = 00000050
LKBSM_TIMEOUTQ = 00000040
LKBSV_MSTCPY = 00000004
LKBSV_TIMEOUTQ = 00000006
LKBSW_FLAGS = 00000028
LKBSW_STATUS = 0000002A
LKMSG$B_TSLT = 0000000E
LKMSG$K_DLCKFND = 0000000B
LKMSG$K_DLM_LENGTH = 00000034
LKMSG$K_REDO_SRCH = 0000000C
LKMSG$K_SRCHDLCK = 0000000A
LKMSG$K_TSRQST = 00000009
LKMSG$S_NEXTLKID = 00000030
LKMSG$S_ORIGCSID = 00000018
LKMSG$S_ORIGEPID = 00000010
LKMSG$S_ORIGLKID = 00000014
LKMSG$S_VCTMCSID = 0000002C
LKMSG$S_VCTMLKID = 00000028
LKMSG$S_VCTMPRI = 00000024
LKMSG$Q_BITMAP_EXP = 0000001C
LKMSG$W_MEMSEQ = 0000000C
LKMSG_BFR = 00000018 R 02
LKMSG_FKB = 00000000 R 02
LOCKFRAME = 00000018
MAX_TSLT = 00000005
PCBSL_DLCKPRI = 0000010C
PCBSL_LOCKQFL = 00000104
PM$SGE_DLCKMSGS_IN = ***** X 03
PM$SGL_DLCKMSGS_OUT = ***** X 03
RCV_DLCK_MSG = 0000048C R 03
REDO_SRCH = 000003F8 R 03

```

RSBSL CSID = 00000038
 SCHSGC PCBVEC ***** X 03
 SEND_DECK_MSG 0000047A R 03
 TSLT_UNITS = 0007A120
 WAIT_FOR_POOL 000004CB R 03

 ! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$040	00000040 (76.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
\$\$\$020	00000510 (1308.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

 ! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.03	00:00:01.40
Command processing	136	00:00:00.41	00:00:01.55
Pass 1	451	00:00:11.62	00:00:44.05
Symbol table sort	0	00:00:01.79	00:00:05.56
Pass 2	248	00:00:02.90	00:00:15.85
Symbol table output	15	00:00:00.09	00:00:00.56
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	889	00:00:16.86	00:01:08.99

The working set limit was 2100 pages.
 99692 bytes (195 pages) of virtual memory were used to buffer the intermediate code.
 There were 100 pages of symbol table space allocated to hold 1684 non-local and 57 local symbols.
 1432 source lines were read in Pass 1, producing 20 object records in Pass 2.
 28 pages of virtual memory were used to define 26 macros.

 ! Macro library statistics !

Macro library name	Macros defined
_\$255\$DUA28:[SYSLOA.OBJ]CLUSTER.MLB;1	1
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	13
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	20

1790 GETS were required to define 20 macros.

There were no errors, warnings or information messages.

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

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

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CSPOPCOM
LIS

CSPWAIT
LIS

CSPRPCAC
LIS

CSPCJFRES
LIS

CSPQUORUM
LIS

DISTRKI
LIS

CSPMOUNT
LIS

CSPVECTOR
LIS

CSPCLIENT
LIS

DSTRLOCK
LIS

DSTRLOCK
LIS