F 13

```
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSSS  YYY         YYY
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSSS  YYY         YYY
UUU         UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSSS  YYY         YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS              YYY         YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS              YYY         YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS              YYY         YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS                 YYY   YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS                 YYY   YYY
UUU         UUU  EEE                    TTT        PPP         PPP  SSS                 YYY   YYY
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP        SSSSSSSSS               YYY
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP        SSSSSSSSS               YYY
UUU         UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP        SSSSSSSSS               YYY
UUU         UUU  EEE                    TTT        PPP                          SSS            YYY
UUU         UUU  EEE                    TTT        PPP                          SSS            YYY
UUU         UUU  EEE                    TTT        PPP                          SSS            YYY
UUU         UUU  EEE                    TTT        PPP                          SSS            YYY
UUU         UUU  EEE                    TTT        PPP                          SSS            YYY
UUU         UUU  EEE                    TTT        PPP                          SSS            YYY
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP              SSSSSSSSSSSS               YYY
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP              SSSSSSSSSSSS               YYY
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP              SSSSSSSSSSSS               YYY
```

FILEID**UETLOCK00

```
UU      UU EEEEEEEEEE TTTTTTTTTT LL          000000    CCCCCCCC  KK      KK   000000      000000
UU      UU EEEEEEEEEE TTTTTTTTTT LL          000000    CCCCCCCC  KK      KK   000000      000000
UU      UU EE             TT     LL        00      00 CC         KK    KK   00      00  00      00
UU      UU EE             TT     LL        00      00 CC         KK    KK   00      00  00      00
UU      UU EE             TT     LL        00      00 CC         KK  KK     00    0000  00    0000
UU      UU EE             TT     LL        00      00 CC         KK  KK     00    0000  00    0000
UU      UU EEEEEEEE       TT     LL        00      00 CC         KKKKKK     00  00  00  00  00  00
UU      UU EEEEEEEE       TT     LL        00      00 CC         KKKKKK     00  00  00  00  00  00
UU      UU EE             TT     LL        00      00 CC         KK  KK     0000    00  0000    00
UU      UU EE             TT     LL        00      00 CC         KK  KK     0000    00  0000    00
UU      UU EE             TT     LL        00      00 CC         KK    KK   00      00  00      00    ....
UU      UU EE             TT     LL        00      00 CC         KK    KK   00      00  00      00    ....
UUUUUUUUUU EEEEEEEEEE     TT     LLLLLLLLLL  000000    CCCCCCCC  KK      KK   000000      000000      ....
UUUUUUUUUU EEEEEEEEEE     TT     LLLLLLLLLL  000000    CCCCCCCC  KK      KK   000000      000000      ....

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

Table of contents

```
0000          .TITLE  UETLOCK00 - Local Lock Manager UETP Test
0000          .IDENT  'V04-000'
0000          .ENABLE SUPPRESSION
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000
0000    ;++
0000    ; FACILITY: VAX/VMS UETP
0000    ;
0000    ; ABSTRACT:
0000    ;       This module serves as both the controlling (driver) and slave (driven)
0000    ;       images of the UETP Lock Manager Test.  Depending on the invocation and
0000    ;       parameters supplied, it chooses to be the driver, in which case it
0000    ;       starts up copies of itself and directs their actions; or the driven
0000    ;       process, in which case it is started by the driver and is told what
0000    ;       action to perform next.
0000    ;
0000    ; ENVIRONMENT:
0000    ;       The DETACH and GROUP privileges and ENQLM=20 are needed to run this
0000    ;       test.
0000    ;
0000    ;--
0000    ;
0000    ; AUTHOR:  Paul Jenq,   CREATION DATE:  8-Aug-1981
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-006 RNH0005         Richard N. Holstein,    26-Apr-1984
0000    ;               V3-5 left SYI$_SCSNODE at 4 instead of 6.  Fix that and force
0000    ;               mailbox logical names to go into a group logical name table.
0000    ;
0000    ;       V03-005 WHM0001         Bill Matthews           14-Apr-1984
0000    ;               Replace reference to SYI$_SCSNODEL/H with SYI$_SCSNODE.
0000    ;
0000    ;       V03-004 RNH0004         Richard N. Holstein,    23-Nov-1983
0000    ;               Use lock names which will allow the test to run on more than
```

```
0000  ;                 one VMS node of a cluster simultaneously - insert 'UETP$node_'.
0000  ;
0000  ;         V03-003 RNH0003         Richard N. Holstein,    17-Nov-1983
0000  ;                 Fix bug which prevented deleting log files on successful runs.
0000  ;
0000  ;         V03-002 RNH0002         Richard N. Holstein,    11-Mar-1983
0000  ;                 Don't signal ending message in EXIT_HANDLER.
0000  ;
0000  ;         V03-001 RNH0001         Richard N. Holstein,    11-Mar-1983
0000  ;                 Eliminate redundant error checking when opening SYS$INPUT.
0000  ;                 Don't try for subprocess logs if the subprocesses weren't
0000  ;                 created.  Indent copied subprocess logs.  Misc fixes carried
0000  ;                 over from device tests.
0000  ;
0000  ;**
```

```
0000            .SBTTL  Declarations
0000    ;
0000    ; INCLUDE FILES:
0000    ;
0000    ;       SYS$LIBRARY:LIB.MLB         for general definitions
0000    ;       SHRLIB$:UETP.MLB            for UETP definitions
0000    ;
0000
0000    ;
0000    ; MACROS:
0000    ;
0000            $ACCDEF                         ; Account msg definition
0000            $CHFDEF                         ; Condition handler
0000            $DEVDEF                         ; Device characteristics
0000            $DIBDEF                         ; Device infomation block
0000            $DVIDEF                         ; $GETDVI ITMLST item codes
0000            $JPIDEF                         ; Job/proc information def
0000            $LCKDEF                         ; LCK definition
0000            $LNMDEF                         ; Logical name services codes
0000            $MSGDEF                         ; Message definition
0000            $SHRDEF                         ; Shared message
0000            $STSDEF                         ; Status definition
0000            $SYIDEF                         ; $GETSYI ITMLST item codes
0000            $UETPDEF                        ; UETP messages
0000    ;
0000    ; Macro to build test table entries.  NOTE WELL!  Some code depends on this
0000    ; table having entries which are eight bytes long (i.e., a quadword).  This
0000    ; length is available as the parameter COMMAND_SIZE.
0000    ;
0000    .MACRO TST_TABLE PROC,TYP,LKMOD,RESR=<A>,PAR=< >,FLAG,TST_FLG=<^X0000>
0000            .IIF LT ^A/PROC/-LOW_PROC,-
0000                    .ERROR 0                ; Illegal proc name PROC in TST_TABLE
0000            .IIF GT ^A/PROC/-HIGH_PROC,-
0000                    .ERROR 0                ; Illegal proc name PROC in TST_TABLE
0000            .ASCII  /PROC/                  ; Process name to perform the action
0000            .BYTE   TYP                     ; Type of system svc: ENQ, ENQW or DEQ
0000            .IIF LT TYP, .ERROR TYP         ; Illegal TYP-argument in TST_TABLE
0000            .IIF GT TYP-MAXCODE, .ERROR TYP ; Illegal TYP-argument in TST_TABLE
0000            .IF B LKMOD
0000                    .BYTE   ^XFF            ; Type of lock - $ENQ LKMODE argument
0000            .IFF
0000                    .BYTE   LCK$K_'LKMOD'MODE ; Type of lock - $ENQ LKMODE argument
0000            .ENDC
0000            .ASCII  /RESR/                  ; One character resource name
0000            .ASCII  /PAR/                   ; One character parent lock
0000            .IF B FLAG
0000                    .BYTE   0               ; Lock characs - $ENQ FLAGS argument
0000            .IFF
0000                    .BYTE   LCK$M_'FLAG     ; Lock characs - $ENQ FLAGS argument
0000            .ENDC
0000            .WORD   TST_FLG                 ; Test action after calling system svc
0000    .ENDM   TST_TABLE
0000
0000    ;
0000    ; This next macro is used to build the table of process names, to generate the
0000    ; parameters we need to check that we only use legal process names and to build
0000    ; various other data structures which rely on process names.  It calls the X
```

```
            0000  ; macro, which must be defined each time to the specific purpose to be
            0000  ; performed.  PROCESS NAMES MUST BE ONE ASCII CHARACTER!
            0000  ;
            0000  .MACRO  PROC_NAMES
            0000          X       P                       ; Master (driver) process
            0000          X       Q                       ; First slave (driven) process
            0000          X       R                       ; Next slave (driven) process
            0000  .ENDM   PROC_NAMES
            0000
            0000  ;
            0000  ; EQUATED SYMBOLS:
            0000  ;
            0000
00000001    0000          RMS$_FACILITY = 1               ; Standard RMS facility code
00740000    0000          UETP = UETP$_FACILITY@STS$V_FAC_NO ; UETP facility code massaged
007410E0    0000          UETP$_ABENDD = UETP!SHR$_ABENDD ; Abort message codes
00741038    0000          UETP$_BEGIND = UETP!SHR$_BEGIND ; Begin message
00741080    0000          UETP$_ENDEDD = UETP!SHR$_ENDEDD ; End message
00741130    0000          UETP$_TEXT   = UETP!SHR$_TEXT   ; Text message
            0000
            0000  ; Code depends not only on individual values of the following, but also on
            0000  ; their ordering.
00000000    0000          ENQ     = 0                     ; Do ENQ. Code depends on this value
00000001    0000          ENQW    = 1                     ; Do ENQW. Code depends on this value
00000002    0000          DEQ     = 2                     ; Do DEQ
00000003    0000          ENDTEST = 3                     ; END OF TEST code
00000003    0000          MAXCODE = 3                     ; Greatest legal TYP field in TST_TABLE
            0000
            0000  ; Generate high and low bounds for process names.  Count process names.  Note
            0000  ; that the first process name is assumed to be the master process and that we
            0000  ; don't include it in the count.
            0000  .MACRO  X       PROCESS
            0000          .IIF NDF PROC_COUNT, PROC_COUNT = -1 ; Initialize count of processes
            0000          PROC_COUNT = PROC_COUNT+1            ; First is master, don't count it
            0000          .IIF NDF LOW_PROC,   LOW_PROC   = ^A/PROCESS/ ; Low bound
            0000          .IIF NDF HIGH_PROC,  HIGH_PROC  = ^A/PROCESS/ ; Initial high bound
            0000          .IIF LT  HIGH_PROC-^A/PROCESS/, HIGH_PROC = ^A/PROCESS/ ; High bound
            0000  .ENDM   X       ; For process name bounds
            0000          PROC_NAMES                      ; Generate bounds and count processes
            0000
            0000  ;
            0000  ; Common event flags
            0000  ;
00000040    0000          DONE_CEF    = 64                ; CEF for action complete by driven proc
00000041    0000          BAST_CEF    = 65                ; CEF for blocking ast delivery...
00000001    0000          BAST_CEF_V  = <65-64>           ; ...and corresponding bit
00000042    0000          CMP_VAL     = 66                ; CEF for compare value...
00000002    0000          CMP_VAL_V   = <66-64>           ; ...and corresponding bit
00000043    0000          UNLOCK_CEF  = 67                ; CEF for unlock
00000044    0000          DLDET_CEF   = 68                ; CEF for deadlock detected...
00000004    0000          DLDET_CEF_V= DLDET_CEF-64       ; ...and corresponding bit
00000045    0000          DLRES_CEF   = 69                ; CEF for deadlock resolved
00000030    0000          DLMASK      = <1@<DLDET_CEF-64>>!- ; Mask for $WFLAND of deadlock flags
            0000                        <1@<DLRES_CEF-64>>
            0000
            0000  ;
            0000  ; Bits in TSTFLG.
```

```
          0000  ;
00000000  0000          BLKAST_V = 0                    ; Blocking AST test
00000001  0000          INCOMP_V = 1                    ; Lock mode incompatibility test
00000002  0000          VALBLK_V = 2                    ; Value block test
00000003  0000          SYNCST_V = 3                    ; SYNCSTS test
00000004  0000          DEADLK_V = 4                    ; Deadlock test - don't know victim
00000005  0000          NOCAST_V = 5                    ; No completion AST
00000006  0000          NOWAIT_V = 6                    ; Do not wait for DONE CEF
00000007  0000          VICTIM_V = 7                    ; This is a deadlock victim for sure
          0000
00000001  0000          BLKAST_M = 1@BLKAST_V           ; And corresponding bit masks
00000002  0000          INCOMP_M = 1@INCOMP_V
00000004  0000          VALBLK_M = 1@VALBLK_V
00000008  0000          SYNCST_M = 1@SYNCST_V
00000010  0000          DEADLK_M = 1@DEADLK_V
00000020  0000          NOCAST_M = 1@NOCAST_V
00000040  0000          NOWAIT_M = 1@NOWAIT_V
00000080  0000          VICTIM_M = 1@VICTIM_V
          0000  ;
          0000  ; Bits in LOCFLG.
          0000  ;
00000000  0000          C_AST_V = 0                     ; Set if completion AST was delivered
00000001  0000          D[PRC_V = 1                     ; Set if $FORCEX done on driven procs
00000001  0000          C_AST_M = 1@C_AST_V
00000002  0000          D[PRC_M = 1@D[PRC_V
          0000  ;
          0000  ; Bits in GLBFLG.
          0000  ;
00000000  0000          DRIVEN_V = 0                    ; Set if we are a driven (slave) proc
00000001  0000          DRIVEN_M = 1@DRIVEN_V
00000001  0000          DUMP_V   = 1                    ; Set if we're running in dump mode
00000002  0000          DUMP_M   = 1@DUMP_V
00000002  0000          BEGIN_MSGV = 2                  ; Set if we've printed our beginning msg
00000004  0000          BEGIN_MSGM = 1@BEGIN_MSGV
          0000  ;
          0000  ; Miscellaneous
          0000  ;
00000001  0000          SYNC_EF     = 1                 ; Local EF for SYNCSTS test
00000004  0000          SNDMSG_EFN  = 4                 ; EFN for QIO to send to driven procs
00000084  0000          TEXT_BUFFER = 132               ; Buffer length for FAO output
00000018  0000          LKSB_SIZE   = 24                ; Size of lock status block in bytes
00000008  0000          COMMAND_SIZE= 8                 ; Size of a TEST_TABLE entry
00000005  0000          SS_SYNCH_EFN= 5                 ; Synch miscellaneous system services
00000004  0000          INDENT      = 4                 ; Spaces to indent when copying logs
```

```
    0000                .SBTTL  Read-Only Data
00000000                .PSECT  RODATA,NOEXE,NOWRT,PAGE
    0000        ;
    0000        ; See the PROC_NAMES macro definition for restrictions on the names of
    0000        ; processes.
    0000        ;       TST_TABLE process,sstype,lock-mode,resource,parent-id,ssflags,action-flags
    0000        ;
    0000
    0000        TEST_TABLE:
    0000        ;
    0000        ; Test of new locks grant
    0000        ;
    0000                TST_TABLE <P>,ENQ,NL,<A>
    0008
    0008                TST_TABLE <P>,ENQ,CR,<B>
    0010
    0010                TST_TABLE <P>,ENQ,CW,<C>
    0018
    0018                TST_TABLE <P>,ENQ,PR,<D>
    0020
    0020                TST_TABLE <P>,ENQ,PW,<E>
    0028
    0028                TST_TABLE <P>,ENQ,EX,<F>
    0030        ;
    0030        ; Test of lock conversion
    0030        ;
    0030                TST_TABLE <P>,ENQW,CR,<A>,,CONVERT
    0038
    0038                TST_TABLE <P>,ENQW,PR,<B>,,CONVERT
    0040
    0040                TST_TABLE <P>,ENQW,PW,<C>,,CONVERT
    0048
    0048                TST_TABLE <P>,ENQW,NL,<D>,,CONVERT
    0050
    0050                TST_TABLE <P>,ENQW,EX,<E>,,CONVERT
    0058
    0058                TST_TABLE <P>,ENQW,CW,<F>,,CONVERT
    0060        ;
    0060        ; Test of Deque
    0060        ;
    0060                TST_TABLE <P>,DEQ,,<A>
    0068
    0068                TST_TABLE <P>,DEQ,,<B>
    0070
    0070                TST_TABLE <P>,DEQ,,<C>
    0078
    0078                TST_TABLE <P>,DEQ,,<D>
    0080
    0080                TST_TABLE <P>,DEQ,,<E>
    0088
    0088                TST_TABLE <P>,DEQ,,<F>
    0090        ;
    0090        ; Test of lock mode compatibility and blocking AST
    0090        ;
    0090                TST_TABLE <P>,ENQW,NL,<A>,,,BLKAST_M
    0098
    0098                TST_TABLE <Q>,ENQW,CR,<A>
```

```
00A0
00A0            TST_TABLE <Q>,ENQW,EX,<A>,,CONVERT
00A8
00A8            TST_TABLE <P>,ENQW,CR,<B>,,,BLKAST_M
00B0
00B0            TST_TABLE <Q>,ENQW,CW,<B>
00B8
00B8            TST_TABLE <Q>,ENQW,PR,<B>,,CONVERT
00C0
00C0            TST_TABLE <Q>,ENQW,EX,<B>,,CONVERT,INCOMP_M
00C8
00C8            TST_TABLE <P>,ENQW,CW,<C>,,,BLKAST_M
00D0
00D0            TST_TABLE <Q>,ENQW,PR,<C>,,,INCOMP_M
00D8
00D8            TST_TABLE <Q>,DEQ,,<C>
00E0
00E0            TST_TABLE <P>,ENQW,CW,<C>,,,BLKAST_M
00E8
00E8            TST_TABLE <Q>,ENQW,CR,<C>
00F0
00F0            TST_TABLE <Q>,ENQW,EX,<C>,,CONVERT,INCOMP_M
00F8
00F8            TST_TABLE <P>,ENQW,PR,<D>,,,BLKAST_M
0100
0100            TST_TABLE <Q>,ENQW,CW,<D>,,,INCOMP_M
0108
0108            TST_TABLE <Q>,DEQ,,<D>
0110
0110            TST_TABLE <P>,ENQW,PR,<D>,,,BLKAST_M
0118
0118            TST_TABLE <Q>,ENQW,EX,<D>,,,INCOMP_M
0120
0120            TST_TABLE <P>,ENQW,PW,<E>,,,BLKAST_M
0128
0128            TST_TABLE <Q>,ENQW,CR,<E>
0130
0130            TST_TABLE <Q>,ENQW,PW,<E>,,CONVERT,INCOMP_M
0138
0138            TST_TABLE <Q>,DEQ,,<E>
0140
0140            TST_TABLE <P>,ENQW,EX,<F>,,,BLKAST_M
0148
0148            TST_TABLE <Q>,ENQW,CR,<F>,,,INCOMP_M
0150
0150            TST_TABLE <Q>,DEQ,,<F>
0158
0158            TST_TABLE <P>,ENQW,EX,<F>,,,BLKAST_M
0160
0160            TST_TABLE <Q>,ENQW,EX,<F>,,,INCOMP_M
0168        ;
0168        ; Test of value block
0168        ;
0168            TST_TABLE <P>,ENQW,CR,<V>,,VALBLK
0170
0170            TST_TABLE <Q>,ENQW,CR,<V>,,VALBLK
0178
```

```
          0178          TST_TABLE <P>,ENQW,PW,<V>,,<VALBLK!LCK$M_CONVERT>,VALBLK_M
          0180
          0180          TST_TABLE <P>,ENQW,CR,<V>,,<VALBLK!LCK$M_CONVERT>,
          0188
          0188          TST_TABLE <Q>,ENQW,PW,<V>,,<VALBLK!LCK$M_CONVERT>,VALBLK_M
          0190    ;
          0190    ; Test of tree structured lock and LCK$M_NOQUEUE flag
          0190    ;
          0190          TST_TABLE <P>,ENQW,CW,<U>
          0198
          0198          TST_TABLE <P>,ENQW,PW,<X>,<U>
          01A0
          01A0          TST_TABLE <P>,ENQW,EX,<Z>,<X>
          01A8
          01A8          TST_TABLE <Q>,ENQW,CR,<U>
          01B0
          01B0          TST_TABLE <Q>,ENQW,PR,<X>,<U>,NOQUEUE,NOCAST_M
          01B8    ;
          01B8    ; Test of LCK$M_SYNCSTS flag
          01B8    ;
          01B8          TST_TABLE <P>,ENQW,PW,<S>,,SYNCSTS,<SYNCST_M!BLKAST_M>
          01C0
          01C0          TST_TABLE <Q>,ENQ,EX,<S>,,SYNCSTS
          01C8    ;
          01C8    ; Test of Local deadlock detection
          01C8    ;
          01C8          TST_TABLE <P>,ENQW,EX,<L>
          01D0
          01D0          TST_TABLE <Q>,ENQW,EX,<M>
          01D8
          01D8          TST_TABLE <P>,ENQ,EX,<M>,,,DEADLK_M
          01E0
          01E0          TST_TABLE <Q>,ENQ,EX,<L>,,,DEADLK_M!NOWAIT_M
          01E8
          01E8          TST_TABLE <R>,ENQW,NL,<N>
          01F0
          01F0          TST_TABLE <R>,ENQW,EX,<N>
          01F8
          01F8          TST_TABLE <R>,ENQW,EX,<N>,,,VICTIM_M
          0200    ;
          0200    ; Tell all detached processes to terminate.  Generate a termination message for
          0200    ; all processes.  The driver process is smart enough to ignore the message
          0200    ; because it must wait for all other processes to finish.
          0200    ;
          0200    .MACRO  X       PROCESS
          0200          TST_TABLE <PROCESS>,ENDTEST,,,,,NOWAIT_M
          0200    .ENDM   X       ; For termination list
          0200          PROC_NAMES                      ; Generate list for termination msgs
          0218
          0218    TABLE_END:                            ; End of Test table
00000000  0218          .LONG   0
          021C    TO_BE_FILLED:                         ; For patch use
0000022C  021C          .BLKQ   2
```

```
                                  022C   401 BEGUN_ADDR:                                  ; Used by UETP$_SATSMS
             6E 75 67 65 62 00'   022C   402          .ASCIC  /begun/
                             05   022C
                                  0232   403 END_ADDR:
    6C 75 66 73 73 65 63 63 75 73 00'   0232   404          .ASCIC  /successful/
                             0A   0232
                                  023D   405 FAIL_ADDR:
          64 65 6C 69 61 66 00'   023D   406          .ASCIC  /failed/
                             06   023D
                                  0244   407
                                  0244   408 MODE:                                        ; Determines some runtime actions...
     45 44 4F 4D 0000024C'010E0000'   0244   409          .ASCID  /MODE/                   ; ...based on log name translation
                                  0250   410
                                  0250   411 SYS$INPUT:                                   ; Name of device from which...
4E 49 24 53 59 53 00000258'010E0000'   0250   412          .ASCID  /SYS$INPUT/              ; ...the test can be aborted
                       54 55 50   025E
                                  0261   413
                                  0261   414 INPUT_ITMLST:                                ; $GETDVI arg list for SYS$INPUT
                      0020 0040   0261   415          .WORD   64,DVI$_DEVNAM           ; We need the equivalence name
              00000481'00000489'   0265   416          .LONG   BUFFER,BUFFER_PTR
                       00000000   026D   417          .LONG   0                        ; Terminate the list
                                  0271   418
                                  0271   419 TEN_SECONDS:                                 ; Brief timeout so that subprocesses...
              FFFFFFFF FA0A1F00   0271   420          .LONG   -10*1000*1000*10,-1      ; ...can finish after $FORCEX
                                  0279   421
                                  0279   422 THREE_MIN:                                   ; A three minute timer so that...
                       94B62E00   0279   423          .LONG   -10*1000*1000*180        ; ...the test never hangs forever
                       FFFFFFFF   027D   424          .LONG   -1
                                  0281   425
                                  0281   426 LK_CEF_DESC:                                 ; Logical name for common EF
54 53 45 54 4B 4C 00000289'010E0000'   0281   427          .ASCID  /LKTEST_CEF/
                    46 45 43 5F   028F
                                  0293   428
                                  0293   429 LKTEST_DESC:                                 ; Image name of driven process
43 4F 4C 54 45 55 0000029B'010E0000'   0293   430          .ASCID  /UETLOCK00.EXE/
           45 58 45 2E 30 30 4B   02A1
                                  02A8   431
                                  02A8   432 ; Variable part of names of detached processes. This depends on one-char
                                  02A8   433 ; names, as described in the PROC_NAMES macro definition.
                                  02A8   434 .MACRO  X       PROCESS
                                  02A8   435          .ASCII  /PROCESS/
                                  02A8   436 .ENDM   X       ; For table of ASCII process names
                                  02A8   437 ALL_PROCS:                                   ; Names of all processes
                                  02A8   438          PROC_NAMES                       ; This list includes the driver process
                       000002A9   02AB   439 PROCS = ALL_PROCS+1                          ; PROCS differs from ALL_PROCS in
                                  02AB   440                                              ; ...it excludes the driver process
                                  02AB   441
                                  02AB   442 LOG_MSGVEC:                                  ; $PUTMSG MSGVEC arg so we can print...
                      0001 0003   02AB   443          .WORD   3,1                      ; ...everything from...
                       00741130   02AF   444          .LONG   UETP$_TEXT               ; ...the driven procs' .LOG file(s)...
                      0001 0001   02B3   445          .WORD   1,1                      ; ...to our SYS$OUTPUT
                      0000051D'   02B7   446          .ADDRESS LOG_MSGPTR
                                  02BB   447
                                  02BB   448 CNTRLCMSG:                                   ; Control C messsage
65 74 72 6F 62 41 000002C3'010E0000'   02BB   449          .ASCID  \Aborted via a user CTRL/C\
72 65 73 75 20 61 20 61 69 76 20 64   02C9
           43 2F 4C 52 54 43 20   02D5
```

```
                                            02DC
                                            02DC   SUBPROC_STRING:                                   ; Mailbox code to tell driven proc...
20 3A 63 6F 72 50 000002E4'010E0000'        02DC            .ASCID  /Proc: /                         ; ...which one it is
                                            02EA                                                     ; THIS STRING MUST FIT INTO A...
                                            02EA                                                     ; ...MAILBOX USED FOR PASSING...
                                            02EA                                                     ; ...COMMANDS TO DRIVEN PROCESSES!
                                            02EA
                                            02EA   ILL_PROC_NAME:                                    ; Driven process name is not in our table
61 67 65 6C 6C 49 000002F2'010E0000'        02EA            .ASCID  /Illegal name for cooperating detached process: !AD./
63 20 72 6F 66 20 65 6D 61 6E 20 6C         02F8
64 20 67 6E 69 74 61 72 65 70 6F 6F         0304
63 6F 72 70 20 64 65 68 63 61 74 65         0310
         2E 44 41 21 20 3A 73 73 65         031C
                                            0325
                                            0325   ERR_IN_TABLE:
6E 72 65 74 6E 49 0000032D'010E0000'        0325            .ASCID  /Internal consistency error:  test specified non-existent process./
6E 65 74 73 69 73 6E 6F 63 20 6C 61         0333
74 20 20 3A 72 6F 72 72 65 20 79 63         033F
65 69 66 69 63 65 70 73 20 74 73 65         034B
65 74 73 69 78 65 2D 6E 6F 6E 20 64         0357
   2E 73 73 65 63 6F 72 70 20 74 6E         0363
                                            036E
                                            036E   SNDMBX_ERRMSG:
6E 64 6C 75 6F 43 00000376'010E0000'        036E            .ASCID  /Couldn't send mailbox to cooperating detached process./
6C 69 61 6D 20 64 6E 65 73 20 74 27         037C
65 70 6F 6F 63 20 6F 74 20 78 6F 62         0388
63 61 74 65 64 20 67 6E 69 74 61 72         0394
2E 73 73 65 63 6F 72 70 20 64 65 68         03A0
                                            03AC
                                            03AC   CAST_ERRMSG:
2C 72 6F 72 72 45 000003B4'010E0000'        03AC            .ASCID  /Error, no completion AST delivered./
69 74 65 6C 70 6D 6F 63 20 6F 6E 20         03BA
76 69 6C 65 64 20 54 53 41 20 6E 6F         03C6
                     2E 64 65 72 65         03D2
                                            03D7
                                            03D7   BAST_ERRMSG:
2C 72 6F 72 72 45 000003DF'010E0000'        03D7            .ASCID  /Error, no blocking AST delivered./
67 6E 69 6B 63 6F 6C 62 20 6F 6E 20         03E5
72 65 76 69 6C 65 64 20 54 53 41 20         03F1
                           2E 64 65         03FD
                                            0400
                                            0400   LKVAL_ERRMSG:
76 20 6B 63 6F 4C 00000408'010E0000'        0400            .ASCID  /Lock value block returned !XL instead of !XL./
72 20 6B 63 6F 6C 62 20 65 75 6C 61         040E
20 4C 58 21 20 64 65 6E 72 75 74 65         041A
21 20 66 6F 20 64 61 65 74 73 6E 69         0426
                           2E 4C 58         0432
                                            0435
                                            0435   NOSYNCH_ERRMSG:
20 57 51 4E 45 24 0000043D'010E0000'        0435            .ASCID  /$ENQW returned !XL instead of SS$_SYNCH./
4C 58 21 20 64 65 6E 72 75 74 65 72         0443
20 66 6F 20 64 61 65 74 73 6E 69 20         044F
      2E 48 43 4E 59 53 5F 24 53 53         045B
                                            0465
                                            0465   CASTSYNCH_ERRMSG:
20 57 51 4E 45 24 0000046D'010E0000'        0465            .ASCID  /$ENQW delivered a completion AST for a resource which/-
20 61 20 64 65 72 65 76 69 6C 65 64         0473
```

```
41 20 6E 6F 69 74 65 6C 70 6D 6F 63   047F
73 65 72 20 61 20 72 6F 66 20 54 53   048B
   68 63 69 68 77 20 65 63 72 75 6F   0497
6F 6E 20 64 6C 75 6F 68 73 09 0A 0D   04A2            <13><10>/       should not be free when LCK$M_SYNCSTS was set./
68 77 20 65 65 72 66 20 65 62 20 74   04AE
4E 59 53 5F 4D 24 4B 43 4C 20 6E 65   04BA
74 65 73 20 73 61 77 20 53 54 53 43   04C6
                                  2E   04D2
                                       04D3
                                       04D3     SYNCH_ERRMSG:
20 57 51 4E 45 24 000004DB'010E0000'   04D3             .ASCID  /$ENQW set a synchronization event flag, though it returned/-
72 68 63 6E 79 73 20 61 20 74 65 73   04E1
76 65 20 6E 6F 69 74 61 7A 69 6E 6F   04ED
68 74 20 2C 67 61 6C 66 20 74 6E 65   04F9
75 74 65 72 20 74 69 20 68 67 75 6F   0505
                        64 65 6E 72   0511
53 43 4E 59 53 5F 24 53 53 09 0A 0D   0515            <13><10>/       SS$_SYNCSTS and delivered no completion AST./
76 69 6C 65 64 20 64 6E 61 20 53 54   0521
70 6D 6F 63 20 6F 6E 20 64 65 72 65   052D
   2E 54 53 41 20 6E 6F 69 74 65 6C   0539
                                       0544
                                       0544     NOSYQUEUE_ERRMSG:
20 57 51 4E 45 24 0000054C'010E0000'   0544             .ASCID  \$ENQW did not queue a request for a resource which was in\-
75 65 75 71 20 74 6F 6E 20 64 69 64   0552
20 74 73 65 75 71 65 72 20 61 20 65   055E
72 75 6F 73 65 72 20 61 20 72 6F 66   056A
73 61 77 20 68 63 69 68 77 20 65 63   0576
                           6E 69 20   0582
75 6F 68 74 09 2F 21 2C 65 73 75 20   0585            \ use,!/        though LCK$M_SYNCSTS was set; status was !XL.\
4E 59 53 5F 4D 24 4B 43 4C 20 68 67   0591
74 65 73 20 73 61 77 20 53 54 53 43   059D
73 61 77 20 73 75 74 61 74 73 20 3B   05A9
                     2E 4C 58 21 20   05B5
                                       05BA
                                       05BA     PAR_ERRMSG:
6F 20 51 4E 45 24 000005C2'010E0000'   05BA             .ASCID  /$ENQ or $ENQW returned !XL instead of SS$_NOTQUEUED./
75 74 65 72 20 57 51 4E 45 24 20 72   05C8
73 6E 69 20 4C 58 21 20 64 65 6E 72   05D4
5F 24 53 53 20 66 6F 20 64 61 65 74   05E0
      2E 44 45 55 45 55 51 54 4F 4E   05EC
                                       05F6
                                       05F6     VICTIM_ERRMSG:
20 67 6E 6F 72 57 000005FE'010E0000'   05F6             .ASCID  \Wrong or no victim selected for deadlock resolution,\-
6D 69 74 63 69 76 20 6F 6E 20 72 6F   0604
6F 66 20 64 65 74 63 65 6C 65 73 20   0610
72 20 6B 63 6F 6C 64 61 65 64 20 72   061C
      2C 6E 6F 69 74 75 6C 6F 73 65   0628
20 64 65 6E 72 75 74 65 72 09 2F 21   0632            \!/     returned lock status of !XL.\
20 73 75 74 61 74 73 20 6B 63 6F 6C   063E
               2E 4C 58 21 20 66 6F   064A
                                       0651
                                       0651     DEQALL_ERRMSG:
6F 20 51 45 44 24 00000659'010E0000'   0651             .ASCID  \$DEQ of all locks returned !XL instead of SS$_NORMAL.\
20 73 6B 63 6F 6C 20 6C 6C 61 20 66   065F
4C 58 21 20 64 65 6E 72 75 74 65 72   066B
20 66 6F 20 64 61 65 74 73 6E 69 20   0677
   2E 4C 41 4D 52 4F 4E 5F 24 53 53   0683
```

```
                                      068E
                                      068E  DEADLK_ERRMSG:
6F 6C 64 61 65 44 00000696'010E0000'  068E           .ASCID  \Deadlock was not detected/resolved in !UL seconds.\
64 20 74 6F 6E 20 73 61 77 20 6B 63   069C
6F 73 65 72 2F 64 65 74 63 65 74 65   06A8
20 4C 55 21 20 6E 69 20 64 65 76 6C   06B4
            2E 73 64 6E 6F 63 65 73   06C0
                                      06C8
                                      06C8  NODLOCK_ERRMSG:
61 63 6F 6C 6C 41 000006D0'010E0000'  06C8           .ASCID  \Allocation of deadlocked resource succeeded despite\-
64 61 65 64 20 66 6F 20 6E 6F 69 74   06D6
75 6F 73 65 72 20 64 65 6B 63 6F 6C   06E2
65 64 65 65 63 63 75 73 20 65 63 72   06EE
         65 74 69 70 73 65 64 20 64   06FA
20 6B 63 6F 6C 64 61 65 64 09 0A 0D   0703                   <13><10>\      deadlock not having been detected.\
62 20 67 6E 69 76 61 68 20 74 6F 6E   070F
64 65 74 63 65 74 65 64 20 6E 65 65   071B
                                  2E  0727
                                      0728
                                      0728  CASTPAR_ERRMSG:
65 6C 70 6D 6F 43 00000730'010E0000'  0728           .ASCID  /Completion AST parameter was !XL instead of !XL./
72 61 70 20 54 53 41 20 6E 6F 69 74   0736
21 20 73 61 77 20 72 65 74 65 6D 61   0742
6F 20 64 61 65 74 73 6E 69 20 4C 58   074E
               2E 4C 58 21 20 66      075A
                                      0760
                                      0760  NOTRMB_ERRMSG:
20 72 6F 72 72 45 00000768'010E0000'  0760           .ASCID  /Error reading termination mailbox from cooperating process./
6D 72 65 74 20 67 6E 69 64 61 65 72   076E
6C 69 61 6D 20 6E 6F 69 74 61 6E 69   077A
6F 6F 63 20 6D 6F 72 66 20 78 6F 62   0786
6F 72 70 20 67 6E 69 74 61 72 65 70   0792
                  2E 73 73 65 63      079E
                                      07A3
                                      07A3  MSGTYP_ERRMSG:
67 61 73 73 65 4D 000007AB'010E0000'  07A3           .ASCID  /Message type in termination mailbox was !XL instead of !XL./
65 74 20 6E 69 20 65 70 79 74 20 65   07B1
61 6D 20 6E 6F 69 74 61 6E 69 6D 72   07BD
58 21 20 73 61 77 20 78 6F 62 6C 69   07C9
66 6F 20 64 61 65 74 73 6E 69 20 4C   07D5
                  2E 4C 58 21 20      07E1
                                      07E6
                                      07E6  TERMBX_ERRMSG:
6E 69 6D 72 65 54 000007EE'010E0000'  07E6           .ASCID  \Termination mailbox received for an unknown process,\-
6F 62 6C 69 61 6D 20 6E 6F 69 74 61   07F4
66 20 64 65 76 69 65 63 65 72 20 78   0800
77 6F 6E 6B 6E 75 20 6E 61 20 72 6F   080C
      2C 73 73 65 63 6F 72 70 20 6E   0818
69 20 73 73 65 63 6F 72 70 09 2F 21   0822                   \!/    process id = !XL, final process status = !XL.\
6E 69 66 20 2C 4C 58 21 20 3D 20 64   082E
73 20 73 73 65 63 6F 72 70 20 6C 61   083A
2E 4C 58 21 20 3D 20 73 75 74 61 74   0846
                                      0852
                                      0852  DETPRC_ERRMSG:
20 72 6F 72 72 45 0000085A'010E0000'  0852           .ASCID  /Error in cooperating detached process, id = !XL, status = !XL./
69 74 61 72 65 70 6F 6F 63 20 6E 69   0860
20 64 65 68 63 61 74 65 64 20 67 6E   086C
```

```
20 64 69 20 2C 73 73 65 63 6F 72 70  0878
75 74 61 74 73 20 2C 4C 58 21 20 3D  0884
         2E 4C 58 21 20 3D 20 73  0890
                                  0898
                                  0898   FILE:                                    ; Fills in RMS_ERR_STRING
      65 6C 69 66 000008A0'010E0000'  0898           .ASCID  /file/
                                  08A4   RECORD:                                  ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 000008AC'010E0000'  08A4           .ASCID  /record/
                                  08B2   RMS_ERR_STRING:                          ; Announces an RMS error
41 21 20 53 4D 52 000008BA'010E0000'  08B2           .ASCID  /RMS !AS error in file !AD/
66 20 6E 69 20 72 6F 72 72 65 20 53  08C0
         44 41 21 20 65 6C 69  08CC
                                  08D3
                                  08D3   RMS_ERRMSG:
72 65 20 53 4D 52 000008DB'010E0000'  08D3           .ASCID  /RMS error with cooperating process log file, !AS./
6F 6F 63 20 68 74 69 77 20 72 6F 72  08E1
6F 72 70 20 67 6E 69 74 61 72 65 70  08ED
6C 69 66 20 67 6F 6C 20 73 73 65 63  08F9
         2E 53 41 21 20 2C 65  0905
                                  090C
                                  090C   COPY_LOG_MSG:                            ; Introduces subprocess log file
6F 20 79 70 6F 43 00000914'010E0000'  090C           .ASCID  /Copy of !AS, log file for subprocess !AD:/
66 20 67 6F 6C 20 2C 53 41 21 20 66  091A
70 62 75 73 20 72 6F 66 20 65 6C 69  0926
   3A 44 41 21 20 73 73 65 63 6F 72  0932
                                  093D
                                  093D   FORCEX_MSG:
73 65 63 6F 72 50 00000945'010E0000'  093D           .ASCID  /Process !AS was forced into exiting./
6F 66 20 73 61 77 20 53 41 21 20 73  094B
78 65 20 6F 74 6E 69 20 64 65 63 72  0957
         2E 67 6E 69 74 69  0963
                                  0969
                                  0969   DUMP_MSG:
73 65 63 6F 72 50 00000971'010E0000'  0969           .ASCID  \Process !AD trying !AC RESNAM=!AD,LKMODE=!AC,FLAGS=#^X!XB,-\-
67 6E 69 79 72 74 20 44 41 21 20 73  0977
3D 4D 41 4E 53 45 52 20 43 41 21 20  0983
21 3D 45 44 4F 4D 4B 4C 2C 44 41 21  098F
58 5E 23 3D 53 47 41 4C 46 2C 43 41  099B
         2D 2C 42 58 21  09A7
63 6F 6C 2D 44 49 52 41 50 09 2F 21  09AC                   \!/     PARID-lock=!AD.  Test flags are ^X!XW.  !%T\
74 73 65 54 20 20 2E 44 41 21 3D 6B  09B8
5E 20 65 72 61 20 73 67 61 6C 66 20  09C4
      54 25 21 20 20 2E 57 58 21 58  09D0
                                  09DA
                                  09DA   NONE:                                    ; !AD string for $FAO
            65 6E 6F 6E  09DA           .ASCII  /none/
             00000004  09DE   NONE_LENGTH = .-NONE
                                  09DE
                                  09DE   TEST_CODES:                              ; List of SS we perform
            000009EA'  09DE           .ADDRESS ENQ_CODE               ; Ordering and content are dependent...
            000009EF'  09E2           .ADDRESS ENQW_CODE              ; ...on the definitions in the...
            000009F5'  09E6           .ADDRESS DEQ_CODE               ; ...Equated Symbols section
                                  09EA   ENQ_CODE:
        51 4E 45 24 00'  09EA           .ASCIC  /$ENQ/
                    04  09EA
                                  09EF   ENQW_CODE:
     57 51 4E 45 24 00'  09EF           .ASCIC  /$ENQW/
```

```
                    05  09EF
                        09F5      DEQ_CODE:
      51 45 44 24 00'  09F5              .ASCIC  /$DEQ/
                    04  09F5
                        09FA
                        09FA      LOCK_MODES:                                    ; List of $ENQ lock modes
             00000A16'  09FA              .ADDRESS NLMODE_CODE                  ; Ordering and content are dependent...
             00000A19'  09FE              .ADDRESS CRMODE_CODE                  ; ...on the definitions supplied...
             00000A1C'  0A02              .ADDRESS CWMODE_CODE                  ; ...by the $LCKDEF macro
             00000A1F'  0A06              .ADDRESS PRMODE_CODE
             00000A22'  0A0A              .ADDRESS PWMODE_CODE
             00000A25'  0A0E              .ADDRESS EXMODE_CODE
             00000A28'  0A12              .ADDRESS NOMODE_CODE                  ; This one is a dummy for $DEQ
                        0A16      NLMODE_CODE:
             4C 4E 00'  0A16              .ASCIC  /NL/
                    02  0A16
                        0A19      CRMODE_CODE:
             52 43 00'  0A19              .ASCIC  /CR/
                    02  0A19
                        0A1C      CWMODE_CODE:
             57 43 00'  0A1C              .ASCIC  /CW/
                    02  0A1C
                        0A1F      PRMODE_CODE:
             52 50 00'  0A1F              .ASCIC  /PR/
                    02  0A1F
                        0A22      PWMODE_CODE:
             57 50 00'  0A22              .ASCIC  /PW/
                    02  0A22
                        0A25      EXMODE_CODE:
             58 45 00'  0A25              .ASCIC  /EX/
                    02  0A25
                        0A28      NOMODE_CODE:
             6F 6E 00'  0A28              .ASCIC  /no/
                    02  0A28
                        0A2B
                        0A2B      GETSYI_ITMLST:                                 ; $GETSYI item list for...
             1067 0006  0A2B              .WORD   6,SYI$_SCSNODE               ; ...my node's name in a cluster
   00000000'0000009A'  0A2F              .ADDRESS SCSNODE,0
              00000000  0A37              .LONG   0
                        0A3B
                        0A3B      ; Set up data structures so that mailboxes which serve as SYS$INPUT to
                        0A3B      ; the processes we create are created in our group logical name table.
                        0A3B
                        0A3B      LNMPRCDIR:                                     ; Table name to force mbx logicals...
52 50 24 4D 4E 4C 00000A43'010E0000'  0A3B   .ASCID  /LNM$PROCESS_DIRECTORY/ ; ...to appear in a group table
54 43 45 52 49 44 5F 53 53 45 43 4F  0A49
                         59 52 4F  0A55
                        0A58
                        0A58      LNMTMPMBX:                                     ; Logical name which tells $CREMBX...
45 54 24 4D 4E 4C 00000A60'010E0000'  0A58   .ASCID  /LNM$TEMPORARY_MAILBOX/ ; ...where to put mbx logical names
4C 49 41 4D 5F 59 52 41 52 4F 50 4D  0A66
                         58 4F 42  0A72
                        0A75
                        0A75      LNMITMLST:                                     ; $CRELNM ITMLST naming where mbx...
            0002 0010'  0A75              .WORD   LNMGRPLEN,LNM$_STRING        ; ...logical name is to be defined
   00000000'000000D2'  0A79              .ADDRESS LNMGRPNUM,0
              00000000  0A81              .LONG   0
```

```
                          0A85          .SBTTL  Read/Write Data
                      00000000          .PSECT  RWDATA,WRT,NOEXE,PAGE
                          0000
                          0000  DRIVEN_DESC:                                  ; Name of driven process. Driven procs...
                0000 000B' 0000         .WORD   TEST_NAME_LEN+1,0             ; ...always have qualifier char at end
                 00000011' 0004         .ADDRESS TEST_NAME_I
                          0008  TEST_NAME_D:                                  ; Name of the test - .ASCID version
                0000 000A' 0008         .WORD   TEST_NAME_LEN,0
                 00000011' 000C         .ADDRESS TEST_NAME_I
                          0010  ; THESE NEXT ITEMS MUST REMAIN CONTIGUOUS!
                          0010  TEST_NAME_C:                                  ; Name of the test - .ASCIC version
                       0A' 0010         .BYTE   TEST_NAME_LEN
                          0011  TEST_NAME_I:                                  ; Name of the test - .ASCII version
  5F 30 30 4B 43 4F 4C 54 45 55 0011    .ASCII  /UETLOCK00_/
                  0000000A 001B TEST_NAME_LEN = .-TEST_NAME_I                 ; Length in chars of our test name
                          001B  PROC_QUALIFIER:                               ; Driven process qualifier appended...
                  0000001C 001B         .BLKB   1                             ; ...to test name
                          001C  ; End of contiguous items
                          001C
                          001C  MBXCHANS:                                     ; Mailbox channels to communicate with detac
                  00000020 001C         .BLKW   PROC_COUNT
                          0020
                          0020  PROCIDS:                                      ; Detached processes id
                  00000028 0020         .BLKL   PROC_COUNT
                  00000000 0028         .LONG   0                             ; End of the proc id table
                          002C
                          002C  LOG_FILE_DESC:                                ; Skeleton name for driven proc log files
                0000 000B' 002C         .WORD   LOG_FILE_LEN,0
                 00000034' 0030         .ADDRESS LOG_FILE_QUAL
                          0034  ; THE FOLLOWING ITEMS MUST REMAIN CONTIGUOUS!
                          0034  LOG_FILE_QUAL:                                ; Character to distinguish between procs
                        00 0034         .BYTE   0
  47 4F 4C 2E 54 53 45 54 4B 4C 0035    .ASCII  /LKTEST.LOG/
                  0000000B 003F LOG_FILE_LEN = .-LOG_FILE_QUAL
                          003F  ; End of items which must remain contiguous.
                          003F
                          003F  ENQLST:                                       ; ENQ(W) QIO arguments list
                          003F          $ENQ    RESNAM = RESR_DESC
                          006F
                          006F  DEQLST:                                       ; DEQ QIO argument list
                          006F          $DEQ
                          0083
                          0083  TST_COMMAND:                                  ; Test command from the test table
         00000000 00000000 0083         .LONG   0,0
                          008B
                          008B  TSTFLG:                                       ; Tells actions after calling sys svc
                      0000 008B         .WORD   0
                          008D
                          008D  RESR_DESC:                                    ; Resource name descriptor
                 00000011' 008D         .LONG   RESR_LEN
                 00000095' 0091         .ADDRESS .+4                          ; Note:  We assume all the following...
            24 50 54 45 55 0095         .ASCII  "UETP$"                       ; (Facility name, for convention's sake)
                          009A  SCSNODE:                                      ; (Node name, so unique within cluster)
                  000000A0 009A         .BLKB   6
                        5F 00A0         .ASCII  "_"                           ; (Separator)
                          00A1  RESR:                                         ; (Resource name)
            4B 43 4F 4C 3F 00A1         .ASCII  /?LOCK/                       ; ...are contiguous
```

```
                00000011  00A6  RESR_LEN = .-RESR_DESC-8
                          00A6
                          00A6  DLOCK_TIME:                               ; Delta time for $SETIMR to prevent...
                000000AA  00A6          .BLKL   1                         ; ...deadlock hangs
                FFFFFFFF  00AA          .LONG   -1
                          00AE
                          00AE  GETUIC:                                   ; Get UIC
                    0004  00AE          .WORD   4                         ; Length of recieve buffer
                    0304  00B0          .WORD   JPI$_UIC                  ; Request UIC
               000000CA'  00B2          .LONG   LK_UIC                    ; Buffer address
                00000000  00B6          .LONG   0                         ; No return length
                    0004  00BA          .WORD   4                         ; Get our base priority
                    0309  00BC          .WORD   JPI$_PRIB
               000000CE'  00BE          .ADDRESS BASE_PRI
                00000000  00C2          .LONG   0
                00000000  00C6          .LONG   0                         ; End of list
                          00CA
                          00CA  LK_UIC:                                   ; Put UIC here
                00000000  00CA          .LONG   0
                          00CE
                          00CE  BASE_PRI:                                 ; Parent process' base priority
                00000000  00CE          .LONG   0
                          00D2
                          00D2  LNMGRPNUM:                                ; Equivalence name giving the table...
 5F 50 55 4F 52 47 24 4D 4E 4C  00D2    .ASCII  /LNM$GROUP_/              ; ...name in which mbx logical names...
                0000000A  00DC          LNMGRP = .-LNMGRPNUM
       20'20'20'20'20'20' 00DC          .BYTE   ^A/ /[6]
                00000010  00E2          LNMGRPLEN = .-LNMGRPNUM           ; ...are to be put
                          00E2
                          00E2  LNMGRPNIL:                                ; Descriptor to convert...
               0000 0006  00E2          .WORD   LNMGRPLEN-LNMGRP,0
               000000DC'  00E6          .ADDRESS LNMGRPNUM+LNMGRP         ; ...UIC to text
                          00EA
                          00EA  DIBBUF_DESC:                              ; Device information buffer descriptor
               0000 0074  00EA          .WORD   DIB$K_LENGTH,0
               000000F2'  00EE          .ADDRESS DIBBUF
                          00F2  DIBBUF:                                   ; Device information block
                00000166  00F2          .BLKB   DIB$K_LENGTH
                          0166
                          0166  MBX_DESC:                                 ; Mailbox logical name descriptor
               00000006'  0166          .LONG   MBXNAM_LEN
               0000016E'  016A          .ADDRESS MBXNAM
                          016E  MBXNAM:                                   ; Mailbox logical name
       58 42 4D 4B 4C 3F  016E          .ASCII  /?LKMBX/
                00000006  0174  MBXNAM_LEN = .-MBXNAM
                          0174
                          0174  MBX_IOSB:                                 ; Mailbox QIO IO status block
       00000000 00000000  0174          .LONG   0,0
                          017C
                          017C  EX_MBXCHAN:                               ; Termination mailbox channel
                    0000  017C          .WORD   0
                          017E  EXIT_MSG:                                 ; Buffer for termination mailbox msg
                000001D2  017E          .BLKB   ACC$K_TERMLEN
                          01D2  EX_PROC_CNT:                              ; Exit process count
                00000002  01D2          .LONG   PROC_COUNT
                          01D6
                          01D6  TTCHAN:                                   ; Terminal channel
```

```
         00000000  01D6          .LONG    0
                   01DA
                   01DA  TYPE:                              ; System service type
         00000003  01DA          .LONG    3
                   01DE
                   01DE  GLBFLG:                            ; Holds flags which remain thru test
             0000  01DE          .WORD    0
                   01E0
                   01E0  LOCFLG:                            ; Local flag - flags for one TST_TABLE event
               00  01E0          .BYTE    0
                   01E1
                   01E1  EF_STATE:                          ; Event flag longword
         00000000  01E1          .LONG    0
                   01E5
                   01E5  SAVED_VAL:                         ; Saved value of first quad of value block
00000000 00000000  01E5          .LONG    0,0
                   01ED  LKID_ADDR:                         ; Lock id address
         00000000  01ED          .LONG    0
                   01F1  LKSB_ADDR:                         ; Lock status block address
         00000000  01F1          .LONG    0
                   01F5  LKSBS:                             ; Buffer for LKSB's
         00000465  01F5          .BLKQ    <3*26>
                   0465
                   0465  EXIT_DESC:                         ; Exit handler descrip
         00000000  0465          .LONG    0
        00000CAA'  0469          .ADDRESS EXIT_HANDLER
         00000001  046D          .LONG    1
        00000511'  0471          .ADDRESS STATUS
                   0475
                   0475  MSG_BLOCK:                         ; Message block for GETMSG
         00000479  0475          .BLKB    4
                   0479
                   0479  FAO_BUF:                           ; FAO output string descriptor
        0000 0084  0479          .WORD    TEXT_BUFFER,0
        00000489'  047D          .ADDRESS BUFFER
                   0481
                   0481  BUFFER_PTR:                        ; Text buffer descriptor
        0000 0084  0481          .WORD    TEXT_BUFFER,0
        00000489'  0485          .ADDRESS BUFFER
                   0489
                   0489  BUFFER:                            ; FAO output and other misc. buffer
         0000050D  0489          .BLKB    TEXT_BUFFER
                   050D
                   050D  ARG_COUNT:                         ; Argument count
         00000000  050D          .LONG    0
                   0511
                   0511  STATUS:                            ; Final status code
         00000000  0511          .LONG    0
                   0515
                   0515  QUAD_STATUS:                       ; IO status block for misc. sys. svcs.
00000000 00000000  0515          .QUAD    0
                   051D
                   051D  LOG_MSGPTR:                        ; $FAO arg for $PUTMSG when copying...
        0000 0000  051D          .WORD    0,0               ; ...the driven proc log file...
        00000525'  0521          .ADDRESS RMS_BUFFER        ; ...to our SYS$OUTPUT
                   0525
                   0525  RMS_BUFFER:                        ; Log file record buffer
```

```
000005A9  0525   771          .BLKB   TEXT_BUFFER
```

```
05A9          .SBTTL  RMS-32 Data Structures
05A9
05A9          .ALIGN  LONG
05AC
05AC  LOG_FAB:
05AC          $FAB -
05AC          FNA = LOG_FILE_QUAL,-
05AC          FNS = LOG_FILE_LEN,-
05AC          ORG = <SEQ>,-
05AC          MRS = TEXT_BUFFER
05FC
05FC  LOG_RAB:
05FC          $RAB -
05FC          FAB = LOG_FAB,-
05FC          UBF = RMS_BUFFER+INDENT,-
05FC          USZ = TEXT_BUFFER
0640
0640  INPUT_FAB:                                   ; Reads SYS$INPUT
0640          $FAB -
0640          FNM = <SYS$INPUT:>
0690
0690  INPUT_RAB:                                   ; Reads SYS$INPUT
0690          $RAB -
0690          FAB = INPUT_FAB,-
0690          UBF = TST_COMMAND,-
0690          USZ = COMMAND_SIZE
```

```
                        06D4        .SBTTL  TESTLOCK - Initialization
                    00000000        .PSECT  TESTLOCK,EXE,NOWRT,PAGE
                        0000        .DEFAULT DISPLACEMENT,WORD
                        0000
                        0000   ;+
                        0000   ;    This module serves as both the controlling (driver) and controlled
                        0000   ;    (driven) process of the Lock Manager test.  Initialize and set up
                        0000   ;    those things which are common to both processes - overhead, determining
                        0000   ;    our function, timers and interprocess communication.
                        0000   ;-
                  0000  0000   .ENTRY  UETLOCK00,^M<>                   ; Entry mask
                        0002
      6D  0A2F'CF   DE  0002        MOVAL   SSERROR,(FP)                ; Declare exception handler
                        0007        $SETSFM_S ENBFLG = #1               ; Enable system service failure mode
                        0010        $DCLEXH_S DESBLK = EXIT_DESC        ; Declare exit handler
                        001B   ;
                        001B   ; Determine if we are the master (driver) process or one of the slave (driven)
                        001B   ; processes.  If we are a driven process, determine which one.  Use that info
                        001B   ; to form our process name and announce us to the world.
                        001B   ;
                        001B        $OPEN   FAB = INPUT_FAB             ; See what sort of thing SYS$INPUT is
           76 50    E9  0026        BLBC    R0,10$                      ; If we've no SYS$INPUT, we are driver
              14    E1  0029        BBC     #DEV$V_MBX,-                ; If SYS$INPUT is not a mailbox...
      70 0680'CF        002B                INPUT_FAB+FAB$L_DEV,10$     ; ...we are driver
                        002F        $CONNECT RAB = INPUT_RAB,-          ; Check further to see if we're driven
                        002F                ERR = RMS_ERROR
                        003E        $GET    RAB = INPUT_RAB,-
                        003E                ERR = RMS_ERROR
         02DC'CF    29  004D        CMPC3   SUBPROC_STRING,-            ; See if mailbox message...
         02E4'CF        0051                SUBPROC_STRING+8,-          ; ...is the one we expect...
         0083'CF        0054                TST_COMMAND                 ; ...to tell us which process we are
              46    12  0057        BNEQ    10$                         ; BR if no match - we're driver
   001B'CF    63    90  0059        MOVB    (R3),PROC_QUALIFIER         ; We're driven proc - copy which one
         0008'CF    B6  005E        INCW    TEST_NAME_D                 ; Attach it to our name...
         0010'CF    96  0062        INCB    TEST_NAME_C                 ; ...in all places we need it
   01DE'CF    01    88  0066        BISB2   #DRIVEN_M,GLBFLG            ; Remember that we're a driven process
                        006B                                            ; We do only what we're told to do
02A9'CF   02  63    3A  006B        LOCC    (R3),#PROC_COUNT,PROCS      ; Have we a legal process name?
              34    12  0071        BNEQ    20$                         ; BR if so - we're all set up
                        0073        $FAO_S  CTRSTR = ILL_PROC_NAME,-    ; Bad name - quit
                        0073                OUTLEN = BUFFER_PTR,-
                        0073                OUTBUF = FAO_BUF,-
                        0073                P1     = #1,-
                        0073                P2     = #PROC_QUALIFIER
         0481'CF    DF  008E        PUSHAL  BUFFER_PTR
              01    DD  0092        PUSHL   #1
     00741132 8F    DD  0094        PUSHL   #UETP$_TEXT!STS$K_ERROR
              03    DD  009A        PUSHL   #3
            0B63    31  009C        BRW     ERROR_EXIT
                        009F   10$:
         0008'CF    B7  009F        DECW    TEST_NAME_D                 ; Fix up our name...
         0010'CF    97  00A3        DECB    TEST_NAME_C                 ; ...wherever it is needed
                        00A7        ;BICB2  #DRIVEN_M,GLBFLG            ; Remember that we're a driver process
                        00A7                                            ; We tell other processes what to do
                        00A7   20$:
                        00A7        $SETPRN_S PRCNAM = TEST_NAME_D      ; Set our process name
                        00B2
```

```
              022C'CF   DF  00B2          PUSHAL  BEGUN_ADDR
              0010'CF   DF  00B6          PUSHAL  TEST_NAME_C
                   02   DD  00BA          PUSHL   #2                      ; Argument count
          007480D9 8F   DD  00BC          PUSHL   #UETP$_SATSMS!STS$K_SUCCESS ; Set the begin message code
        00000000'GF 04  FB  00C2          CALLS   #4,G^LIB$SIGNAL         ; Print the UETP begin time message
           01DE'CF 04   A8  00C9          BISW2   #BEGIN_MSGM,GLBFLG      ; Indicate that sentinel was given
                            00CE  ;
                            00CE  ; Set timer to catch hangs waiting for another process or for any hangs we
                            00CE  ; may encounter ourself.
                            00CE  ;
                            00CE          $SETIMR_S DAYTIM = THREE_MIN,-  ; Set 3 minute timer
                            00CE                  ASTADR = TIME_OUT,-
                            00CE                  REQIDT = #1
                            00E1  ;
                            00E1  ; Enable control C handler if run from terminal.
                            00E1  ;
                   02   E1  00E1          BBC     S^#DEV$V_TRM,-          ; BR if SYS$INPUT is NOT a terminal
          66 0680'CF        00E3                  INPUT_FAB+FAB$L_DEV,50$
                            00E7          $GETDVI_S DEVNAM = SYS$INPUT,-      ; Get the name of...
                            00E7                  EFN    = #SS_SYNCH_EFN,- ; ...device which may abort test
                            00E7                  ITMLST = INPUT_ITMLST,-
                            00E7                  IOSB   = QUAD_STATUS
          45 0515'CF    E9  0103          BLBC    QUAD_STATUS,50$         ; Avoid CTRL/C handler if any error
                            0108          $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C AST handling
                            0108                  CHAN   = TTCHAN
                            0119          $QIOW_S CHAN   = TTCHAN,-       ; Enable CTRL/C AST's...
                            0119                  FUNC   = #IO$_SETMODE!IO$M_CTRLCAST,-
                            0119                  P1     = CCASTHAND
              0008'CF   DF  013A          PUSHAL  TEST_NAME_D             ; ...and tell the user...
                   01   DD  013E          PUSHL   #1                      ; ...
          0074832B 8F   DD  0140          PUSHL   #UETP$_ABORTC!STS$K_SUCCESS ; ...how to abort gracefully...
        00000000'GF 03  FB  0146          CALLS   #3,G^LIB$SIGNAL         ; ...
                            014D  ;
                            014D  ; Create and associate with a common event flag block.  This will be used to
                            014D  ; signal that a particular operation of a driven process has been completed.
                            014D  ;
                            014D  50$:
                            014D          $ASCEFC_S EFN = #DONE_CEF,-     ; Create and assoc common EF cluster
                            014D                  NAME = LK_CEF_DESC
                            0162  ;
                            0162  ; The logical name MODE is available to knowledgeable users to invoke special
                            0162  ; actions in this test.
                            0162  ;
                            0162          $TRNLOG_S LOGNAM = MODE,-       ; If MODE translates...
                            0162                  RSLLEN = BUFFER_PTR,-
                            0162                  RSLBUF = FAO_BUF
  0489'CF   504D5544 8F D1  017B          CMPL    #^A/DUMP/,BUFFER        ; ...to "DUMP"...
                   05   12  0184          BNEQ    60$
           01DE'CF 02   A8  0186          BISW2   #DUMP_M,GLBFLG          ; ...then we type a running log
                            018B  60$:
                            018B          $GETSYI_S ITMLST = GETSYI_ITMLST ; Get my node's node name
                            01A0  ;
                            01A0  ; All things common to the driver and driven processes have been done.  Split
                            01A0  ; up and get on to the main work of the test.
                            01A0  ;
        03 01DE'CF 00   E1  01A0          BBC     #DRIVEN_V,GLBFLG,DRIVER ; BR if we are the driver process
                 010F   31  01A6          BRW     DRIVEN                  ; We are a driven process
```

```
                          01A9          .SBTTL  TESTLOCK - Driver Process
                          01A9  ;+
                          01A9  ;       This is the controlling process of the lock manager UETP test.  It
                          01A9  ;       creates detached, "cooperating" process(es), then, based on the entries
                          01A9  ;       in TST_TABLE, either calls Lock Manager system services or sends mail
                          01A9  ;       to (one or more) cooperating process(es) to have the process(es) call
                          01A9  ;       a Lock Manager system service.  Results are checked.
                          01A9  ;-
                          01A9  DRIVER:
                          01A9          $CREMBX_S CHAN = EX_MBXCHAN,-   ; Create and associate termination mbx
                          01A9                  MAXMSG = #ACC$K_TERMLEN,-
                          01A9                  BUFQUO = #256
                          01C6          $GETCHN_S CHAN = EX_MBXCHAN,-   ; Get channel infomation - we need...
                          01C6                  PRIBUF = DIBBUF_DESC    ; ...the mailbox unit number
                          01DC          $GETJPI_S ITMLST = GETUIC       ; Get our UIC and our base priority
                02    DD  01F1          PUSHL   #2                      ; We'll convert a 2-byte...
   7E    00E2'CF    3C  01F3          MOVZWL  LNMGRPNIL,-(SP)         ; ...integer to ASCII...
         00E2'CF    DF  01F8          PUSHAL  LNMGRPNIL
         00CC'CF    DF  01FC          PUSHAL  LK_UIC+2                ; ...using group number of...
00000000'GF    04   FB  0200          CALLS   #4,G^OTS$CVT_L_TO       ; ...the process which runs us
                          0207          $CRELNM_S TABNAM = LNMPRCDIR,-  ; Force SYS$INPUT mbx name...
                          0207                  LOGNAM = LNMTMPMBX,-    ; ...to appear in...
                          0207                  ITMLST = LNMITMLST      ; ...a group logical name table
            00D9    30  021E          BSBW    CREATE_PROCS            ; Go subroutine to create detached procs
                          0221  ;
                          0221  ; Have an outstanding read to the termination mailbox, should any detached
                          0221  ; process end
                          0221  ;
                          0221          $QIO_S  CHAN = EX_MBXCHAN,-     ; Read termination mailbox
                          0221                  FUNC = #IO$_READVBLK,-
                          0221                  ASTADR = EX_MBX_AST,-
                          0221                  IOSB = MBX_IOSB,-
                          0221                  P1 = EXIT_MSG,-
                          0221                  P2 = #ACC$K_TERMLEN
                          024C  ;
                          024C  ; For each line in TEST_TABLE, either execute the command directly or send a
                          024C  ; message to one of the driven processes.  Wait for the driven process to
                          024C  ; complete the command, if necessary.
                          024C  ;
      59    0000'CF    DE  024C          MOVAL   TEST_TABLE,R9           ; Address of beginning of test table
                          0251  GET_COMMAND:
   0083'CF    69    7D  0251          MOVQ    (R9),TST_COMMAND        ; Get an entry of the test table
02A8'CF    0083'CF    91  0256          CMPB    TST_COMMAND,ALL_PROCS   ; Is it for self?
              3A    13  025D          BEQL    20$                     ; BR if yes
                          025F
                          025F          $CLREF_S EFN = #DONE_CEF        ; Clear CEF which says command is done
            011A    30  026C          BSBW    SEND_MSG                ; Send command to the detached process
   0F 0089'CF    04    E1  026F          BBC     #DEADLK_V,TST_COMMAND+6,10$ ; BR if not deadlock special case
                          0275          $WFLAND_S EFN = #DONE_CEF,-     ; Deadlock situation has been set up...
                          0275                  MASK = #DLMASK          ; ...we wait until it's resolved
                          0284  10$:
   19 0089'CF    06    E0  0284          BBS     #NOWAIT_V,TST_COMMAND+6,30$ ; BR if no_wait_done_cef bit set
                          028A          $WAITFR_S EFN = #DONE_CEF       ; Wait until another process finish
                          0297                                          ; executing the test command
              0A    11  0297          BRB     30$                     ; We've finished this table entry
                          0299  20$:
   0084'CF    03    91  0299          CMPB    #ENDTEST,TST_COMMAND+1  ; Is this endtest for driver process?
```

```
          03   13  029E              BEQL    30$                     ; BR if so - we'll really end when...
                   02A0                                              ; ...the test table is finished
        014F   30  02A0              BSBW    EXECUTE                 ; Go execute the test command
                   02A3      30$:
   59  08 A9   DE  02A3              MOVAL   COMMAND_SIZE(R9),R9     ; Get address of next test table entry
          69   D5  02A7              TSTL    (R9)                    ; Is it end of the table?
          03   13  02A9              BEQL    WAIT_PRCS               ; BR if yes
        FFA3   31  02AB              BRW     GET_COMMAND             ; Next entry
                   02AE
                   02AE      WAIT_PRCS:
                   02AE              $HIBER_S                        ; Hibernate until created procs exit
        0021   31  02B5              BRW     SUC_EXIT                ; The test is over, let's go home
```

```
               02B8          .SBTTL  TESTLOCK - Driven Process
               02B8     ;+
               02B8     ;        This is the controlled process of the Lock Manager UETP test.  It
               02B8     ;        is created by te driver process, then waits for commands from the
               02B8     ;        driver process to tell it how to call Lock Manager system services.
               02B8     ;        In some cases, results are checked to see if they are what we expect.
               02B8     ;        Note that SYS$INPUT is already defined as the way commands are passed
               02B8     ;        by the driver program.
               02B8     ;-
               02B8     DRIVEN:
               02B8     GET_NEXTMSG:
               02B8              $GET     RAB = INPUT_RAB,-          ; Read a command
               02B8                       ERR = RMS_ERROR
    0128   30  02C7              BSBW     EXECUTE                    ; Perform it
               02CA
               02CA              $SETEF_S EFN = #DONE_CEF            ; Tell the driver process we're done
      DF   11  02D7              BRB      GET_NEXTMSG
```

```
                          02D9       .SBTTL  Clean Up and Exit
                          02D9 ;+
                          02D9 ;     Arrive here from either the driver or the driven process.  Process
                          02D9 ;     exit or failure is the same in either case.
                          02D9 ;-
                          02D9 SUC_EXIT:                                    ; Successfully exit
10000000'8F     D0        02D9       MOVL    #SS$_NORMAL!STS$M_INHIB_MSG,-
    0511'CF               02DF               STATUS
                          02E2       $EXIT_S STATUS                       ; Exit with status
                          02ED
                          02ED ; To use FAIL_OUT, push to the stack the address of an error message and
                          02ED ; BRW FAIL_OUT.
                          02ED FAIL_OUT:                                    ; Failure exit
         01     DD        02ED       PUSHL   #1                           ; Arg count
 00741132 8F    DD        02EF       PUSHL   #UETP$_TEXT!STS$K_ERROR      ; UETP error code
         03     DD        02F5       PUSHL   #3                           ; Argument count
       0908     31        02F7       BRW     ERROR_EXIT                   ; Error exit
```

```
                              02FA          .SBTTL  Create Detached Processes and Mailboxes
                              02FA     ;++
                              02FA     ; FUNCTIONAL DESCRIPTION:
                              02FA     ;         Create detached processes and associated mailboxes routine.  Such
                              02FA     ;         processes are called driven processes because they only perform what
                              02FA     ;         this, the driver process, tells them to do.  Detached processes with
                              02FA     ;         the same UIC are created according to the process indicators (Q, R,
                              02FA     ;         etc.) stored at addresses beginning PROCS.  The created processes
                              02FA     ;         will run the same image as this, but will be able to distinguish itself
                              02FA     ;         from this process by the content of the mailbox which is its SYS$INPUT.
                              02FA     ;         There are mailboxes associated with each created process as
                              02FA     ;         communication channels.
                              02FA     ;
                              02FA     ; CALLING SEQUENCE:
                              02FA     ;         BSBW    CREATE_PROCS
                              02FA     ;
                              02FA     ; INPUT PARAMETERS:
                              02FA     ;         NONE
                              02FA     ;
                              02FA     ; IMPLICIT INPUTS:
                              02FA     ;         Process name table, PROCS
                              02FA     ;
                              02FA     ; OUTPUT PARAMETERS:
                              02FA     ;
                              02FA     ;         NONE
                              02FA     ; IMPLICIT OUTPUTS:
                              02FA     ;         Exit if error
                              02FA     ;
                              02FA     ; COMPLETION CODES:
                              02FA     ;         NONE
                              02FA     ;
                              02FA     ; SIDE EFFECTS:
                              02FA     ;         Creates detached, cooperating processes
                              02FA     ;--
                              02FA     CREATE_PROCS:
         02DC'CF         28   02FA          MOVC3   SUBPROC_STRING,-           ; Initialize mailbox message...
         02E4'CF              02FE                  SUBPROC_STRING+8,-         ; ...that will tell driven proc...
         0083'CF              0301                  TST_COMMAND                ; ...which one it is
         56   53         D0   0304          MOVL    R3,R6                      ; Save this pointer into TST_COMMAND
    52   02A9'CF         DE   0307          MOVAL   PROCS,R2                   ; Addr of process name indicators
    53   001C'CF         DE   030C          MOVAL   MBXCHANS,R3                ; Addr of associated mailbox channel
    54   0020'CF         DE   0311          MOVAL   PROCIDS,R4                 ; Addr of process ids
              55         D4   0316          CLRL    R5                         ; Indexes into process qualifier table
                              0318     10$:
  001B'CF     6245       90   0318          MOVB    (R2)[R5],PROC_QUALIFIER    ; Set up process name
  016E'CF     6245       90   031E          MOVB    (R2)[R5],MBXNAM            ; Set up assoc mailbox logical name
  0034'CF     6245       90   0324          MOVB    (R2)[R5],LOG_FILE_QUAL     ; Set up log file name
       66     6245       90   032A          MOVB    (R2)[R5],(R6)              ; Set up command to say "driven proc"
                              032E
                              032E          $CREMBX_S CHAN = (R3)[R5],-        ; Create mailbox and assign channel
                              032E                  MAXMSG = #COMMAND_SIZE,-   ; Maximum message length in bytes
                              032E                  BUFQUO = #512,-            ; Buffer quota
                              032E                  LOGNAM = MBX_DESC          ; Logical name 'Q'LKMBX,'R'LKMBX, etc.
                              034A
                              034A          $CREPRC_S PIDADR = (R4)[R5],-      ; Create detached process
                              034A                  IMAGE = LKTEST_DESC,-      ; Image
                              034A                  OUTPUT = LOG_FILE_DESC,-   ; SYS$OUTPUT
```

```
                    034A  1078              INPUT  = MBX_DESC,-    ; SYS$INPUT
                    034A  1079              PRCNAM = DRIVEN_DESC,- ; Proc name
                    034A  1080              UIC = LK_UIC,-         ; Same UIC as this controller
                    034A  1081              BASPRI = BASE_PRI,-    ; Same base priority
                    034A  1082              MBXUNT = DIBBUF+DIB$W_UNIT ; Termination mailbox unit number
                    037B  1083
      7E   6345  3C 037B  1084      MOVZWL  (R3)[R5],-(SP)          ; Send the driven proc's first command
 03AC'CF     01  FB 037F  1085      CALLS   #1,MBX_QIO              ; Tell it that it's a driven process!
                    0384  1086
   90 55     02  F2 0384  1087      AOBLSS  #PROC_COUNT,R5,10$      ; Next process if any more
                 05 0388  1088      RSB
```

```
                              0389        .SBTTL  Determine Which Driven Process Gets Command
                              0389   ;++
                              0389   ; FUNCTIONAL DESCRIPTION:
                              0389   ;       Send message to cooperative processes routine
                              0389   ;
                              0389   ; CALLING SEQUENCE:
                              0389   ;       BSBW    SEND_MSG
                              0389   ;
                              0389   ; INPUT PARAMETERS:
                              0389   ;       NONE
                              0389   ;
                              0389   ; IMPLICIT INPUTS:
                              0389   ;       Message is in TST_COMMAND
                              0389   ;
                              0389   ; OUTPUT PARAMETERS:
                              0389   ;       NONE
                              0389   ;
                              0389   ; IMPLICIT OUTPUTS:
                              0389   ;       Mailbox message sent to the designated process
                              0389   ;
                              0389   ; COMPLETION CODES:
                              0389   ;       In STATUS if error
                              0389   ;
                              0389   ; SIDE EFFECTS:
                              0389   ;       NONE
                              0389   ;--
                              0389
                              0389   SEND_MSG:
                   55   D4    0389          CLRL    R5                      ; Index into driven processes
                              038B   10$:                                   ; Which process?
  02A9'CF45   0083'CF   91    038B          CMPB    TST_COMMAND,PROCS[R5]   ; Compare process indicator
                   0B   13    0393          BEQL    20$                     ; BR if equal
            F2 55  02   F2    0395          AOBLSS  #PROC_COUNT,R5,10$      ; Try again if more process indicators
              0325'CF   DF    0399          PUSHAL  ERR_IN_TABLE            ; Error message
                 FF4D   31    039D          BRW     FAIL_OUT                ; Failure out
                              03A0   20$:
       7E   001C'CF45   3C    03A0          MOVZWL  MBXCHANS[R5],-(SP)
         03AC'CF   01   FB    03A6          CALLS   #1,MBX_QIO              ; Send the message
                        05    03AB          RSB
```

```
                          03AC          .SBTTL  Send Messages to Driven Processes
                          03AC    ;++
                          03AC    ; FUNCTIONAL DESCRIPTION:
                          03AC    ;       This routine is called to send a message to any one of the driven
                          03AC    ;       processes.  It does a $QIO and checks to see that the message was sent
                          03AC    ;       correctly.
                          03AC    ;
                          03AC    ; CALLING SEQUENCE:
                          03AC    ;       CALLx   #1,MBX_QIO
                          03AC    ;
                          03AC    ; INPUT PARAMETERS:
                          03AC    ;       04(AP) has the channel by which we access the mailbox
                          03AC    ;
                          03AC    ; IMPLICIT INPUTS:
                          03AC    ;       NONE
                          03AC    ;
                          03AC    ; OUTPUT PARAMETERS:
                          03AC    ;       NONE
                          03AC    ;
                          03AC    ; IMPLICIT OUTPUTS:
                          03AC    ;       Message written to mailbox.
                          03AC    ;
                          03AC    ; COMPLETION CODES:
                          03AC    ;       Result of $QIO
                          03AC    ;
                          03AC    ; SIDE EFFECTS:
                          03AC    ;       Program terminates if an error occurs.
                          03AC    ;
                          03AC    ;--
                          03AC
                          03AC    MBX_QIO:
                   0FFC   03AC          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                          03AE
                          03AE          $QIOW_S EFN = #SNDMSG_EFN,-     ; Send the message
                          03AE                  CHAN = 04(AP),-
                          03AE                  FUNC = #IO$_WRITEVBLK,-
                          03AE                  IOSB = MBX_IOSB,-
                          03AE                  P1 = TST_COMMAND,-
                          03AE                  P2 = #COMMAND_SIZE
         01 0174'CF  E9   03D0          BLBC    MBX_IOSB,10$            ; BR if IOSB is not right
                     04   03D5          RET                             ; Return
                          03D6    10$:
0511'CF    0174'CF   3C   03D6          MOVZWL  MBX_IOSB,STATUS         ; Use IOSB status as exit status...
           0511'CF   DD   03DD          PUSHL   STATUS                  ; ...and a message of complaint
           036E'CF   DF   03E1          PUSHAL  SNDMBX_ERRMSG
                01   DD   03E5          PUSHL   #1
       00741132 8F   DD   03E7          PUSHL   #UETP$_TEXT!STS$K_ERROR
                04   DD   03ED          PUSHL   #4
              0810   31   03EF          BRW     ERROR_EXIT
```

```
                         03F2         .SBTTL  Execute Test Commands in the Test Table
                         03F2   ;++
                         03F2   ; FUNCTIONAL DESCRIPTION:
                         03F2   ;       Decode the test command and set up the service parameters accordingly,
                         03F2   ;       Execute the service (ENQ,ENQW,DEQ) requested
                         03F2   ;
                         03F2   ; CALLING SEQUENCE:
                         03F2   ;       BSBW    EXECUTE
                         03F2   ;
                         03F2   ; INPUT PARAMETERS:
                         03F2   ;       NONE
                         03F2   ;
                         03F2   ; IMPLICIT INPUTS:
                         03F2   ;       Test command stored in TST_COMMAND quadword
                         03F2   ;
                         03F2   ; OUTPUT PARAMETERS:
                         03F2   ;       NONE
                         03F2   ;
                         03F2   ; IMPLICIT OUTPUTS:
                         03F2   ;       Address of LKSB in R3
                         03F2   ;
                         03F2   ; COMPLETION CODES:
                         03F2   ;       In STATUS if error
                         03F2   ;
                         03F2   ; SIDE EFFECTS:
                         03F2   ;       Exit if error,
                         03F2   ;       BSBW CHECK_UP if ENQW to check correctness
                         03F2   ;       An AST may be generated for some locks
                         03F2   ;--
                         03F2
                         03F2   EXECUTE:
         01E0'CF   94    03F2           CLRB    LOCFLG                  ; Clear all local flags
     56  0083'CF   DE    03F6           MOVAL   TST_COMMAND,R6          ; Addr of the test command
              86   95    03FB           TSTB    (R6)+                   ; Skip the process indicator
    01DA'CF   86   9A    03FD           MOVZBL  (R6)+,TYPE              ; Get type of service, ENQ, ENQW or DEQ
    01DA'CF   03   D1    0402           CMPL    #ENDTEST,TYPE           ; Is it end of test for driven process?
              03   12    0407           BNEQ    10$                     ; BR if not
            FECD   31    0409           BRW     SUC_EXIT                ; We're done if it is
                         040C   10$:
  05 01DE'CF  01   E1    040C           BBC     #DUMP_V,GLBFLG,20$      ; BR if we don't dump every command
     052F'CF  00   FB    0412           CALLS   #0,DUMP_COMMAND         ; Type every command to SYS$OUTPUT
                         0417   20$:
     57  003F'CF   DE    0417           MOVAL   ENQLST,R7               ; Assume ENQ(W), Addr of ENQ arg list
           04 A7   D4    041C           CLRL    ENQ$_EFN(R7)            ; Assume EFN = 0
      08 A7   86   9A    041F           MOVZBL  (R6)+,ENQ$_LKMODE(R7)   ; Get lock mode
          52  86   9A    0423           MOVZBL  (R6)+,R2                ; Get resource indicator
    00A1'CF   52   90    0426           MOVB    R2,RESR                 ; Set resource name
52  00000041 8F    C2    042B           SUBL2   #^A/A/,R2               ; Index of associated LKSB
          52  18   C4    0432           MULL2   #LKSB_SIZE,R2           ; Displacement of the LKSB from LKSBS
     53  01F5'C2   DE    0435           MOVAL   LKSBS(R2),R3            ; Address of associated LKSB
    01F1'CF   53   D0    043A           MOVL    R3,LKSB_ADDR            ; Store the addr of LKSB
 01ED'CF   04 A3   DE    043F           MOVAL   4(R3),LKID_ADDR         ; Address of the LOCK ID
  02     01DA'CF   D1    0445           CMPL    TYPE,#DEQ               ; Is it DEQ?
              03   12    044A           BNEQ    30$                     ; BR if not
            00CD   31    044C           BRW     DEQS                    ; Yes, go dequeue
                         044F   30$:
     0C A7    53   D0    044F           MOVL    R3,ENQ$_LKSB(R7)        ; Address of LKSB
```

```
         54   86  9A 0453       MOVZBL  (R6)+,R4                        ; Parent lock indicator
            18 A7  D4 0456       CLRL    ENQ$_PARID(R7)                  ; Assume no parent lock
         20   54  91 0459       CMPB    R4,#^A/ /                       ; Is there parent lock?
              14  13 045C       BEQL    40$                             ; BR if no
 54   00000041 8F  C2 045E       SUBL2   #^A/A/,R4                       ; Index of LKSB for parent lock
            54 18  C4 0465       MULL2   #24,R4                          ; Displacement of LKSB of parent lock
      54   01F5'C4 DE 0468       MOVAL   LKSBS(R4),R4                    ; Address of LKSB of parent lock
   18 A7     04 A4 D0 046D       MOVL    4(R4),ENQ$_PARID(R7)            ; Parent lock id
                     0472 40$:
      10 A7   86  9A 0472       MOVZBL  (R6)+,ENQ$_FLAGS(R7)            ; Get enq flag
   0D 10 A7   03  E1 0476       BBC     #LCK$V_SYNCSTS,ENQ$_FLAGS(R7),50$ ; BR if not SYNCSTS flag test
      04 A7   01  D0 047B       MOVL    #SYNC_EF,ENQ$_EFN(R7)           ; Set up EFN for SYNCSTS test
                     047F       $CLREF_S EFN = #SYNC_EF                 ; We need to initialize the efn for it
                     0488 50$:
 1C A7   07D6'CF  DE 0488       MOVAL   COMP_AST,ENQ$_ASTADR(R7)        ; Addr of completion AST
 20 A7   01ED'CF  D0 048E       MOVL    LKID_ADDR,ENQ$_ASTPRM(R7)       ; Set AST param to be addr of LKID
     008B'CF   86  B0 0494       MOVW    (R6)+,TSTFLG                    ; Get test flag
  03 008B'CF   05  E1 0499       BBC     #NOCAST_V,TSTFLG,60$            ; BR if no_completion_ast bit not set
           1C A7   D4 049F       CLRL    ENQ$_ASTADR(R7)                 ; No completion AST
                     04A2 60$:
           24 A7   D4 04A2       CLRL    ENQ$_BLKAST(R7)                 ; Assume blocking ast not requested
  06 008B'CF   00  E1 04A5       BBC     #BLKAST_V,TSTFLG,70$            ; BR if block_ast bit not set
 24 A7   0807'CF  DE 04AB       MOVAL   BLOCK_AST,ENQ$_BLKAST(R7)       ; Set address of blocking AST routine
                     04B1 70$:
  46 008B'CF   04  E1 04B1       BBC     #DEADLK_V,TSTFLG,80$            ; BR if not deadlock special case
                     04B7       $CLREF_S EFN = #DLDET_CEF               ; Deadlock must be detected and...
                     04C4       $CLREF_S EFN = #DLRES_CEF               ; ...resolved. One flag per condition
 1C A7   083F'CF  DE 04D1       MOVAL   DLOCK_AST,ENQ$_ASTADR(R7)       ; Deadlock gets special AST
     FBD3E280 8F  C5 04D7       MULL3   #-10000000*<PROC_COUNT+5>,-     ; To prevent hanging forever...
 51   00000000'GF     04DD               G^LCK$GL_WAITTIME,R1            ; ...while waiting for...
    00A6'CF   51  D0 04E3       MOVL    R1,DLOCK_TIME                   ; ...deadlock detection....
                     04E8       $SETIMR_S DAYTIM = DLOCK_TIME,-         ; ...we set a timer...
                     04E8                 ASTADR = DLOCK_TO_AST,-       ; ...which will get us out...
                     04E8                 REQIDT = LKID_ADDR            ; ...if need be
                     04FD 80$:
    0D 01DA'CF   E8 04FD       BLBS    TYPE,ENQWS                      ; Dispatch for correct system service
                     0502                                               ; (Correctness of TYPE guaranteed by...
                     0502                                               ; ...assembly time checks)
                     0502 ENQS:
                     0502       $ENQ_G  ENQLST                          ; Enqueue
           028E   30 050B       BSBW    CHK_SS                          ; Check R0 to see if we queued up OK
                  05 050E       RSB                                     ; Return
                     050F ENQWS:
                     050F       $ENQW_G ENQLST                          ; Enqueue and wait for EFN
           008C   30 0518       BSBW    CHECK_UP                        ; Go check final status
                  05 051B       RSB                                     ; Return
                     051C DEQS:
 0073'CF     04 A3 D0 051C       MOVL    4(R3),DEQLST+DEQ$_LKID          ; Lock id of the lock to be dequeued
                     0522       $DEQ_G  DEQLST                          ; Dequeue
           026E   30 052B       BSBW    CHK_SS                          ; Check R0 to see if we dequeued OK
                  05 052E       RSB                                     ; Return
```

```
                        052F          .SBTTL  Type the Current Command
                        052F     ;++
                        052F     ; FUNCTIONAL DESCRIPTION:
                        052F     ;        Type the current command to be executed.
                        052F     ;
                        052F     ; CALLING SEQUENCE:
                        052F     ;        CALLS   #0,DUMP_COMMAND
                        052F     ;
                        052F     ; INPUT PARAMETERS:
                        052F     ;        NONE
                        052F     ;
                        052F     ; IMPLICIT INPUTS:
                        052F     ;        TST_COMMAND filled with next command to execute
                        052F     ;
                        052F     ; OUTPUT PARAMETERS:
                        052F     ;        NONE
                        052F     ;
                        052F     ; IMPLICIT OUTPUTS:
                        052F     ;        NONE
                        052F     ;
                        052F     ; COMPLETION CODES:
                        052F     ;        NONE
                        052F     ;
                        052F     ; SIDE EFFECTS:
                        052F     ;        Message to SYS$OUTPUT
                        052F     ;--
                        052F
                        052F     DUMP_COMMAND:
                  0FFC  052F          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0531
51   0084'CF      9A    0531          MOVZBL  TST_COMMAND+1,R1          ; Get the type of SS call
52   0085'CF      9A    0536          MOVZBL  TST_COMMAND+2,R2          ; Get the kind of lock we tried
     51   02      D1    053B          CMPL    #DEQ,R1                   ; Were we doing a $DEQ?
          03      12    053E          BNEQ    10$                       ; BR if not
     52   06      D0    0540          MOVL    #LCK$K_EXMODE+1,R2        ; No mode for $DEQ, say that
                        0543     10$:
     53   01      D0    0543          MOVL    #1,R3                     ; Assume that we've a parent lock
54   0087'CF      DE    0546          MOVAL   TST_COMMAND+4,R4
     64   20      91    054B          CMPB    #^A7 /,(R4)               ; But have we?
          08      12    054E          BNEQ    20$                       ; BR if we have
     53   04      D0    0550          MOVL    #NONE_LENGTH,R3           ; We have none...
54   09DA'CF      DE    0553          MOVAL   NONE,R4                   ; ...supply appropriate message
                        0558     20$:
                        0558          $FAO_S  CTRSTR = DUMP_MSG,-       ; Format the TEST_TABLE entry...
                        0558                  OUTLEN = BUFFER_PTR,-     ; ...into something humanly readable
                        0558                  OUTBUF = FAO_BUF,-
                        0558                  P1     = #1,-
                        0558                  P2     = #TST_COMMAND+0,-
                        0558                  P3     = TEST_CODES[R1],-
                        0558                  P4     = #1,-
                        0558                  P5     = #TST_COMMAND+3,-
                        0558                  P6     = LOCK_MODES[R2],-
                        0558                  P7     = TST_COMMAND+5,-
                        0558                  P8     = R3,-
                        0558                  P9     = R4,-
                        0558                  P10    = TST_COMMAND+6,-
                        0558                  P11    = #0
```

```
          0481'CF   DF  0593  1346          PUSHAL  BUFFER_PTR
               01   DD  0597  1347          PUSHL   #1
       00741133 8F  DD  0599  1348          PUSHL   #UETP$_TEXT!STS$K_INFO
    00000000'GF  03 FB  059F  1349          CALLS   #3,G^LIB$SIGNAL
                    04  05A6  1350          RET
```

```
                                05A7          .SBTTL  Check Correctness of the ENQW Test
                                05A7  ;++
                                05A7  ; FUNCTIONAL DESCRIPTION:
                                05A7  ;       Check correctness of enqdeq service routine.
                                05A7  ;
                                05A7  ; CALLING SEQUENCE:
                                05A7  ;       BSBW    CHECK_UP
                                05A7  ;
                                05A7  ; INPUT PARAMETERS:
                                05A7  ;       NONE
                                05A7  ;
                                05A7  ; IMPLICIT INPUTS:
                                05A7  ;       Status code in R0, LKSB
                                05A7  ;       Address of LKSB in R3
                                05A7  ;       Address of $ENQW argument list in R7
                                05A7  ;
                                05A7  ; OUTPUT PARAMETERS:
                                05A7  ;       NONE
                                05A7  ;
                                05A7  ; IMPLICIT OUTPUTS:
                                05A7  ;       Error message if error
                                05A7  ;
                                05A7  ; COMPLETION CODES:
                                05A7  ;       in STATUS if error
                                05A7  ;
                                05A7  ; SIDE EFFECTS:
                                05A7  ;       BSBW    CHK_XXXX for further check
                                05A7  ;--
                                05A7
                                05A7  CHECK_UP:
   03 10 A7     03      E1      05A7          BBC     #LCK$V_SYNCSTS,ENQ$_FLAGS(R7),10$ ; BR if not SYNCSTS test
               00D6     31      05AC          BRW     CHK_SYNCSTS                     ; Check SYNCSTS flag test
                                05AF  10$:
   03 10 A7     02      E1      05AF          BBC     #LCK$V_NOQUEUE,ENQ$_FLAGS(R7),20$ ; BR if NOQUEUE not specfied
               0173     31      05B4          BRW     CHK_NOQUEUE                     ; Check NOQUEUE flag test
                                05B7  20$:
 03 008B'CF     07      E1      05B7          BBC     #VICTIM_V,TSTFLG,30$            ; BR if not dlock test w/ known victim
               01A3     31      05BD          BRW     CHK_DEADLOCK                    ; Check for deadlock - return via subr
                                05C0  30$:
               01D9     30      05C0          BSBW    CHK_SS                          ; Check R0 = #SS$_NORMAL
               01F3     30      05C3          BSBW    CHK_LKSB                        ; Check LKSB status code = #SS$_NORMAL
                                05C6
 03 008B'CF     05      E0      05C6          BBS     #NOCAST_V,TSTFLG,40$            ; BR if no_completion_ast specified
               0012     30      05CC          BSBW    CHK_CAST                        ; Go check completion ast delivered
                                05CF  40$:
 03 008B'CF     01      E1      05CF          BBC     #INCOMP_V,TSTFLG,50$            ; Incompatible flag set?
               0017     30      05D5          BSBW    CHK_BLOCKAST                    ; Go check blocking ast delivered
                                05D8  50$:
   03 10 A7     00      E1      05D8          BBC     #LCK$V_VALBLK,ENQ$_FLAGS(R7),60$ ; BR if not value block test
               002A     30      05DD          BSBW    CHK_VALBLK                      ; Go check value block test
                                05E0  60$:
                05      05E0                  RSB
```

```
                        05E1          .SBTTL  Check Completion AST
                        05E1     ;++
                        05E1     ; FUNCTIONAL DESCRIPTION:
                        05E1     ;        Check completion ast delivered routine.
                        05E1     ;
                        05E1     ; CALLING SEQUENCE:
                        05E1     ;        BSBW    CHK_CAST
                        05E1     ;
                        05E1     ; INPUT PARAMETERS:
                        05E1     ;        NONE
                        05E1     ;
                        05E1     ; IMPLICIT INPUTS:
                        05E1     ;        NONE
                        05E1     ;
                        05E1     ; OUTPUT PARAMETERS:
                        05E1     ;        NONE
                        05E1     ;
                        05E1     ; IMPLICIT OUTPUTS:
                        05E1     ;        Exit message if error
                        05E1     ;
                        05E1     ; COMPLETION CODES:
                        05E1     ;        in STATUS if error
                        05E1     ;
                        05E1     ; SIDE EFFECTS:
                        05E1     ;        Exit if error
                        05E1     ;--
                        05E1
                        05E1     CHK_CAST:
07 01E0'CF    00   E4   05E1              BBSC    #C_AST_V,LOCFLG,10$     ; BR if comp_ast bit set (AST deliv.)
       03AC'CF     DF   05E7              PUSHAL  CAST_ERRMSG             ; Error message
          FCFF     31   05EB              BRW     FAIL_OUT                ; Failure exit
                        05EE     10$:
                   05   05EE              RSB                             ; Return
```

```
        05EF          .SBTTL  Check Blocking AST
        05EF    ;++
        05EF    ; FUNCTIONAL DESCRIPTION:
        05EF    ;       Check blocking ast delivered routine.  The delivery of blocking AST
        05EF    ;       to another is indicated by setting a common EF BAST_CEF.
        05EF    ;
        05EF    ; CALLING SEQUENCE:
        05EF    ;       BSBW    CHK_BLOCKAST
        05EF    ;
        05EF    ; INPUT PARAMETERS:
        05EF    ;       NONE
        05EF    ;
        05EF    ; IMPLICIT INPUTS:
        05EF    ;       NONE
        05EF    ;
        05EF    ; OUTPUT PARAMETERS:
        05EF    ;       NONE
        05EF    ;
        05EF    ; IMPLICIT OUTPUTS:
        05EF    ;       Error message if error
        05EF    ;
        05EF    ; COMPLETION CODES:
        05EF    ;       In STATUS if error
        05EF    ;
        05EF    ; SIDE EFFECTS:
        05EF    ;       NONE
        05EF    ;--
        05EF
        05EF    CHK_BLOCKAST:
        05EF            $WAITFR_S EFN = #BAST_CEF       ; Wait for blocking ast delivered
        05FC            $CLREF_S EFN = #BAST_CEF        ; Reset the blocking ast CEF flag
   05   0609            RSB                             ; Return
```

```
                          060A          .SBTTL  Check Lock Value Block
                          060A  ;++
                          060A  ; FUNCTIONAL DESCRIPTION:
                          060A  ;       Check value block test routine.
                          060A  ;
                          060A  ; CALLING SEQUENCE:
                          060A  ;       BSBW    CHK_VALBLK
                          060A  ;
                          060A  ; INPUT PARAMETERS:
                          060A  ;       NONE
                          060A  ;
                          060A  ; IMPLICIT INPUTS:
                          060A  ;       Address of $ENQ argument list in R7
                          060A  ;
                          060A  ; OUTPUT PARAMETERS:
                          060A  ;       NONE
                          060A  ;
                          060A  ; IMPLICIT OUTPUTS:
                          060A  ;       Error message if error
                          060A  ;
                          060A  ; COMPLETION CODES:
                          060A  ;       In STATUS if error
                          060A  ;
                          060A  ; SIDE EFFECTS:
                          060A  ;       NONE
                          060A  ;--
                          060A
                          060A  CHK_VALBLK:
       58  01F1'CF    D0  060A          MOVL    LKSB_ADDR,R8                    ; Addr of LKSB
    07 10 A7    01    E0  060F          BBS     #LCK$V_CONVERT,ENQ$_FLAGS(R7),10$ ; BR if not a new lock
  01E5'CF    08 A8    7D  0614          MOVQ    8(R8),SAVED_VAL                 ; Save the low quadword of lock value bk
                      05  061A          RSB                                     ; Return
                          061B  10$:                                            ; Not new lock
  01 008B'CF    02    E0  061B          BBS     #VALBLK_V,TSTFLG,20$            ; BR if VALBLK test flag set
                      05  0621          RSB                                     ; Return
                          0622  20$:
                          0622          $READEF_S EFN = #CMP_VAL,-              ; Read the CEF flag
                          0622                  STATE = EF_STATE
  11 01E1'CF    02    E0  0633          BBS     #CMP_VAL_V,EF_STATE,30$         ; BR if CMP_VAL flag set
                          0639          $SETEF_S EFN = #CMP_VAL                 ; Set the common EF CMP_VAL
             08 A8    D6  0646          INCL    8(R8)                           ; Increment the user lock value block
                      05  0649          RSB                                     ; Return
                          064A  30$:
           01E5'CF    D6  064A          INCL    SAVED_VAL                       ; Increment the saved value
  01E5'CF    08 A8    D1  064E          CMPL    8(R8),SAVED_VAL                 ; Is the lock value OK?
                0E    12  0654          BNEQ    40$                             ; Error if not equal
                          0656          $CLREF_S EFN = #CMP_VAL                 ; Reset the CEF
                      05  0663          RSB                                     ; Return
                          0664  40$:
                          0664          $FAO_S  CTRSTR = LKVAL_ERRMSG,-         ; Value block error message
                          0664                  OUTLEN = BUFFER_PTR,-
                          0664                  OUTBUF = FAO_BUF,-
                          0664                  P1     = 08(R8),-
                          0664                  P2     = SAVED_VAL
           0481'CF    DF  067E          PUSHAL  BUFFER_PTR
              FC68    31  0682          BRW     FAIL_OUT
                          0685
```

```
                          0685          .SBTTL  Check SYNCSTS Flag Routine
                          0685  ;++
                          0685  ; FUNCTIONAL DESCRIPTION:
                          0685  ;       Check LCK$M_SYNCSTS flag test routine
                          0685  ;
                          0685  ; CALLING SEQUENCE:
                          0685  ;       BSBW    CHK_SYNCSTS
                          0685  ;
                          0685  ; INPUT PARAMETERS:
                          0685  ;       NONE
                          0685  ;
                          0685  ; IMPLICIT INPUTS:
                          0685  ;       R0 is the returned status from $ENQW
                          0685  ;       LOCFLG synchronizes us with other test routines
                          0685  ;       Test flag in TSTFLG
                          0685  ;
                          0685  ; OUTPUT PARAMETERS:
                          0685  ;       NONE
                          0685  ;
                          0685  ; IMPLICIT OUTPUTS:
                          0685  ;       Error message if error
                          0685  ;
                          0685  ; COMPLETION CODES:
                          0685  ;       In STATUS if error
                          0685  ;
                          0685  ; SIDE EFFECTS:
                          0685  ;       NONE
                          0685  ;--
                          0685  CHK_SYNCSTS:
 56 008B'CF    03    E1   0685          BBC     #SYNCST_V,TSTFLG,30$     ; BR if SYNCH flag cleared
    0000'8F    50    B1   068B          CMPW    R0,#SS$_SYNCH            ; Compare R0 status code
               28    13   0690          BEQL    10$                      ; BR if equal
               50    DD   0692          PUSHL   R0                       ; Save the status...
                          0694          $FAO_S  CTRSTR = NOSYNCH_ERRMSG,- ; ...and give a useful error
                          0694                  OUTLEN = BUFFER_PTR,-
                          0694                  OUTBUF = FAO_BUF,-
                          0694                  P1     = R0
          0481'CF    DF   06A9          PUSHAL  BUFFER_PTR
               01    DD   06AD          PUSHL   #1
      00741132 8F    DD   06AF          PUSHL   #UETP$_TEXT!STS$K_ERROR
               04    DD   06B5          PUSHL   #4
             0548    31   06B7          BRW     ERROR_EXIT
                          06BA  10$:
 07 01E0'CF    00    E1   06BA          BBC     #C_AST_V,LOCFLG,20$      ; Should no comp ast delivered
          0465'CF    DF   06C0          PUSHAL  CASTSYNCH_ERRMSG         ; Error message
             FC26    31   06C4          BRW     FAIL_OUT                 ; Failure exit
                          06C7  20$:
                          06C7          $READEF_S EFN = #SYNC_EF,-       ; Read EF
                          06C7                  STATE = EF_STATE
 4F 01E1'CF    01    E1   06D4          BBC     #SYNC_EF,EF_STATE,50$    ; Error if SYNC EFN set
          04D3'CF    DF   06DA          PUSHAL  SYNCH_ERRMSG             ; Error message
             FC0C    31   06DE          BRW     FAIL_OUT                 ; Failure out
                          06E1  30$:
    0000'8F    50    B1   06E1          CMPW    R0,#SS$_NORMAL           ; Check R0
               28    13   06E6          BEQL    40$                      ; BR if equal
               50    DD   06E8          PUSHL   R0                       ; Save the status...
                          06EA          $FAO_S  CTRSTR = NOSYQUEUE_ERRMSG,- ; ...and give a useful error
```

```
                    06EA  1587              OUTLEN = BUFFER_PTR,-
                    06EA  1588              OUTBUF = FAO_BUF,-
                    06EA  1589              P1     = R0
       0481'CF  DF  06FF  1590      PUSHAL  BUFFER_PTR
            01  DD  0703  1591      PUSHL   #1
   00741132 8F  DD  0705  1592      PUSHL   #UETP$_TEXT!STS$K_ERROR
            04  DD  070B  1593      PUSHL   #4
          04F2  31  070D  1594      BRW     ERROR_EXIT
                    0710  1595 40$:
                    0710  1596      $SETEF_S EFN = #UNLOCK_CEF      ; Set unlock event flag
                    071D  1597      $WAITFR_S EFN = #SYNC_EF        ; Wait for SYNC_EF
          FEB8  30  0726  1598      BSBW    CHK_CAST                ; Comp ast should be delivered
                    0729  1599 50$:
                05  0729  1600      RSB                             ; Return
```

```
                          072A         .SBTTL  Check NOQUEUE Flag Routine
                          072A  ;++
                          072A  ; FUNCTIONAL DESCRIPTION:
                          072A  ;       Check LCK$M_NOQUEUE flag function test routine
                          072A  ;
                          072A  ; CALLING SEQUENCE:
                          072A  ;       BSBW    CHK_NOQUEUE
                          072A  ;
                          072A  ; INPUT PARAMETERS:
                          072A  ;       NONE
                          072A  ;
                          072A  ; IMPLICIT INPUTS:
                          072A  ;       Status code in R0
                          072A  ;
                          072A  ; OUTPUT PARAMETERS:
                          072A  ;       NONE
                          072A  ;
                          072A  ; IMPLICIT OUTPUTS:
                          072A  ;       Error message if error
                          072A  ;
                          072A  ; COMPLETION CODES:
                          072A  ;       STATUS if error
                          072A  ;
                          072A  ; SIDE EFFECTS:
                          072A  ;       NONE
                          072A  ;
                          072A  ;--
                          072A
                          072A  CHK_NOQUEUE:
   0000'8F     50   B1    072A          CMPW    R0,#SS$_NOTQUEUED       ; Is R0 status code OK?
               01   12    072F          BNEQ    10$                     ; BR if not
                    05    0731          RSB                             ; Return
                          0732  10$:
   0511'CF     50   D0    0732          MOVL    R0,STATUS               ; Save returned status as exit status
                          0737          $FAO_S  CTRSTR = PAR_ERRMSG,-   ; Wrong status returned
                          0737                  OUTLEN = BUFFER_PTR,-
                          0737                  OUTBUF = FAO_BUF,-
                          0737                  P1     = STATUS
        0511'CF     DD    074E          PUSHL   STATUS
        0481'CF     DF    0752          PUSHAL  BUFFER_PTR
             01     DD    0756          PUSHL   #1
    00741132 8F     DD    0758          PUSHL   #UETP$_TEXT!STS$K_ERROR
             04     DD    075E          PUSHL   #4
           049F     31    0760          BRW     ERROR_EXIT
```

```
                        0763          .SBTTL  Check Deadlock Test
                        0763    ;++
                        0763    ; FUNCTIONAL DESCRIPTION:
                        0763    ;       Check deadlock test routine.  This routine is executed only in those
                        0763    ;       situations where a victim can be preselected and the results of
                        0763    ;       requesting a lock predetermined.
                        0763    ;
                        0763    ; CALLING SEQUENCE:
                        0763    ;       BSBW    CHK_DEADLOCK
                        0763    ;
                        0763    ; INPUT PARAMETERS:
                        0763    ;       Status code in R0
                        0763    ;
                        0763    ; IMPLICIT INPUTS:
                        0763    ;       R3 points to lock status block
                        0763    ;
                        0763    ; OUTPUT PARAMETERS:
                        0763    ;       NONE
                        0763    ;
                        0763    ; IMPLICIT OUTPUTS:
                        0763    ;       Error message if expected status not found
                        0763    ;
                        0763    ; COMPLETION CODES:
                        0763    ;       NONE
                        0763    ;
                        0763    ; SIDE EFFECTS:
                        0763    ;       Program exits if expected status not found
                        0763    ;--
                        0763
                        0763    CHK_DEADLOCK:
 0000'8F     63   B1    0763            CMPW    (R3),#SS$_DEADLOCK      ; Is the status code OK?
             01   12    0768            BNEQ    10$                     ; BR if not
                  05    076A            RSB
                        076B    10$:
 0511'CF     63   3C    076B            MOVZWL  (R3),STATUS             ; Save returned status as exit status
                        0770            $FAO_S  CTRSTR = VICTIM_ERRMSG,- ; Wrong status returned
                        0770                    OUTLEN = BUFFER_PTR,-
                        0770                    OUTBUF = FAO_BUF,-
                        0770                    P1     = STATUS
     0511'CF      DD    0787            PUSHL   STATUS
     0481'CF      DF    078B            PUSHAL  BUFFER_PTR
          01      DD    078F            PUSHL   #1
 00741132 8F      DD    0791            PUSHL   #UETP$_TEXT!STS$K_ERROR
          04      DD    0797            PUSHL   #4
        0466      31    0799            BRW     ERROR_EXIT
```

```
                    079C          .SBTTL  Check Status Code Subroutine
                    079C  ;++
                    079C  ; FUNCTIONAL DESCRIPTION:
                    079C  ;       Subroutine to check normal status code in R0 (CHK_SS)
                    079C  ;       and LKSB (CHK_LKSB), exit if error is found.
                    079C  ;
                    079C  ; CALLING SEQUENCE:
                    079C  ;       BSBW    CHK_SS
                    079C  ;       BSBW    CHK_LKSB
                    079C  ;
                    079C  ; INPUT PARAMETERS:
                    079C  ;       NONE
                    079C  ;
                    079C  ; IMPLICIT INPUTS:
                    079C  ;       Status code in R0 (CHK_SS), in LKSB (CHK_LKSB)
                    079C  ;       Address of LKSB in R3 (CHK_LKSB)
                    079C  ;
                    079C  ; OUTPUT PARAMETERS:
                    079C  ;       NONE
                    079C  ;
                    079C  ; IMPLICIT OUTPUTS:
                    079C  ;       Exit if error
                    079C  ;
                    079C  ; COMPLETION CODES:
                    079C  ;       In STATUS if error
                    079C  ;
                    079C  ; SIDE EFFECTS:
                    079C  ;       Exit if error
                    079C  ;--
                    079C
                    079C  CHK_SS:
0000'8F    50  B1   079C          CMPW    R0,#SS$_NORMAL          ; Success ?
           01  12   07A1          BNEQ    10$                     ; BR if not
           05       07A3          RSB                             ; Return
                    07A4  10$:
0511'CF    50  D0   07A4          MOVL    R0,STATUS               ; Status code in STATUS
           02  F0   07A9          INSV    #STS$K_ERROR,-          ; We'll force severity to be...
           00       07AB                  #STS$V_SEVERITY,-       ; ...an error, always
0511'CF    03       07AC                  #STS$S_SEVERITY,STATUS
     0511'CF   DD   07B0          PUSHL   STATUS
          01   DD   07B4          PUSHL   #1
        0449   31   07B6          BRW     ERROR_EXIT
                    07B9
                    07B9
                    07B9  CHK_LKSB:
0000'8F    63  B1   07B9          CMPW    (R3),#SS$_NORMAL        ; Status code in LKSB correct?
           01  12   07BE          BNEQ    10$                     ; BR if not
           05       07C0          RSB
                    07C1  10$:
0511'CF    63  3C   07C1          MOVZWL  (R3),STATUS             ; Error status code
           02  F0   07C6          INSV    #STS$K_ERROR,-          ; We'll force severity to be...
           00       07C8                  #STS$V_SEVERITY,-       ; ...an error, always
0511'CF    03       07C9                  #STS$S_SEVERITY,STATUS
     0511'CF   DD   07CD          PUSHL   STATUS
          01   DD   07D1          PUSHL   #1
        042C   31   07D3          BRW     ERROR_EXIT
```

```
                          07D6        .SBTTL  Completion AST Routine
                          07D6  ;++
                          07D6  ; FUNCTIONAL DESCRIPTION:
                          07D6  ;       ENQ(W) completion ast routine. This routine set a flag in LOCFLG
                          07D6  ;       to indicate the completion ast being delivered and check the ast
                          07D6  ;       parameter.
                          07D6  ;
                          07D6  ; CALLING SEQUENCE:
                          07D6  ;       Called via AST when system service ENQ(W) complete
                          07D6  ;
                          07D6  ; INPUT PARAMETERS:
                          07D6  ;       AST parameter = LKID_ADDR
                          07D6  ;
                          07D6  ; IMPLICIT INPUTS:
                          07D6  ;       NONE
                          07D6  ;
                          07D6  ; OUTPUT PARAMETERS:
                          07D6  ;       NONE
                          07D6  ;
                          07D6  ; IMPLICIT OUTPUTS:
                          07D6  ;       Error message if error
                          07D6  ;
                          07D6  ; COMPLETION CODES:
                          07D6  ;       In STATUS if error
                          07D6  ;
                          07D6  ; SIDE EFFECTS:
                          07D6  ;       NONE
                          07D6  ;--
                          07D6
                          07D6  COMP_AST:
                   0FFC   07D6        .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                          07D8
      01E0'CF   01   88   07D8        BISB2   #C_AST_M,LOCFLG          ; Set completion ast deliv flag
   01ED'CF   04 AC   D1   07DD        CMPL    4(AP),LKID_ADDR          ; AST parameter OK ?
                01   12   07E3        BNEQ    10$                      ; Error if not equal
                     04   07E5        RET
                          07E6  10$:
                          07E6        $FAO_S  CTRSTR = CASTPAR_ERRMSG,- ; AST parameter error message
                          07E6                OUTLEN = BUFFER_PTR,-
                          07E6                OUTBUF = FAO_BUF,-
                          07E6                P1     = 04(AP),-
                          07E6                P2     = LKID_ADDR
           0481'CF   DF   0800        PUSHAL  BUFFER_PTR
              FAE6   31   0804        BRW     FAIL_OUT                 ; Failure exit
```

```
                          0807        .SBTTL  Blocking AST Routine
                          0807  ;++
                          0807  ; FUNCTIONAL DESCRIPTION:
                          0807  ;       This blocking ast routine is called whenever the lock granted
                          0807  ;       blocks another lock request. The routine set a common EF for another
                          0807  ;       process to check the ast delivered. It then dequeue the lock such
                          0807  ;       that another process can get the requested lock. In SYNCSTS flag test,
                          0807  ;       The routine waits for a common EF before dequeue the lock.
                          0807  ;
                          0807  ; CALLING SEQUENCE:
                          0807  ;       Called via blocking AST
                          0807  ;
                          0807  ; INPUT PARAMETERS:
                          0807  ;       LKID_ADDR as AST parameter
                          0807  ;
                          0807  ; IMPLICIT INPUTS:
                          0807  ;       NONE
                          0807  ;
                          0807  ; OUTPUT PARAMETERS:
                          0807  ;       NONE
                          0807  ;
                          0807  ; IMPLICIT OUTPUTS:
                          0807  ;       Error message if error found
                          0807  ;
                          0807  ; COMPLETION CODES:
                          0807  ;       In STATUS if error
                          0807  ;
                          0807  ; SIDE EFFECTS:
                          0807  ;       NONE
                          0807  ;--
                          0807
                          0807  BLOCK_AST:
                   0FFC   0807        .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                          0809
                          0809        $SETEF_S EFN = #BAST_CEF         ; Set a CEF BAST_CEF
  0D 008B'CF    08   E1   0816        BBC     #SYNCST_M,TSTFLG,10$     ; BR if SYNCST flag clear
                          081C        $WAITFR_S EFN = #UNLOCK_CEF      ; Wait for the unlock flag
                          0829  10$:
     58   01ED'CF    D0   0829        MOVL    LKID_ADDR,R8             ; Address of LKID
                          082E        $DEQ_S  LKID = (R8)              ; Dequeue the lock
           FF5E      30   083B        BSBW    CHK_SS                   ; Check R0
                     04   083E        RET
```

```
                              083F            .SBTTL  Deadlock Detection and Resolution Routine
                              083F    ;++
                              083F    ; FUNCTIONAL DESCRIPTION:
                              083F    ;       This routine is specified as the completion AST routine when detecting
                              083F    ;       and resolving a deadlock situation between processes each of which is
                              083F    ;       equally likely to be the deadlock victim.
                              083F    ;
                              083F    ; CALLING SEQUENCE:
                              083F    ;       Called via AST if deadlock is detected or resolved
                              083F    ;
                              083F    ; INPUT PARAMETERS:
                              083F    ;       04(AP) is LKID_ADDR
                              083F    ;
                              083F    ; IMPLICIT INPUTS:
                              083F    ;       Associated lock status block
                              083F    ;
                              083F    ; OUTPUT PARAMETERS:
                              083F    ;       NONE
                              083F    ;
                              083F    ; IMPLICIT OUTPUTS:
                              083F    ;       Error message if error found
                              083F    ;
                              083F    ; COMPLETION CODES:
                              083F    ;       NONE
                              083F    ;
                              083F    ; SIDE EFFECTS:
                              083F    ;       Program exits on some errors.  Common event flags signifying deadlock
                              083F    ;       detection or resolution may be set.  The victim process dequeues its
                              083F    ;       request for a deadlocked resource.  Timers to prevent permanent
                              083F    ;       deadlock are cancelled.
                              083F    ;--
                              083F
                              083F    DLOCK_AST:
                        0FFC  083F            .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                              0841
   53   04 AC     04    C3    0841            SUBL3   #4,04(AP),R3            ; Figure the LKSB address
      63   0000'8F      B1    0846            CMPW    #SS$_DEADLOCK,(R3)      ; Are we the victim process?
                  32    12    084B            BNEQ    10$                     ; BR if not
                              084D            $SETEF_S EFN = #DLDET_CEF       ; Indicate that a victim has been found
                              085A            $DEQ_S  FLAGS = #LCK$M_DEQALL   ; Dequeue all my locks, allowing...
                              0869                                            ; ...other process(es) to succeed
   50   00000000'8F     D1    0869            CMPL    #SS$_NORMAL,R0          ; Did we dequeue all locks?
                  60    12    0870            BNEQ    30$                     ; BR if not
                              0872            $CANTIM_S REQIDT = 04(AP)       ; Dlock was found, we don't need timer
                        04    087E            RET                             ; We're done with deadlock detection
```

```
                           087F      10$:
            FF37      30   087F              BSBW    CHK_LKSB                     ; See if we got our lock, broken dlock
                           0882              $READEF_S EFN = #DLDET_CEF,-          ; Did we get it because...
                           0882                      STATE = EF_STATE
    07 01E1'CF    04  E0   0893              BBS     #DLDET_CEF_V,EF_STATE,20$ ; ...some other process got deadlock?
         06C8'CF      DF   0899              PUSHAL  NODLOCK_ERRMSG
            FA4D      31   089D              BRW     FAIL_OUT
                           08A0      20$:
                           08A0              $SETEF_S EFN = #DLRES_CEF        ; Indicate that deadlock has been resolved
                           08AD              $DEQ_S  FLAGS = #LCK$M_DEQALL    ; Dequeue all my locks, allowing...
                           08BC                                               ; ...other tests to run
    50   00000000'8F  D1   08BC              CMPL    #SS$_NORMAL,R0           ; Did we dequeue all locks?
              0D      12   08C3              BNEQ    30$                      ; BR if not
                           08C5              $CANTIM_S REQIDT = 04(AP)        ; Dlock was resolved, we don't need timer
                      04   08D1              RET
                           08D2
                           08D2      30$:
       0511'CF    50  D0   08D2              MOVL    R0,STATUS                ; Save returned status as exit status
                           08D7              $FAO_S  CTRSTR = DEQALL_ERRMSG,- ; Wrong status returned
                           08D7                      OUTLEN = BUFFER_PTR,-
                           08D7                      OUTBUF = FAO_BUF,-
                           08D7                      P1     = STATUS
         0511'CF      DD   08EE              PUSHL   STATUS
         0481'CF      DF   08F2              PUSHAL  BUFFER_PTR
              01      DD   08F6              PUSHL   #1
     00741132 8F      DD   08F8              PUSHL   #UETP$_TEXT!STS$K_ERROR
              04      DD   08FE              PUSHL   #4
            02FF      31   0900              BRW     ERROR_EXIT
```

```
                        0903          .SBTTL  Deadlock Timeout AST Routine
                        0903  ;++
                        0903  ; FUNCTIONAL DESCRIPTION:
                        0903  ;       This routine executes only if deadlock was not detected or resolved to
                        0903  ;       this process's satisfaction in some reasonable time, a multiple of the
                        0903  ;       SYSGEN parameter of the interval for deadlock detection.
                        0903  ;
                        0903  ; CALLING SEQUENCE:
                        0903  ;       Called via $SETIMR AST
                        0903  ;
                        0903  ; INPUT PARAMETERS:
                        0903  ;       04(AP) is LKID_ADDR
                        0903  ;
                        0903  ; IMPLICIT INPUTS:
                        0903  ;       NONE
                        0903  ;
                        0903  ; OUTPUT PARAMETERS:
                        0903  ;       NONE
                        0903  ;
                        0903  ; IMPLICIT OUTPUTS:
                        0903  ;       Error message
                        0903  ;
                        0903  ; COMPLETION CODES:
                        0903  ;       NONE
                        0903  ;
                        0903  ; SIDE EFFECTS:
                        0903  ;       Common event flags specifying deadlock detection and resolution are
                        0903  ;       set.  The resource requested by this process is dequeued.
                        0903  ;--
                        0903
                        0903  DLOCK_TO_AST:
                  0FFC  0903          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0905
              07    C5  0905          MULL3   #PROC_COUNT+5,-           ; Figure how long...
51     00000000'GF      0907                  G^LCK$GL_WAITTIME,R1      ; ...we waited for deadlock
                        090D          $FAO_S  CTRSTR = DEADLK_ERRMSG,-  ; Tell the world that deadlock...
                        090D                  OUTLEN = BUFFER_PTR,-     ; ...detection seemed to fail
                        090D                  OUTBUF = FAO_BUF,-
                        090D                  P1     = R1
         0481'CF    DF  0922          PUSHAL  BUFFER_PTR
            F9C4    31  0926          BRW     FAIL_OUT                  ; We can't continue - fail with msg
```

```
                          0929          .SBTTL  Termination Mailbox AST Routine
                          0929  ;++
                          0929  ; FUNCTIONAL DESCRIPTION:
                          0929  ;       Receives the termination mailboxes from driven processes.
                          0929  ;
                          0929  ; CALLING SEQUENCE:
                          0929  ;       Called via AST
                          0929  ;
                          0929  ; INPUT PARAMETERS:
                          0929  ;       NONE
                          0929  ;
                          0929  ; IMPLICIT INPUTS:
                          0929  ;       MBX_IOSB has the result of the read of a termination mailbox
                          0929  ;       EXIT_MSG is the buffer into which the mail is written
                          0929  ;
                          0929  ; OUTPUT PARAMETERS:
                          0929  ;       NONE
                          0929  ;
                          0929  ; IMPLICIT OUTPUTS:
                          0929  ;       Error message if incorrect or inconsistent info in mailbox
                          0929  ;
                          0929  ; COMPLETION CODES:
                          0929  ;       NONE
                          0929  ;
                          0929  ; SIDE EFFECTS:
                          0929  ;       May cause program termination
                          0929  ;
                          0929  ;--
                          0929
                          0929  EX_MBX_AST:
                    0FFC  0929          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                          092B
     5E 0174'CF     E9    092B          BLBC    MBX_IOSB,50$                ; BR if IO was not successful
  03    017E'CF     B1    0930          CMPW    EXIT_MSG+ACC$W_MSGTYP,#MSG$_DELPROC ; Is it a termination msg?
             73     12    0935          BNEQ    60$                         ; BR if not
  52    0020'CF     DE    0937          MOVAL   PROCIDS,R2                  ; Addr of process ids
                          093C  10$:
  62    0178'CF     D1    093C          CMPL    MBX_IOSB+4,(R2)             ; Process id found?
             0B     13    0941          BEQL    20$                         ; BR if yes
     52    04 A2    DE    0943          MOVAL   4(R2),R2                    ; Next process id
             62     D5    0947          TSTL    (R2)                        ; End of process id table?
             F1     12    0949          BNEQ    10$                         ; Try next if more
           007F     31    094B          BRW     70$                         ; Process id not found
                          094E  20$:
0000'8F 0182'CF     B1    094E          CMPW    EXIT_MSG+ACC$L_FINALSTS,#SS$_NORMAL ; Deleted normally?
             03     13    0955          BEQL    30$
           0095     31    0957          BRW     80$                         ; BR if not
                          095A  30$:
        01D2'CF     D7    095A          DECL    EX_PROC_CNT                 ; Decrement exist proc count
             03     12    095E          BNEQ    40$                         ; BR if there are more to wait for
           00B3     31    0960          BRW     WAKE_UP                     ; Go wake up the controller process
                          0963  40$:
                          0963          $QIO_S  CHAN = EX_MBXCHAN,-         ; QIO read to mailbox for next process
                          0963                  FUNC = #IO$_READVBLK,-
                          0963                  ASTADR = EX_MBX_AST,-
                          0963                  IOSB = MBX_IOSB,-
                          0963                  P1 = EXIT_MSG,-
```

```
                            0963                   P2 = #ACC$K_TERMLEN
                      04    098D            RET                             ; Return
                            098E      50$:
 0511'CF   0174'CF    3C    098E            MOVZWL  MBX_IOSB,STATUS         ; Use mailbox status as exit status
           0511'CF    DD    0995            PUSHL   STATUS
           0760'CF    DF    0999            PUSHAL  NOTRMB_ERRMSG
                01    DD    099D            PUSHL   #1
       00741132 8F    DD    099F            PUSHL   #UETP$_TEXT!STS$K_ERROR
                04    DD    09A5            PUSHL   #4
              0258    31    09A7            BRW     ERROR_EXIT
                            09AA      60$:
      56   017E'CF    3C    09AA            MOVZWL  EXIT_MSG+ACC$W_MSGTYP,R6 ; Complain about the kind of msg we got
                            09AF            $FAO_S  CTRSTR = MSGTYP_ERRMSG,-
                            09AF                    OUTLEN = BUFFER_PTR,-
                            09AF                    OUTBUF = FAO_BUF,-
                            09AF                    P1     = R6,-
                            09AF                    P2     = #MSG$_DELPROC
           0481'CF    DF    09C6            PUSHAL  BUFFER_PTR              ; Error in termination mailbox
              F920    31    09CA            BRW     FAIL_OUT                ; Failure out
                            09CD      70$:
                            09CD            $FAO_S  CTRSTR = TERMBX_ERRMSG,- ; We got a message for the wrong proc
                            09CD                    OUTLEN = BUFFER_PTR,-
                            09CD                    OUTBUF = FAO_BUF,-
                            09CD                    P1     = MBX_IOSB+4,-
                            09CD                    P2     = EXIT_MSG+ACC$L_FINALSTS
           0481'CF    DF    09E8            PUSHAL  BUFFER_PTR
              F8FE    31    09EC            BRW     FAIL_OUT
                            09EF      80$:
  20 01E0'CF    01    E0    09EF            BBS     #DLPRC_V,LOCFLG,90$     ; BR if the process is deleted by LOCK00
                            09F5            $FAO_S  CTRSTR = DETPRC_ERRMSG,- ; Error in detached proccess
                            09F5                    OUTLEN = BUFFER_PTR,-
                            09F5                    OUTBUF = FAO_BUF,-
                            09F5                    P1     = (R2),-
                            09F5                    P2     = EXIT_MSG+ACC$L_FINALSTS
           0481'CF    DF    0A0E            PUSHAL  BUFFER_PTR
              F8D8    31    0A12            BRW     FAIL_OUT                ; Failure out
                            0A15      90$:
                      04    0A15            RET
                            0A16      WAKE_UP:
                            0A16            $WAKE_S                         ; Wake up the controller
                      04    0A21            RET                             ; Return
```

```
                    0A22          .SBTTL  Timer Expiration Routine
                    0A22    ;++
                    0A22    ; FUNCTIONAL DESCRIPTION:
                    0A22    ;         This routine will be called only if the timer which was set to prevent
                    0A22    ;         program hangs goes off.
                    0A22    ;
                    0A22    ; CALLING SEQUENCE:
                    0A22    ;         Called via AST at $SETIMR expiration.
                    0A22    ;
                    0A22    ; INPUT PARAMETERS:
                    0A22    ;         NONE
                    0A22    ;
                    0A22    ; IMPLICIT INPUTS:
                    0A22    ;         NONE
                    0A22    ;
                    0A22    ; OUTPUT PARAMETERS:
                    0A22    ;         NONE
                    0A22    ;
                    0A22    ; IMPLICIT OUTPUTS:
                    0A22    ;         NONE
                    0A22    ;
                    0A22    ; COMPLETION CODES:
                    0A22    ;         #SS$_TIMEOUT
                    0A22    ;
                    0A22    ; SIDE EFFECTS:
                    0A22    ;         NONE
                    0A22    ;
                    0A22    ;--
                    0A22
                    0A22    TIME_OUT:
              0FFC  0A22          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                    0A24
00000000'8F   DD    0A24          PUSHL   #SS$_TIMEOUT              ; Push the signal name
         01   DD    0A2A          PUSHL   #1                        ; Push the argument count total
       01D3   31    0A2C          BRW     ERROR_EXIT                ; Bail out completely
```

```
0A2F        .SBTTL  System Service Exception Handler
0A2F  ;++
0A2F  ; FUNCTIONAL DESCRIPTION:
0A2F  ;         This routine is executed if a software or hardware exception occurs or
0A2F  ;         if a LIB$SIGNAL system service is used to output a message.
0A2F  ;         Information about this method of handling messages and errors can be
0A2F  ;         found in the VMS COMMON RUN-TIME manual and in the VMS SYSTEM SERVICE
0A2F  ;         manual.
0A2F  ;
0A2F  ; CALLING SEQUENCE:
0A2F  ;         Entered via an exception from the system
0A2F  ;
0A2F  ; INPUT PARAMETERS:
0A2F  ;         ERROR_COUNT    = previous cumulative error count
0A2F  ;                          ----------------
0A2F  ;            AP ---->      !       2      !
0A2F  ;                          !--------------!
0A2F  ;                          ! SIGNL ARY PNT!
0A2F  ;                          !--------------!
0A2F  ;                          ! MECH  ARY PNT!
0A2F  ;                          !--------------!  ---------
0A2F  ;                          !      4       !          ^
0A2F  ;                          !--------------!          !
0A2F  ;                          ! ESTABLISH FP !          !
0A2F  ;                          !--------------!          !
0A2F  ;                          !    DEPTH     !  Mechanism Array
0A2F  ;                          !--------------!          !
0A2F  ;                          !      R0      !          !
0A2F  ;                          !--------------!          !
0A2F  ;                          !      R1      !          v
0A2F  ;                          !--------------!  ---------
0A2F  ;                          !      N       !          ^
0A2F  ;                          !--------------!          !
0A2F  ;                          !CONDITION NAME!          !
0A2F  ;                          !--------------!          !
0A2F  ;                          !N-3 ADDITIONAL!    Signal Array
0A2F  ;                          !LONG WORD ARGS!          !
0A2F  ;                          !--------------!          !
0A2F  ;                          !      PC      !          !
0A2F  ;                          !--------------!          !
0A2F  ;                          !     PSL      !          v
0A2F  ;                          !--------------!  ---------
0A2F  ; IMPLICIT INPUTS:
0A2F  ;         NONE
0A2F  ;
0A2F  ; OUTPUT PARAMETERS:
0A2F  ;         NONE
0A2F  ;
0A2F  ; IMPLICIT OUTPUTS:
0A2F  ;         NONE
0A2F  ;
0A2F  ; COMPLETION CODES:
0A2F  ;         NONE
0A2F  ;
0A2F  ; SIDE EFFECTS:
0A2F  ;         May branch to ERROR_EXIT
0A2F  ;--
```

```
                                 0A2F
                                 0A2F   SSERROR:
                          0FFC   0A2F           .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                 0A31
                                 0A31           $SETAST_S ENBFLG = #0               ; Disable AST delivery
                    01     DD    0A3A           PUSHL   #1                          ; Assume ASTs were enabled
      50    00000000'8F    D1    0A3C           CMPL    #SS$_WASSET,R0              ; Were ASTs enabled?
                    02     13    0A43           BEQL    10$                         ; BR if they were
                    6E     D4    0A45           CLRL    (SP)                        ; Set ASTs to remain disabled
                                 0A47   10$:
                                 0A47           $SETSFM_S ENBFLG = #0               ; Disable SS failure mode
                    01     DD    0A50           PUSHL   #1                          ; Assume SS failure mode was enabled
      50    00000000'8F    D1    0A52           CMPL    #SS$_WASSET,R0              ; Was SS failure mode enabled?
                    02     13    0A59           BEQL    20$                         ; BR if it was
                    6E     D4    0A5B           CLRL    (SP)                        ; Set SS failure mode to remain off
                                 0A5D   20$:
          56     04 AC     D0    0A5D           MOVL    CHF$L_SIGARGLST(AP),R6      ; Get the signal array pointer
                    10     ED    0A61           CMPZV   #STS$V_FAC_NO,-             ; Is this a message from LIB$SIGNAL?
                    0C           0A63                   #STS$S_FAC_NO,-
                 04 A6           0A64                   CHF$L_SIG_NAME(R6),-
           00000074 8F           0A66                   #UETP$_FACILITY
                    30     12    0A6B           BNEQ    30$                         ; BR if this is not a UETP exception
          66        02     C2    0A6D           SUBL2   #2,CHF$L_SIG_ARGS(R6)       ; Drop the PC and PSL
                                 0A70           $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6) ; Print the message
                    01     BA    0A7F           POPR    #^M<R0>                     ; Restore SS failure mode...
                                 0A81           $SETSFM_S ENBFLG = R0               ; ...
                    01     BA    0A8A           POPR    #^M<R0>                     ; Restore AST enable...
                                 0A8C           $SETAST_S ENBFLG = R0               ; ...
      50    00000000'8F    D0    0A95           MOVL    #SS$_NORMAL,R0              ; Supply a standard status for exit
                    04           0A9C           RET                                 ; Return to the program
                                 0A9D   30$:
            000000C0'8F    D1    0A9D           CMPL    #SS$_SSFAIL,-               ; RMS fails come through as SS fails
                 04 A6           0AA3                   CHF$L_SIG_NAME(R6)
                    26     12    0AA5           BNEQ    40$                         ; BR if this can't be an RMS failure
                    10     ED    0AA7           CMPZV   #STS$V_FAC_NO,-             ; Is it an RMS failure?
                    0C           0AA9                   #STS$S_FAC_NO,-
                 08 A6           0AAA                   CHF$L_SIG_ARG1(R6),-
                    01           0AAC                   #RMS$_FACILITY
                    1E     12    0AAD           BNEQ    40$                         ; BR if not
                    01     BA    0AAF           POPR    #^M<R0>                     ; Restore SS failure mode...
                                 0AB1           $SETSFM_S ENBFLG = R0               ; ...
                    01     BA    0ABA           POPR    #^M<R0>                     ; Restore AST enable...
                                 0ABC           $SETAST_S ENBFLG = R0               ; ...
      50    00000000'8F    D0    0AC5           MOVL    #SS$_NORMAL,R0              ; Supply a standard status for exit
                    04           0ACC           RET                                 ; RMS_ERROR gets to handle these
                                 0ACD   40$:
     0511'CF     04 A6     D0    0ACD           MOVL    CHF$L_SIG_NAME(R6),STATUS   ; Save the status
                    58     D4    0AD3           CLRL    R8                          ; Assume for now it's not SS failure
0511'CF     00000000'8F    D1    0AD5           CMPL    #SS$_SSFAIL,STATUS          ; But is it a System Service failure?
                    3B     12    0ADE           BNEQ    60$                         ; BR if not - no special case message
                                 0AE0           $GETMSG_S MSGID = CHF$L_SIG_ARG1(R6),- ; Get SS failure code associated text
                                 0AE0                   MSGLEN = BUFFER_PTR,-
                                 0AE0                   BUFADR = FAO_BUF,-
                                 0AE0                   FLAGS  = #14,-
                                 0AE0                   OUTADR = MSG_BLOCK
             0476'CF       95    0AF8           TSTB    MSG_BLOCK+1                 ; Get FAO arg count for SS failure code
                    17     13    0AFC           BEQL    50$                         ; Don't use $GETMSG if no $FAO args...
```

```
           0481'CF   DF  0AFE  2203         PUSHAL  BUFFER_PTR                    ; ...else build up...
                01   DD  0B02  2204         PUSHL   #1                            ; ...a message describing...
       00741130 8F   DD  0B04  2205         PUSHL   #UETP$_TEXT                   ; ...why the System Service failed
             08 A6   F0  0B0A  2206         INSV    CHF$L_SIG_ARG1(R6),-          ; Give the message...
                00       0B0D  2207                 #STS$V_SEVERITY,-             ; ...the correct severity code
             6E 03       0B0E  2208                 #STS$S_SEVERITY,(SP)
             58 03   D0  0B10  2209         MOVL    #3,R8                         ; Count the number of args we pushed
             06      11  0B13  2210         BRB     60$
                         0B15  2211 50$:
             08 A6   DD  0B15  2212         PUSHL   CHF$L_SIG_ARG1(R6)            ; Save SS failure code
             58 01   D0  0B18  2213         MOVL    #1,R8                         ; Count the number of args we pushed
                         0B1B  2214 60$:
       57 66 04      C5  0B1B  2215         MULL3   #4,CHF$L_SIG_ARGS(R6),R7 ; Convert longwords to bytes
             5E 57   C2  0B1F  2216         SUBL2   R7,SP                         ; Save the current signal array...
    6E 04 A6 57      28  0B22  2217         MOVC3   R7,CHF$L_SIG_NAME(R6),(SP) ; ...on the stack
       7E 66 58      C1  0B27  2218         ADDL3   R8,CHF$L_SIG_ARGS(R6),-(SP) ; Push the current arg count
              00D4   31  0B2B  2219         BRW     ERROR_EXIT
```

```
                         0B2E        .SBTTL  RMS Error Handler
                         0B2E   ;++
                         0B2E   ; FUNCTIONAL DESCRIPTION:
                         0B2E   ;       This routine handles error returns from RMS calls.
                         0B2E   ;
                         0B2E   ; CALLING SEQUENCE:
                         0B2E   ;       Called by RMS when a file processing error is found.
                         0B2E   ;
                         0B2E   ; INPUT PARAMETERS:
                         0B2E   ;       The FAB or RAB associated with the RMS call.
                         0B2E   ;
                         0B2E   ; IMPLICIT INPUTS:
                         0B2E   ;       NONE
                         0B2E   ;
                         0B2E   ; OUTPUT PARAMETERS:
                         0B2E   ;       NONE
                         0B2E   ;
                         0B2E   ; IMPLICIT OUTPUTS:
                         0B2E   ;       Error message
                         0B2E   ;
                         0B2E   ; COMPLETION CODES:
                         0B2E   ;       NONE
                         0B2E   ;
                         0B2E   ; SIDE EFFECTS:
                         0B2E   ;       Program may exit, depending on severity of the error.
                         0B2E   ;
                         0B2E   ;--
                         0B2E
                         0B2E   RMS_ERROR:
                  0FFC   0B2E        .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                         0B30
       56    04 AC   D0  0B30        MOVL    4(AP),R6                    ; See whether we're dealing with...
          66    03   91  0B34        CMPB    #FAB$C_BID,FAB$B_BID(R6)  ; ...a FAB or a RAB
                16   12  0B37        BNEQ    10$                         ; BR if it's a RAB
       57  0898'CF   DE  0B39        MOVAL   FILE,R7                     ; FAB-specific code:  text string...
          58    56   D0  0B3E        MOVL    R6,R8                       ; ...address of FAB...
             0C A6   DD  0B41        PUSHL   FAB$L_STV(R6)               ; ...STV field for error...
             08 A6   DD  0B44        PUSHL   FAB$L_STS(R6)               ; ...STS field for error...
  0511'CF    08 A6   D0  0B47        MOVL    FAB$L_STS(R6),STATUS        ; ...and save the error code
                15   11  0B4D        BRB     COMMON                      ; FAB and RAB share other code
                         0B4F   10$:
       57  08A4'CF   DE  0B4F        MOVAL   RECORD,R7                   ; RAB-specific code:  text string...
       58    3C A6   D0  0B54        MOVL    RAB$L_FAB(R6),R8            ; ...address of associated FAB...
             0C A6   DD  0B58        PUSHL   RAB$L_STV(R6)               ; ...STV field for error...
             08 A6   DD  0B5B        PUSHL   RAB$L_STS(R6)               ; ...STS field for error...
  0511'CF    08 A6   D0  0B5E        MOVL    RAB$L_STS(R6),STATUS        ; ...and save the error code
                         0B64   COMMON:
       5A    34 A8   9A  0B64        MOVZBL  FAB$B_FNS(R8),R10           ; Get the file name size
                         0B68        $FAO_S  CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
                         0B68                OUTLEN = BUFFER_PTR,-
                         0B68                OUTBUF = FAO_BUF,-
                         0B68                P1     = R7,-
                         0B68                P2     = R10,-
                         0B68                P3     = FAB$L_FNA(R8)
          0481'CF    DF  0B82        PUSHAL  BUFFER_PTR                  ; ...and arguments for ERROR_EXIT...
                01   DD  0B86        PUSHL   #1                          ; ...
     00741130 8F     DD  0B88        PUSHL   #UETP$_TEXT                 ; ...
```

```
          00    EF  0B8E  2278          EXTZV   #STS$V_SEVERITY,-
          03        0B90  2279                  #STS$S_SEVERITY,-
  59   0511'CF      0B91  2280                  STATUS,R9          ; ...get the severity code...
      6E    59  88  0B95  2281          BISB2   R9,(SP)            ; ...and add it into the signal name
          05    DD  0B98  2282          PUSHL   #5                 ; Current arg count
        0065    31  0B9A  2283          BRW     ERROR_EXIT
```

```
                             0B9D             .SBTTL  CTRL/C Handler
                             0B9D     ;++
                             0B9D     ; FUNCTIONAL DESCRIPTION:
                             0B9D     ;       This routine handles CTRL/C AST's
                             0B9D     ;
                             0B9D     ; CALLING SEQUENCE:
                             0B9D     ;       Called via AST
                             0B9D     ;
                             0B9D     ; INPUT PARAMETERS:
                             0B9D     ;       NONE
                             0B9D     ;
                             0B9D     ; IMPLICIT INPUTS:
                             0B9D     ;       NONE
                             0B9D     ;
                             0B9D     ; OUTPUT PARAMETERS:
                             0B9D     ;       NONE
                             0B9D     ;
                             0B9D     ; IMPLICIT OUTPUTS:
                             0B9D     ;       NONE
                             0B9D     ;
                             0B9D     ; COMPLETION CODES:
                             0B9D     ;       NONE
                             0B9D     ;
                             0B9D     ; SIDE EFFECTS:
                             0B9D     ;       NONE
                             0B9D     ;
                             0B9D     ;--
                             0B9D
                             0B9D     CCASTHAND:
                       0FFC  0B9D             .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                             0B9F
     28 01DE'CF    00   E0   0B9F             BBS     #DRIVEN_V,GLBFLG,30$     ; Skip this if we are a driven proc
        52 02A9'CF      DE   0BA5             MOVAL   PROCS,R2                 ; Addr of process indicator
        53 0020'CF      DE   0BAA             MOVAL   PROCIDS,R3               ; Addr of detached process id
                 55     D4   0BAF             CLRL    R5                       ; Index
                             0BB1     10$:
               6345     D5   0BB1             TSTL    (R3)[R5]                 ; Is the process exited ?
                 13     13   0BB4             BEQL    20$                      ; BR if yes
                             0BB6             $FORCEX_S PIDADR = (R3)[R5],-    ; Delete the process
                             0BB6                     CODE = #0                ; Status is zero
         01E0'CF   02   88   0BC4             BISB2   #DLPRC_M,LOCFLG          ; Set the delete process flag
                             0BC9     20$:
           E4 55   02   F2   0BC9             AOBLSS  #PROC_COUNT,R5,10$       ; Next if more
                             0BCD
                             0BCD     30$:
            02BB'CF     DF   0BCD             PUSHAL  CNTRLCMSG                ; Set message pointer
                 01     DD   0BD1             PUSHL   #1                       ; Set arg count
        00741130 8F     DD   0BD3             PUSHL   #UETP$_TEXT!STS$K_WARNING ; Set signal name
                 00     DD   0BD9             PUSHL   #0                       ; Indicate an abnormal termination
            0008'CF     DF   0BDB             PUSHAL  TEST_NAME_D              ; ...
                 02     DD   0BDF             PUSHL   #2                       ; ...
        007410E0 8F     DD   0BE1             PUSHL   #UETP$_ABENDD!STS$K_WARNING ; ...
     00000000'GF   07   FB   0BE7             CALLS   #7,G^LIB$SIGNAL          ; Output the message...
                        D0   0BEE             MOVL    #<STS$K_SUCCESS!SS$_CONTROLC-- ; ...and exit status
                             0BEF                     STS$K_SUCCESS+STS$K_WARNING>+-
 0511'CF   10000000'8F       0BEF                     STS$M_INHIB_MSG,STATUS
                             0BF7             $EXIT_S STATUS                   ; Terminate program cleanly
```

```
                            0C02        .SBTTL  Error Exit
                            0C02  ;++
                            0C02  ; FUNCTIONAL DESCRIPTION:
                            0C02  ;         This routine prints an error message and exits.
                            0C02  ;
                            0C02  ; CALLING SEQUENCE:
                            0C02  ;         PUSHx error specific information on the stack
                            0C02  ;         PUSHL current argument count
                            0C02  ;         BRW    ERROR_EXIT
                            0C02  ;
                            0C02  ; INPUT PARAMETERS:
                            0C02  ;         Arguments to LIB$SIGNAL, as above
                            0C02  ;
                            0C02  ; IMPLICIT INPUTS:
                            0C02  ;         NONE
                            0C02  ;
                            0C02  ; OUTPUT PARAMETERS:
                            0C02  ;         Message to SYS$OUTPUT (log file) and SYS$ERROR
                            0C02  ;
                            0C02  ; IMPLICIT OUTPUTS:
                            0C02  ;         Program exit
                            0C02  ;
                            0C02  ; COMPLETION CODES:
                            0C02  ;         NONE
                            0C02  ;
                            0C02  ; SIDE EFFECTS:
                            0C02  ;         NONE
                            0C02  ;
                            0C02  ;--
                            0C02
                            0C02  ERROR_EXIT:
 1C 01DE'CF    02     E0    0C02          BBS     #BEGIN_MSGV,GLBFLG,5$    ; BR if already given beginning sentinel
       022C'CF        DF    0C08          PUSHAL  BEGUN_ADDR               ; Not yet, so do it before giving error
       0010'CF        DF    0C0C          PUSHAL  TEST_NAME_C
            02        DD    0C10          PUSHL   #2
     007480D9 8F      DD    0C12          PUSHL   #UETP$_SATSMS!STS$K_SUCCESS
  00000000'GF  04     FB    0C18          CALLS   #4,G^LIB$SIGNAL          ; Squirt out the message...
       01DE'CF 04     A8    0C1F          BISW2   #BEGIN_MSGM,GLBFLG       ; ...and mark it as done
                            0C24  5$:
 40 01DE'CF    00     E0    0C24          BBS     #DRIVEN_V,GLBFLG,30$     ; Skip this if we are a driven proc
 52    02A9'CF        DE    0C2A          MOVAL   PROCS,R2                 ; Addr of process indicator
 53    0020'CF        DE    0C2F          MOVAL   PROCIDS,R3               ; Addr of detached process id
            55        D4    0C34          CLRL    R5                       ; Index
                            0C36  10$:
          6345        D5    0C36          TSTL    (R3)[R5]                 ; Is the process exited ?
            13        13    0C39          BEQL    20$                      ; BR if yes
                            0C3B          $FORCEX_S PIDADR = (R3)[R5],-    ; Delete the process
                            0C3B                  CODE = #0                ; Status is zero
     01E0'CF   02     88    0C49          BISB2   #DLPRC_M,LOCFLG          ; Set the delete process flag
                            0C4E  20$:
     E4 55     02     F2    0C4E          AOBLSS  #PROC_COUNT,R5,10$       ; Next if more
                            0C52          $SCHDWK_S DAYTIM = TEN_SECONDS   ; Cheap way to allow process rundown...
                            0C63          $HIBER_S                         ; ...for the ones we just zapped
                            0C6A
                            0C6A  30$:
       0511'CF        D5    0C6A          TSTL    STATUS                   ; Was any exit status supplied?
            09        12    0C6E          BNEQ    40$                      ; BR if one was
```

```
        007410E2 8F   D0  0C70  2400       MOVL    #UETP$_ABENDD!STS$K_ERROR,- ; Supply a generic one otherwise
            0511'CF       0C76  2401               STATUS
                          0C79  2402 40$:
  050D'CF     04   8E  C1 0C79  2403       ADDL3   (SP)+,#4,ARG_COUNT         ; Get total # args, pop partial count
                   00  DD 0C7F  2404       PUSHL   #0                         ; Push the time parameter
            0008'CF    DF 0C81  2405       PUSHAL  TEST_NAME_D                ; Push test name...
                   02  DD 0C85  2406       PUSHL   #2                         ; ...arg count...
        007410E2 8F    DD 0C87  2407       PUSHL   #UETP$_ABENDD!STS$K_ERROR  ; ...and signal name
00000000'GF   050D'CF  FB 0C8D  2408       CALLS   ARG_COUNT,G^LIB$SIGNAL     ; Print the message
0511'CF   10000000 8F  C8 0C96  2409       BISL    #STS$M_INHIB_MSG,STATUS    ; Don't print messages twice
                          0C9F  2410       $EXIT_S STATUS                     ; Exit in error
```

```
                          0CAA         .SBTTL  Exit Handler
                          0CAA   ;++
                          0CAA   ; FUNCTIONAL DESCRIPTION:
                          0CAA   ;       Output the log file of detached processes to SYS$OUTPUT and exit.
                          0CAA   ;
                          0CAA   ; CALLING SEQUENCE:
                          0CAA   ;       Invoked automatically by $EXIT system service.
                          0CAA   ;
                          0CAA   ; INPUT PARAMETERS:
                          0CAA   ;       NONE
                          0CAA   ;
                          0CAA   ; IMPLICIT INPUTS:
                          0CAA   ;       STATUS contains the exit status
                          0CAA   ;       Log files of cooperating detached processes
                          0CAA   ;
                          0CAA   ; OUTPUT PARAMETERS:
                          0CAA   ;       NONE
                          0CAA   ;
                          0CAA   ; IMPLICIT OUTPUTS:
                          0CAA   ;       Contents of cooperating processes written to SYS$OUTPUT
                          0CAA   ;
                          0CAA   ; COMPLETION CODES:
                          0CAA   ;       NONE
                          0CAA   ;
                          0CAA   ; SIDE EFFECTS:
                          0CAA   ;       Log files of cooperating detached processes are deleted
                          0CAA   ;
                          0CAA   ;--
                          0CAA   EXIT_HANDLER:
                    0FFC  0CAA         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                          0CAC
                          0CAC         $SETSFM_S ENBFLG = #0           ; We can't trap System Service errors
                          0CB5         $SETAST_S ENBFLG = #0           ; ASTs could screw us up now
                          0CBE
     17 01DE'CF   02  E0  0CBE         BBS     #BEGIN_MSGV,GLBFLG,5$   ; BR if already given beginning sentinel
        022C'CF       DF  0CC4         PUSHAL  BEGUN_ADDR              ; Not yet, so do it before giving error
        0010'CF       DF  0CC8         PUSHAL  TEST_NAME_C
             02       DD  0CCC         PUSHL   #2
    007480D9 8F       DD  0CCE         PUSHL   #UETP$_SATSMS!STS$K_SUCCESS
 00000000'GF      04  FB  0CD4         CALLS   #4,G^LIB$SIGNAL         ; Squirt out the message...
                          0CDB         BISW2   #BEGIN_MSGM,GLBFLG      ; ...and mark it as done
                          0CDB   5$:
                          0CDB
     03 01DE'CF   00  E0  0CDB         BBS     #DRIVEN_V,GLBFLG,10$    ; Ignore log files if we're driven proc
           0071       30  0CE1         BSBW    TYPE_LOG_FILES          ; Dump driven process log file(s)
                          0CE4
                          0CE4   10$:
        0511'CF       D5  0CE4         TSTL    STATUS                  ; The only way we get no status...
             35       12  0CE8         BNEQ    EXIT_MESSAGE            ; ...is if we were $FORCEXed. BR if OK
                          0CEA         $FAO_S  CTRSTR = FORCEX_MSG,-   ; Type a message to tell of our...
                          0CEA                 OUTLEN = BUFFER_PTR,-   ; ...$FORCEX plight
                          0CEA                 OUTBUF = FAO_BUF,-
                          0CEA                 P1     = #TEST_NAME_D
        0481'CF       DF  0D03         PUSHAL  BUFFER_PTR
             01       DD  0D07         PUSHL   #1
    00741132 8F       DD  0D09         PUSHL   #UETP$_TEXT!STS$K_ERROR
 00000000'GF      03  FB  0D0F         CALLS   #3,G^LIB$SIGNAL
```

```
 0511'CF   107410E2 8F   D0  0D16        MOVL    #UETP$_ABENDD!STS$K_ERROR!- ; Supply a default exit status
                             0D1F                STS$M_INHIB_MSG,STATUS
                             0D1F
                             0D1F   EXIT_MESSAGE:
              0232'CF    DF  0D1F        PUSHAL  END_ADDR                  ; Assume we completed successfully
           05 0511'CF    E8  0D23        BLBS    STATUS,10$                ; But did we?
         6E   023D'CF    DE  0D28        MOVAL   FAIL_ADDR,(SP)            ; Modify message if we failed
                             0D2D   10$:
              0010'CF    DF  0D2D        PUSHAL  TEST_NAME_C
                   02    DD  0D31        PUSHL   #2                        ; Argument count
           007480D9 8F   DD  0D33        PUSHL   #UETP$_SATSMS             ; Set the end message code
         00   0511'CF    F0  0D39        INSV    STATUS,#STS$V_SEVERITY,-  ; Give the test status
              6E   03        0D3E                #STS$S_SEVERITY,(SP)
                   04    DD  0D40        PUSHL   #4
              51   5E    D0  0D42        MOVL    SP,R1
                             0D45        $PUTMSG_S MSGVEC = (R1)           ; Print the ending message
                         04  0D54        RET
```

```
                        0D55     ;
                        0D55     ; This routine does "typical" UETP log file output, i.e., it reads a file,
                        0D55     ; searching for a keyword used as a beginning sentinel.  When the sentinel
                        0D55     ; is found, all lines of the file up to, but not including, a line containing
                        0D55     ; and ending sentinel are copied to SYS$OUTPUT.  An end of file counts as an
                        0D55     ; ending sentinel.  The copying algorithm is done for all the driven
                        0D55     ; processes started up by this driver.
                        0D55     ;
                        0D55     TYPE_LOG_FILES:
                 56  D4 0D55             CLRL    R6                          ; Initialize pointer to table of...
                        0D57                                                 ; ...qualifiers for driven proc names
                        0D57     10$:
          0020'CF46  D5 0D57             TSTL    PROCIDS[R6]                 ; Was this process ever started?
                 03  12 0D5C             BNEQ    15$                         ; BR if it was
               00CC  31 0D5E             BRW     60$                         ; It wasn't. There can be no log to copy
                        0D61     15$:
  0034'CF 02A9'CF46  90 0D61             MOVB    PROCS[R6],LOG_FILE_QUAL     ; Set up name for this log file
                        0D69             $OPEN   FAB = LOG_FAB               ; Open the log file
               012A  30 0D74             BSBW    FAB_ERROR
              11 50  E9 0D77             BLBC    R0,17$                      ; Don't try to read if we have error
                        0D7A             $CONNECT RAB = LOG_RAB              ; Connect to log file
               010B  30 0D85             BSBW    RAB_ERROR
              03 50  E8 0D88             BLBS    R0,20$                      ; BR if all OK
                        0D8B     17$:
               0089  31 0D8B             BRW     50$                         ; Don't try to read if we have error
                        0D8E     20$:
                        0D8E             $GET    RAB = LOG_RAB               ; Read a record looking for sentinel
              31 50  E9 0D99             BLBC    R0,30$                      ; BR if error
        54  022C'CF  9B 0D9C             MOVZBW  BEGUN_ADDR,R4               ; Form MATCHC length argument
        022D'CF  54  39 0DA1             MATCHC  R4,BEGUN_ADDR+1,-           ; Did we find a beginning sentinel?
   0525'CF  0084 8F     0DA6                     #TEXT_BUFFER,RMS_BUFFER
                 E0  12 0DAC             BNEQ    20$                         ; Loop for next record if not
                        0DAE             $GET    RAB = LOG_RAB               ; Anticipate the next record
        54  0232'CF  9B 0DB9             MOVZBW  END_ADDR,R4                 ; Form MATCHC length argument
        0233'CF  54  39 0DBE             MATCHC  R4,END_ADDR+1,-             ; Did we find an ending sentinel?
   0525'CF  0084 8F     0DC3                     #TEXT_BUFFER,RMS_BUFFER
                 4C  13 0DC9             BEQL    50$                         ; BR if we did - nothing to copy
                 67  10 0DCB             BSBB    TYPE_INTRO                  ; Introduce anything we might copy
                        0DCD     30$:
                        0DCD             $GET    RAB = LOG_RAB               ; Read a record to copy
              2D 50  E9 0DD8             BLBC    R0,40$                      ; BR if error
        54  0232'CF  9B 0DDB             MOVZBW  END_ADDR,R4                 ; Form MATCHC length argument
        0233'CF  54  39 0DE0             MATCHC  R4,END_ADDR+1,-             ; Did we find an ending sentinel?
   0525'CF  0084 8F     0DE5                     #TEXT_BUFFER,RMS_BUFFER
                 2A  13 0DEB             BEQL    50$                         ; BR if we did
         061E'CF 04  A1 0DED             ADDW3   #INDENT,LOG_RAB+RAB$W_RSZ,- ; Set the size of our message
            051D'CF     0DF2                     LOG_MSGPTR
                        0DF5             $PUTMSG_S MSGVEC = LOG_MSGVEC       ; All driven proc msgs go to SYS$OUTPUT
                 C5  11 0E06             BRB     30$                         ; Do the next record
                        0E08     40$:
     00000000'8F 50  D1 0E08             CMPL    R0,#RMS$_EOF                ; End of file ?
                 06  13 0E0F             BEQL    50$                         ; BR if it is
               007F  30 0E11             BSBW    RAB_ERROR                   ; Give an error otherwise...
              16 50  E9 0E14             BLBC    R0,60$                      ; ...and skip the rest of this one
                        0E17     50$:
                        0E17             $CLOSE  FAB = LOG_FAB               ; Close the log file, ignoring errors
                        0E22             $ERASE  FAB = LOG_FAB               ; Delete the log file, ignoring errors
```

```
                         0E2D  2544 60$:
FF24 56    01    01   F1  0E2D  2545          ACBL    #PROC_COUNT-1,#1,R6,10$ ; Loop through all driven processes
                      05  0E33  2546          RSB                              ; Return to mainline code
```

```
                                 0E34  TYPE_INTRO:
     051D'CF   00000084 8F  D0  0E34            MOVL    #TEXT_BUFFER,LOG_MSGPTR ; Set up...
          50    02A9'C6     9E  0E3D            MOVAB   PROCS(R6),R0            ; ...a message...
                                0E42            $FAO_S  CTRSTR = COPY_LOG_MSG,- ; ...to introduce the log file...
                                0E42                    OUTLEN = LOG_MSGPTR,-
                                0E42                    OUTBUF = LOG_MSGPTR,-
                                0E42                    P1     = #LOG_FILE_DESC,-
                                0E42                    P2     = #1,-
                                0E42                    P3     = R0
                                0E5F            $PUTMSG_S MSGVEC = LOG_MSGVEC   ; ...if there is stuff to copy
                     02     90  0E70            MOVB    #RAB$C_RFA,-            ; Now set up...
                061A'CF         0E72                    LOG_RAB+RAB$B_RAC
                                0E75            $FIND   RAB = LOG_RAB,-         ; ...so that we can...
                                0E75                    ERR = RMS_ERROR
                     00     90  0E84            MOVB    #RAB$C_SEQ,-            ; ...reread that last record
                061A'CF         0E86                    LOG_RAB+RAB$B_RAC
      20      00 8F     00  2C  0E89            MOVC5   #0,#0,#^A/ /,=          ; Set up indentation to mark...
            0525'CF     04      0E8E                    #INDENT,RMS_BUFFER      ; ...subprocess log from our messages
                        05      0E92            RSB
                                0E93
                                0E93  RAB_ERROR:
                  01 50 E9      0E93            BLBC    R0,10$                  ; If there is no error...
                        05      0E96            RSB                             ; ...we take no action
                                0E97  10$:
                0608'CF DD      0E97            PUSHL   LOG_RAB+RAB$L_STV       ; Get the specific error
                0604'CF DD      0E9B            PUSHL   LOG_RAB+RAB$L_STS
                     0C 11      0E9F            BRB     ALL_ERROR               ; Rest of code is same for FAB & RAB
                                0EA1
                                0EA1
                                0EA1
                                0EA1  FAB_ERROR:
                  01 50 E9      0EA1            BLBC    R0,10$                  ; If there is no error...
                        05      0EA4            RSB                             ; ...we take no action
                                0EA5  10$:
                05B8'CF DD      0EA5            PUSHL   LOG_FAB+FAB$L_STV       ; Get the specific error
                05B4'CF DD      0EA9            PUSHL   LOG_FAB+FAB$L_STS
                                0EAD            ;BRB    ALL_ERROR               ; Rest of code is same for FAB & RAB
                                0EAD
                                0EAD
                                0EAD
                                0EAD  ALL_ERROR:
                     50 DD      0EAD            PUSHL   R0                      ; Register must be preserved
                                0EAF            $FAO_S  CTRSTR = RMS_ERRMSG,-   ; RMS error message
                                0EAF                    OUTLEN = BUFFER_PTR,-
                                0EAF                    OUTBUF = FAO_BUF,-
                                0EAF                    P1     = #LOG_FILE_DESC
                  50 8ED0       0EC8            POPL    R0                      ; Register must be preserved
                0481'CF DF      0ECB            PUSHAL  BUFFER_PTR
                     01 DD      0ECF            PUSHL   #1                      ; Arg count
          00741132 8F DD        0ED1            PUSHL   #UETP$_TEXT!STS$K_ERROR ; Message code
     00000000'GF  05  FB        0ED7            CALLS   #5,G^LIB$SIGNAL         ; Output - note that R0 is preserved
                        05      0EDE            RSB                             ; Return with R0 intact
                                0EDF
                                0EDF            .END    UETLOCK00
```

$$.TAB	= 00000690	R	03	DEADLK_ERRMSG	0000068E	R	02
$$.TABEND	= 000006D4	R	03	DEADLK_M	= 00000010		
$$.TMP	= 00000000			DEADLK_V	= 00000004		
$$.TMP1	= 00000002			DEQ	= 00000002		
$$.TMP2	= 000000CF			DEQ$_ACMODE	= 0000000C		
$$.TMPX	= 00000000	R	04	DEQ$_FLAGS	= 00000010		
$$.TMPX1	= 0000000A			DEQ$_LKID	= 00000004		
$$ARGS	= 00000004			DEQ$_NARGS	= 00000004		
$$T1	= 00000000			DEQ$_VALBLK	= 00000008		
$$T2	= 00000004			DEQALL_ERRMSG	00000651	R	02
ACC$K_TERMLEN	= 00000054			DEQLST	0000006F	R	03
ACC$L_FINALSTS	= 00000004			DEQS	0000051C	R	05
ACC$W_MSGTYP	= 00000000			DEQ_CODE	000009F5	R	02
ALL_ERROR	00000EAD	R	05	DETPRC_ERRMSG	00000852	R	02
ALL_PROCS	000002A8	R	02	DEV$V_MBX	= 00000014		
ARG_COUNT	0000050D	R	03	DEV$V_TRM	= 00000002		
BASE_PRI	000000CE	R	03	DIB$K_LENGTH	= 00000074		
BAST_CEF	= 00000041			DIB$W_UNIT	= 0000000C		
BAST_CEF_V	= 00000001			DIBBUF	000000F2	R	03
BAST_ERRMSG	000003D7	R	02	DIBBUF_DESC	000000EA	R	03
BEGIN_MSGM	= 00000004			DLDET_CEF	= 00000044		
BEGIN_MSGV	= 00000002			DLDET_CEF_V	= 00000004		
BEGUN_ADDR	0000022C	R	02	DLMASK	= 00000030		
BLKAST_M	= 00000001			DLOCK_AST	0000083F	R	05
BLKAST_V	= 00000000			DLOCK_TIME	000000A6	R	03
BLOCK_AST	00000807	R	05	DLOCK_TO_AST	00000903	R	05
BUFFER	00000489	R	03	DLPRC_M	= 00000002		
BUFFER_PTR	00000481	R	03	DLPRC_V	= 00000001		
CASTPAR_ERRMSG	00000728	R	02	DLRES_CEF	= 00000045		
CASTSYNCH_ERRMSG	00000465	R	02	DONE_CEF	= 00000040		
CAST_ERRMSG	000003AC	R	02	DRIVEN	000002B8	R	05
CCASTHAND	00000B9D	R	05	DRIVEN_DESC	00000000	R	03
CHECK_UP	000005A7	R	05	DRIVEN_M	= 00000001		
CHF$L_SIGARGLST	= 00000004			DRIVEN_V	= 00000000		
CHF$L_SIG_ARG1	= 00000008			DRIVER	000001A9	R	05
CHF$L_SIG_ARGS	= 00000000			DUMP_COMMAND	0000052F	R	05
CHF$L_SIG_NAME	= 00000004			DUMP_M	= 00000002		
CHK_BLOCKAST	000005EF	R	05	DUMP_MSG	00000969	R	02
CHK_CAST	000005E1	R	05	DUMP_V	= 00000001		
CHK_DEADLOCK	00000763	R	05	DVI$_DEVNAM	= 00000020		
CHK_LKSB	000007B9	R	05	EF_STATE	000001E1	R	03
CHK_NOQUEUE	0000072A	R	05	ENDTEST	= 00000003		
CHK_SS	0000079C	R	05	END_ADDR	00000232	R	02
CHK_SYNCSTS	00000685	R	05	ENQ	= 00000000		
CHK_VALBLK	0000060A	R	05	ENQ$_ACMODE	= 00000028		
CMP_VAL	= 00000042			ENQ$_ASTADR	= 0000001C		
CMP_VAL_V	= 00000002			ENQ$_ASTPRM	= 00000020		
CNTRLCMSG	000002BB	R	02	ENQ$_BLKAST	= 00000024		
COMMAND_SIZE	= 00000008			ENQ$_EFN	= 00000004		
COMMON	00000B64	R	05	ENQ$_FLAGS	= 00000010		
COMP_AST	000007D6	R	05	ENQ$_LKMODE	= 00000008		
COPY_LOG_MSG	0000090C	R	02	ENQ$_LKSB	= 0000000C		
CREATE_PROCS	000002FA	R	05	ENQ$_NARGS	= 0000000B		
CRMODE_CODE	00000A19	R	02	ENQ$_PARID	= 00000018		
CWMODE_CODE	00000A1C	R	02	ENQ$_PROT	= 0000002C		
C_AST_M	= 00000001			ENQ$_RESNAM	= 00000014		
C_AST_V	= 00000000			ENQLST	0000003F	R	03

```
ENQS                             00000502 R       05
ENQW                           = 00000001
ENQWS                            0000050F R       05
ENQW_CODE                        000009EF R       02
ENQ_CODE                         000009EA R       02
ERROR_EXIT                       00000C02 R       05
ERR_IN_TABLE                     00000325 R       02
EXECUTE                          000003F2 R       05
EXIT_DESC                        00000465 R       03
EXIT_HANDLER                     00000CAA R       05
EXIT_MESSAGE                     00000D1F R       05
EXIT_MSG                         0000017E R       03
EXMODE_CODE                      00000A25 R       02
EX_MBXCHAN                       0000017C R       03
EX_MBX_AST                       00000929 R       05
EX_PROC_CNT                      000001D2 R       03
FAB$B_BID                      = 00000000
FAB$B_FNS                      = 00000034
FAB$C_BID                      = 00000003
FAB$C_BLN                      = 00000050
FAB$C_SEQ                      = 00000000
FAB$C_VAR                      = 00000002
FAB$L_ALQ                      = 00000010
FAB$L_DEV                      = 00000040
FAB$L_FNA                      = 0000002C
FAB$L_FOP                      = 00000004
FAB$L_STS                      = 00000008
FAB$L_STV                      = 0000000C
FAB$V_CHAN_MODE                = 00000002
FAB$V_FILE_MODE                = 00000004
FAB$V_LNM_MODE                 = 00000000
FAB$W_GBC                      = 00000048
FAB_ERROR                        00000EA1 R       05
FAIL_ADDR                        0000023D R       02
FAIL_OUT                         000002ED R       05
FAO_BUF                          00000479 R       03
FILE                             00000898 R       02
FORCEX_MSG                       0000093D R       02
GETSYI_ITMLST                    00000A2B R       02
GETUIC                           000000AE R       03
GET_COMMAND                      00000251 R       05
GET_NEXTMSG                      000002B8 R       05
GLBFLG                           000001DE R       03
HIGH_PROC                      = 00000052
ILL_PROC_NAME                    000002EA R       02
INCOMP_M                       = 00000002
INCOMP_V                       = 00000001
INDENT                         = 00000004
INPUT_FAB                        00000640 R       03
INPUT_ITMLST                     00000261 R       02
INPUT_RAB                        00000690 R       03
IO$M_CTRLCAST                    ********     X   05
IO$_READVBLK                     ********     X   05
IO$_SETMODE                      ********     X   05
IO$_WRITEVBLK                    ********     X   05
JPI$_PRIB                      = 00000309
JPI$_UIC                       = 00000304

LCK$GL_WAITTIME                  ********     X   05
LCK$K_CRMODE                   = 00000001
LCK$K_CWMODE                   = 00000002
LCK$K_EXMODE                   = 00000005
LCK$K_NLMODE                   = 00000000
LCK$K_PRMODE                   = 00000003
LCK$K_PWMODE                   = 00000004
LCK$M_CONVERT                  = 00000002
LCK$M_DEQALL                   = 00000001
LCK$M_NOQUEUE                  = 00000004
LCK$M_SYNCSTS                  = 00000008
LCK$M_VALBLK                   = 00000001
LCK$V_CONVERT                  = 00000001
LCK$V_NOQUEUE                  = 00000002
LCK$V_SYNCSTS                  = 00000003
LCK$V_VALBLK                   = 00000000
LIB$SIGNAL                       ********     X   05
LKID_ADDR                        000001ED R       03
LKSBS                            000001F5 R       03
LKSB_ADDR                        000001F1 R       03
LKSB_SIZE                      = 00000018
LKTEST_DESC                      00000293 R       02
LKVAL_ERRMSG                     00000400 R       02
LK_CEF_DESC                      00000281 R       02
LK_UIC                           000000CA R       03
LNM$_STRING                    = 00000002
LNMGRP                         = 0000000A
LNMGRPLEN                      = 00000010
LNMGRPNIL                        000000E2 R       03
LNMGRPNUM                        000000D2 R       03
LNMITMLST                        00000A75 R       02
LNMPRCDIR                        00000A3B R       02
LNMTMPMBX                        00000A58 R       02
LOCFLG                           000001E0 R       03
LOCK_MODES                       000009FA R       02
LOG_FAB                          000005AC R       03
LOG_FILE_DESC                    0000002C R       03
LOG_FILE_LEN                   = 0000000B
LOG_FILE_QUAL                    00000034 R       03
LOG_MSGPTR                       0000051D R       03
LOG_MSGVEC                       000002AB R       02
LOG_RAB                          000005FC R       03
LOW_PROC                       = 00000050
MAXCODE                        = 00000003
MBXCHANS                         0000001C R       03
MBXNAM                           0000016E R       03
MBXNAM_LEN                     = 00000006
MBX_DESC                         00000166 R       03
MBX_IOSB                         00000174 R       03
MBX_QIO                          000003AC R       05
MODE                             00000244 R       02
MSG$_DELPROC                   = 00000003
MSGTYP_ERRMSG                    000007A3 R       02
MSG_BLOCK                        00000475 R       03
NLMODE_CODE                      00000A16 R       02
NOCAST_M                       = 00000020
NOCAST_V                       = 00000005
```

```
NODLOCK_ERRMSG          000006C8 R     02
NOMODE_CODE             00000A28 R     02
NONE                    000009DA R     02
NONE_LENGTH           = 00000004
NOSYNCH_ERRMSG          00000435 R     02
NOSYQUEUE_ERRMSG        00000544 R     02
NOTRMB_ERRMSG           00000760 R     02
NOWAIT_M              = 00000040
NOWAIT_V              = 00000006
OTS$CVT_L_TO            ********    X  05
PAR_ERRMSG              000005BA R     02
PRMODE_CODE             00000A1F R     02
PROCIDS                 00000020 R     03
PROCS                 = 000002A9 R     02
PROC_COUNT            = 00000002
PROC_QUALIFIER          0000001B R     03
PWMODE_CODE             00000A22 R     02
QUAD_STATUS             00000515 R     03
RAB$B_RAC             = 0000001E
RAB$C_BID             = 00000001
RAB$C_BLN             = 00000044
RAB$C_RFA             = 00000002
RAB$C_SEQ             = 00000000
RAB$L_CTX             = 00000018
RAB$L_FAB             = 0000003C
RAB$L_ROP             = 00000004
RAB$L_STS             = 00000008
RAB$L_STV             = 0000000C
RAB$W_RSZ             = 00000022
RAB_ERROR               00000E93 R     05
RECORD                  000008A4 R     02
RESR                    000000A1 R     03
RESR_DESC               0000008D R     03
RESR_LEN              = 00000011
RMS$_EOF                ********    X  05
RMS$_FACILITY         = 00000001
RMS_BUFFER              00000525 R     03
RMS_ERRMSG              000008D3 R     02
RMS_ERROR               00000B2E R     05
RMS_ERR_STRING          000008B2 R     02
SAVED_VAL               000001E5 R     03
SCSNODE                 0000009A R     03
SEND_MSG                00000389 R     05
SHR$_ABENDD           = 000010E0
SHR$_BEGIND           = 00001038
SHR$_ENDEDD           = 00001080
SHR$_TEXT             = 00001130
SNDMBX_ERRMSG           0000036E R     02
SNDMSG_EFN            = 00000004
SS$_CONTROLC            ********    X  05
SS$_DEADLOCK            ********    X  05
SS$_NORMAL              ********    X  05
SS$_NOTQUEUED           ********    X  05
SS$_SSFAIL              ********    X  05
SS$_SYNCH               ********    X  05
SS$_TIMEOUT             ********    X  05
SS$_WASSET              ********    X  05
SSERROR                 00000A2F R     05
SS_SYNCH_EFN          = 00000005
STATUS                  00000511 R     03
STS$K_ERROR           = 00000002
STS$K_INFO            = 00000003
STS$K_SUCCESS         = 00000001
STS$K_WARNING         = 00000000
STS$M_INHIB_MSG       = 10000000
STS$S_FAC_NO          = 0000000C
STS$S_SEVERITY        = 00000003
STS$V_FAC_NO          = 00000010
STS$V_SEVERITY        = 00000000
SUBPROC_STRING          000002DC R     02
SUC_EXIT                000002D9 R     05
SYI$_SCSNODE          = 00001067
SYNCH_ERRMSG            000004D3 R     02
SYNCST_M              = 00000008
SYNCST_V              = 00000003
SYNC_EF               = 00000001
SYS$ASCEFC              ********   GX  05
SYS$ASSIGN              ********   GX  05
SYS$CANTIM              ********   GX  05
SYS$CLOSE               ********   GX  05
SYS$CLREF               ********   GX  05
SYS$CONNECT             ********   GX  05
SYS$CRELNM              ********   GX  05
SYS$CREMBX              ********   GX  05
SYS$CREPRC              ********   GX  05
SYS$DCLEXH              ********   GX  05
SYS$DEQ                 ********   GX  05
SYS$ENQ                 ********   GX  05
SYS$ENQW                ********   GX  05
SYS$ERASE               ********   GX  05
SYS$EXIT                ********   GX  05
SYS$FAO                 ********    X  05
SYS$FIND                ********   GX  05
SYS$FORCEX              ********   GX  05
SYS$GET                 ********   GX  05
SYS$GETCHN              ********   GX  05
SYS$GETDVI              ********   GX  05
SYS$GETJPI              ********   GX  05
SYS$GETMSG              ********   GX  05
SYS$GETSYI              ********   GX  05
SYS$HIBER               ********   GX  05
SYS$INPUT               00000250 R     02
SYS$OPEN                ********   GX  05
SYS$PUTMSG              ********   GX  05
SYS$QIO                 ********   GX  05
SYS$QIOW                ********   GX  05
SYS$READEF              ********   GX  05
SYS$SCHDWK              ********   GX  05
SYS$SETAST              ********   GX  05
SYS$SETEF               ********   GX  05
SYS$SETIMR              ********   GX  05
SYS$SETPRN              ********   GX  05
SYS$SETSFM              ********   GX  05
SYS$TRNLOG              ********   GX  05
```

```
SYS$WAITFR                   ********  GX    05
SYS$WAKE                     ********  GX    05
SYS$WFLAND                   ********  GX    05
TABLE_END                    00000218 R      02
TEN_SECONDS                  00000271 R      02
TERMBX_ERRMSG                000007E6 R      02
TEST_CODES                   000009DE R      02
TEST_NAME_C                  00000010 R      03
TEST_NAME_D                  00000008 R      03
TEST_NAME_I                  00000011 R      03
TEST_NAME_LEN              = 0000000A
TEST_TABLE                   00000000 R      02
TEXT_BUFFER                = 00000084
THREE_MIN                    00000279 R      02
TIME_OUT                     00000A22 R      05
TO_BE_FILLED                 0000021C R      02
TSTFLG                       0000008B R      03
TST_COMMAND                  00000083 R      03
TTCHAN                       000001D6 R      03
TYPE                         000001DA R      03
TYPE_INTRO                   00000E34 R      05
TYPE_LOG_FILES               00000D55 R      05
UETLOCK00                    00000000 RG     05
UETP                       = 00740000
UETP$_ABENDD               = 007410E0
UETP$_ABORTC               = 0074832B
UETP$_BEGIND               = 00741038
UETP$_ENDEDD               = 00741080
UETP$_FACILITY             = 00000074
UETP$_SATSMS               = 007480D9
UETP$_TEXT                 = 00741130
UNLOCK_CEF                 = 00000043
VALBLK_M                   = 00000004
VALBLK_V                   = 00000002
VICTIM_ERRMSG                000005F6 R      02
VICTIM_M                   = 00000080
VICTIM_V                   = 00000007
WAIT_PRCS                    000002AE R      05
WAKE_UP                      00000A16 R      05

                                   +----------------+
                                   ! 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
RODATA                       00000A85  (  2693.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD   NOWRT NOVEC PAGE
RWDATA                       000006D4  (  1748.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD     WRT NOVEC PAGE
$RMSNAM                      0000000A  (    10.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
TESTLOCK                     00000EDF  (  3807.)  05 (  5.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD   NOWRT NOVEC PAGE
```

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

Phase                       Page faults     CPU Time        Elapsed Time
-----                       -----------     --------        ------------
Initialization                      37      00:00:00.08     00:00:00.65
Command processing                 140      00:00:00.66     00:00:02.55
Pass 1                             838      00:00:29.75     00:00:59.11
Symbol table sort                    0      00:00:02.47     00:00:05.09
Pass 2                             468      00:00:08.45     00:00:14.43
Symbol table output                  2      00:00:00.30     00:00:00.93
Psect synopsis output                2      00:00:00.03     00:00:00.11
Cross-reference output               0      00:00:00.00     00:00:00.00
Assembler run totals              1490      00:00:41.74     00:01:22.88

The working set limit was 2000 pages.
160602 bytes (314 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1649 non-local and 83 local symbols.
2601 source lines were read in Pass 1, producing 45 object records in Pass 2.
88 pages of virtual memory were used to define 79 macros.

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

Macro library name                           Macros defined
------------------                           --------------
_$255$DUA28:[SHRLIB]UETP.MLB;1                            1
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                            0
_$255$DUA28:[SYSLIB]STARLET.MLB;2                        72
TOTALS (all libraries)                                   73

2025 GETS were required to define 73 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:UETLOCK00/OBJ=OBJ$:UETLOCK00 MSRC$:UETLOCK00/UPDATE=(ENH$:UETLOCK00)+EXECML$/LIB+SHRLIB$:UETP/LIB
```

0427 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

UETINIT00
LIS

UETINIT01
LIS

UETLOCK00
LIS

UETMA7800
LIS