```
UUU          UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSS  YYY          YYY
UUU          UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSS  YYY          YYY
UUU          UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSS  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
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP                SSSSSSSSSSS              YYY
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP                SSSSSSSSSSS              YYY
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP                SSSSSSSSSSS              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

(2)	74	Declarations	20
(3)	233	Read-Only Data	75
(5)	600	Read/Write Data	
(6)	773	RMS-32 Data Structures	
(7)	800	TESTLOCK - Initialization	
(8)	915	TESTLOCK - Driver Process	
(9)	985	TESTLOCK - Driven Process	
(10)	1003	Clean Up and Exit	64
(11)	1021	Create Detached Processes and Mailboxes	
(12)	1090	Determine Which Driven Process Gets Command	41
(13)	1130	Send Messages to Driven Processes	66
(14)	1180	Execute Test Commands in the Test Table	
(15)	1289	Type the Current Command	
(16)	1352	Check Correctness of the ENQW Test	
(17)	1405	Check Completion AST	72
(18)	1439	Check Blocking AST	6F
(19)	1472	Check Lock Value Block	6F
(20)	1530	Check SYNCSTS Flag Routine	6C
(21)	1602	Check NOQUEUE Flag Routine	
(22)	1646	Check Deadlock Test	
(23)	1692	Check Status Code Subroutine	
(24)	1749	Completion AST Routine	6F
(25)	1794	Blocking AST Routine	66
(26)	1837	Deadlock Detection and Resolution Routine	70
(28)	1912	Deadlock Timeout AST Routine	
(29)	1954	Termination Mailbox AST Routine	
(30)	2053	Timer Expiration Routine	
(31)	2089	System Service Exception Handler	73
(32)	2221	RMS Error Handler	6F
(33)	2285	CTRL/C Handler	78
(34)	2343	Error Exit	
(35)	2412	Exit Handler	
			73
			67
			30
			21
			58
			63
			74
			SE

```
0000     1          .TITLE  UETLOCK00 - Local Lock Manager UETP Test
0000     2          .IDENT  'V04-000'
0000     3          .ENABLE SUPPRESSION
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27
0000    28 ;++
0000    29 ; FACILITY: VAX/VMS UETP
0000    30 ;
0000    31 ; ABSTRACT:
0000    32 ;       This module serves as both the controlling (driver) and slave (driven)
0000    33 ;       images of the UETP Lock Manager Test.  Depending on the invocation and
0000    34 ;       parameters supplied, it chooses to be the driver, in which case it
0000    35 ;       starts up copies of itself and directs their actions; or the driven
0000    36 ;       process, in which case it is started by the driver and is told what
0000    37 ;       action to perform next.
0000    38 ;
0000    39 ; ENVIRONMENT:
0000    40 ;       The DETACH and GROUP privileges and ENQLM=20 are needed to run this
0000    41 ;       test.
0000    42 ;
0000    43 ;--
0000    44 ;
0000    45 ; AUTHOR:  Paul Jenq,   CREATION DATE:  8-Aug-1981                          52
0000    46 ;                                                                          54
0000    47 ; MODIFIED BY:
0000    48 ;
0000    49 ;       V03-006 RNH0005         Richard N. Holstein,    26-Apr-1984
0000    50 ;               V3-5 left SYI$_SCSNODE at 4 instead of 6.  Fix that and force  45
0000    51 ;               mailbox logical names to go into a group logical name table.   4C
0000    52 ;
0000    53 ;       V03-005 WHM0001         Bill Matthews           14-Apr-1984
0000    54 ;               Replace reference to SYI$_SCSNODEL/H with SYI$_SCSNODE.
0000    55 ;
0000    56 ;       V03-004 RNH0004         Richard N. Holstein,    23-Nov-1983
0000    57 ;               Use lock names which will allow the test to run on more than
```

```
0000     58 ;                  one VMS node of a cluster simultaneously - insert 'UETP$node_'.
0000     59 ;
0000     60 ;       V03-003 RNH0003         Richard N. Holstein,    17-Nov-1983
0000     61 ;               Fix bug which prevented deleting log files on successful runs.
0000     62 ;
0000     63 ;       V03-002 RNH0002         Richard N. Holstein,    11-Mar-1983
0000     64 ;               Don't signal ending message in EXIT_HANDLER.
0000     65 ;
0000     66 ;       V03-001 RNH0001         Richard N. Holstein,    11-Mar-1983
0000     67 ;               Eliminate redundant error checking when opening SYS$INPUT.
0000     68 ;               Don't try for subprocess logs if the subprocesses weren't
0000     69 ;               created.  Indent copied subprocess logs.  Misc fixes carried
0000     70 ;               over from device tests.
0000     71 ;
0000     72 ;**
```

```
0000     74          .SBTTL  Declarations
0000     75 ;
0000     76 ; INCLUDE FILES:
0000     77 ;
0000     78 ;       SYS$LIBRARY:LIB.MLB     for general definitions
0000     79 ;       SHRLIB$:UETP.MLB        for UETP definitions
0000     80 ;
0000     81
0000     82 ;
0000     83 ; MACROS:
0000     84 ;
0000     85          $ACCDEF                 ; Account msg definition
0000     86          $CHFDEF                 ; Condition handler
0000     87          $DEVDEF                 ; Device characteristics
0000     88          $DIBDEF                 ; Device infomation block
0000     89          $DVIDEF                 ; $GETDVI ITMLST item codes
0000     90          $JPIDEF                 ; Job/proc information def
0000     91          $LCKDEF                 ; LCK definition
0000     92          $LNMDEF                 ; Logical name services codes
0000     93          $MSGDEF                 ; Message definition
0000     94          $SHRDEF                 ; Shared message
0000     95          $STSDEF                 ; Status definition
0000     96          $SYIDEF                 ; $GETSYI ITMLST item codes
0000     97          $UETPDEF                ; UETP messages
0000     98 ;
0000     99 ; Macro to build test tab e entries.  NOTE WELL!  Some code depends on this
0000    100 ; table having entries which are eight bytes long (i.e., a quadword).  This
0000    101 ; length is available as the parameter COMMAND_SIZE.
0000    102 ;
0000    103 .MACRO TST_TABLE PROC,TYP,LKMOD,RESR=<A>,PAR=< >,FLAG,TST_FLG=<^X0000>
0000    104         .IIF LT ^A/PROC/-LOW_PROC,-
0000    105                 .ERROR 0                ; Illegal proc name PROC in TST_TABLE
0000    106         .IIF GT ^A/PROC/-HIGH_PROC,-
0000    107                 .ERROR 0                ; Illegal proc name PROC in TST_TABLE
0000    108         .ASCII  /PROC/                  ; Process name to perform the action
0000    109         .BYTE   TYP                     ; Type of system svc: ENQ, ENQW or DEQ
0000    110         .IIF LT TYP, .ERROR TYP         ; Illegal TYP-argument in TST_TABLE
0000    111         .IIF GT TYP-MAXCODE, .ERROR TYP ; Illegal TYP-argument in TST_TABLE
0000    112         .IF B LKMOD
0000    113                 .BYTE   ^XFF            ; Type of lock - $ENQ LKMODE argument
0000    114         .IFF
0000    115                 .BYTE   LCK$K_'LKMOD'MODE ; Type of lock - $ENQ LKMODE argument
0000    116         .ENDC
0000    117         .ASCII  /RESR/                  ; One character resource name
0000    118         .ASCII  /PAR/                   ; One character parent lock
0000    119         .IF B FLAG
0000    120                 .BYTE   0               ; Lock characs - $ENQ FLAGS argument
0000    121         .IFF
0000    122                 .BYTE   LCK$M_'FLAG     ; Lock characs - $ENQ FLAGS argument
0000    123         .ENDC
0000    124         .WORD   TST_FLG                 ; Test action after calling system svc
0000    125 .ENDM   TST_TABLE
0000    126
0000    127 ;
0000    128 ; This next macro is used to build the table of process names, to generate the
0000    129 ; parameters we need to check that we only use legal process names and to build
0000    130 ; various other data structures which rely on process names.  It calls the X
```

```
                     0000   131 ; macro, which must be defined each time to the specific purpose to be
                     0000   132 ; performed.  PROCESS NAMES MUST BE ONE ASCII CHARACTER!
                     0000   133 ;
                     0000   134 .MACRO  PROC_NAMES
                     0000   135         X       P                       ; Master (driver) process
                     0000   136         X       Q                       ; First slave (driven) process
                     0000   137         X       R                       ; Next slave (driven) process
                     0000   138 .ENDM   PROC_NAMES
                     0000   139
                     0000   140 ;
                     0000   141 ; EQUATED SYMBOLS:
                     0000   142 ;
                     0000   143
00000001             0000   144         RMS$_FACILITY = 1               ; Standard RMS facility code
00740000             0000   145         UETP = UETP$_FACILITY@STS$V_FAC_NO ; UETP facility code massaged
007410E0             0000   146         UETP$_ABENDD = UETP!SHR$_ABENDD ; Abort message codes
00741038             0000   147         UETP$_BEGIND = UETP!SHR$_BEGIND ; Begin message
00741080             0000   148         UETP$_ENDEDD = UETP!SHR$_ENDEDD ; End message
00741130             0000   149         UETP$_TEXT   = UETP!SHR$_TEXT   ; Text message
                     0000   150
                     0000   151 ; Code depends not only on individual values of the following, but also on
                     0000   152 ; their ordering.
00000000             0000   153         ENQ     = 0                     ; Do ENQ. Code depends on this value
00000001             0000   154         ENQW    = 1                     ; Do ENQW. Code depends on this value
00000002             0000   155         DEQ     = 2                     ; Do DEQ
00000003             0000   156         ENDTEST = 3                     ; END OF TEST code
00000003             0000   157         MAXCODE = 3                     ; Greatest legal TYP field in TST_TABLE
                     0000   158
                     0000   159 ; Generate high and low bounds for process names.  Count process names.  Note
                     0000   160 ; that the first process name is assumed to be the master process and that we
                     0000   161 ; don't include it in the count.
                     0000   162 .MACRO  X       PROCESS
                     0000   163         .IIF NDF PROC_COUNT, PROC_COUNT = -1 ; Initialize count of processes
                     0000   164         PROC_COUNT = PROC_COUNT+1        ; First is master, don't count it
                     0000   165         .IIF NDF LOW_PROC,   LOW_PROC   = ^A/PROCESS/ ; Low bound
                     0000   166         .IIF NDF HIGH_PROC,  HIGH_PROC  = ^A/PROCESS/ ; Initial high bound
                     0000   167         .IIF LT  HIGH_PROC-^A/PROCESS/, HIGH_PROC = ^A/PROCESS/ ; High bound
                     0000   168 .ENDM   X       ; For process name bounds
                     0000   169         PROC_NAMES                      ; Generate bounds and count processes
                     0000   170
                     0000   171 ;
                     0000   172 ; Common event flags
                     0000   173 ;
00000040             0000   174         DONE_CEF    = 64                ; CEF for action complete by driven proc
00000041             0000   175         BAST_CEF    = 65                ; CEF for blocking ast delivery...
00000001             0000   176         BAST_CEF_V = <65-64>            ; ...and correspoinding bit
00000042             0000   177         CMP_VAL     = 66                ; CEF for compare value...
00000002             0000   178         CMP_VAL_V  = <66-64>            ; ...and corresponding bit
00000043             0000   179         UNLOCK_CEF = 67                 ; CEF for unlock
00000044             0000   180         DLDET_CEF  = 68                 ; CEF for deadlock detected...
00000004             0000   181         DLDET_CEF_V= DLDET_CEF-64       ; ...and corresponding bit
00000045             0000   182         DLRES_CEF  = 69                 ; CEF for deadlock resolved
00000030             0000   183         DLMASK     = <1@<DLDET_CEF-64>>!- ; Mask for $WFLAND of deadlock flags
                     0000   184                      <1@<DLRES_CEF-64>>
                     0000   185
                     0000   186 ;
                     0000   187 ; Bits in TSTFLG.
```

```
                     0000   188 ;
00000000             0000   189         BLKAST_V = 0                    ; Blocking AST test
00000001             0000   190         INCOMP_V = 1                    ; Lock mode incompatibility test
00000002             0000   191         VALBLK_V = 2                    ; Value block test
00000003             0000   192         SYNCST_V = 3                    ; SYNCSTS test
00000004             0000   193         DEADLK_V = 4                    ; Deadlock test - don't know victim
00000005             0000   194         NOCAST_V = 5                    ; No completion AST
00000006             0000   195         NOWAIT_V = 6                    ; Do not wait for DONE CEF
00000007             0000   196         VICTIM_V = 7                    ; This is a deadlock victim for sure
                     0000   197
00000001             0000   198         BLKAST_M = 1@BLKAST_V           ; And corresponding bit masks
00000002             0000   199         INCOMP_M = 1@INCOMP_V
00000004             0000   200         VALBLK_M = 1@VALBLK_V
00000008             0000   201         SYNCST_M = 1@SYNCST_V
00000010             0000   202         DEADLK_M = 1@DEADLK_V
00000020             0000   203         NOCAST_M = 1@NOCAST_V
00000040             0000   204         NOWAIT_M = 1@NOWAIT_V
00000080             0000   205         VICTIM_M = 1@VICTIM_V
                     0000   206 ;
                     0000   207 ; Bits in LOCFLG.
                     0000   208 ;
00000000             0000   209         C_AST_V = 0                     ; Set if completion AST was delivered
00000001             0000   210         D[PRC_V = 1                     ; Set if $FORCEX done on driven procs
00000001             0000   211         C_AST_M = 1@C_AST_V
00000002             0000   212         D[PRC_M = 1@D[PRC_V
                     0000   213 ;
                     0000   214 ; Bits in GLBFLG.
                     0000   215 ;
00000000             0000   216         DRIVEN_V = 0                    ; Set if we are a driven (slave) proc
00000001             0000   217         DRIVEN_M = 1@DRIVEN_V
00000001             0000   218         DUMP_V   = 1                    ; Set if we're running in dump mode
00000002             0000   219         DUMP_M   = 1@DUMP_V
00000002             0000   220         BEGIN_MSGV = 2                  ; Set if we've printed our beginning msg
00000004             0000   221         BEGIN_MSGM = 1@BEGIN_MSGV
                     0000   222 ;
                     0000   223 ; Miscellaneous
                     0000   224 ;
00000001             0000   225         SYNC_EF      = 1                ; Local EF for SYNCSTS test
00000004             0000   226         SNDMSG_EFN   = 4                ; EFN for QIO to send to driven procs
00000084             0000   227         TEXT_BUFFER = 132               ; Buffer length for FAO output
00000018             0000   228         LKSB_SIZE    = 24               ; Size of lock status block in bytes
00000008             0000   229         COMMAND_SIZE= 8                 ; Size of a TEST_TABLE entry
00000005             0000   230         SS_SYNCH_EFN= 5                 ; Synch miscellaneous system services
00000004             0000   231         INDENT       = 4                ; Spaces to indent when copying logs
```

```
                        .SBTTL  Read-Only Data
00000000                .PSECT  RODATA,NOEXE,NOWRT,PAGE
                ;
                ; See the PROC_NAMES macro definition for restrictions on the names of
                ; processes.
                ;       TST_TABLE process,sstype,lock-mode,resource,parent-id,ssflags,action-flags
                ;

TEST_TABLE:
                ;
                ; Test of new locks grant
                ;
                        TST_TABLE <P>,ENQ,NL,<A>

                        TST_TABLE <P>,ENQ,CR,<B>

                        TST_TABLE <P>,ENQ,CW,<C>

                        TST_TABLE <P>,ENQ,PR,<D>

                        TST_TABLE <P>,ENQ,PW,<E>

                        TST_TABLE <P>,ENQ,EX,<F>
                ;
                ; Test of lock conversion
                ;
                        TST_TABLE <P>,ENQW,CR,<A>,,CONVERT

                        TST_TABLE <P>,ENQW,PR,<B>,,CONVERT

                        TST_TABLE <P>,ENQW,PW,<C>,,CONVERT

                        TST_TABLE <P>,ENQW,NL,<D>,,CONVERT

                        TST_TABLE <P>,ENQW,EX,<E>,,CONVERT

                        TST_TABLE <P>,ENQW,CW,<F>,,CONVERT
                ;
                ; Test of Deque
                ;
                        TST_TABLE <P>,DEQ,,<A>

                        TST_TABLE <P>,DEQ,,<B>

                        TST_TABLE <P>,DEQ,,<C>

                        TST_TABLE <P>,DEQ,,<D>

                        TST_TABLE <P>,DEQ,,<E>

                        TST_TABLE <P>,DEQ,,<F>
                ;
                ; Test of lock mode compatibility and blocking AST
                ;
                        TST_TABLE <P>,ENQW,NL,<A>,,,BLKAST_M

                        TST_TABLE <Q>,ENQW,CR,<A>
```

```
00A0   290
00A0   291          TST_TABLE <Q>,ENQW,EX,<A>,,CONVERT
00A8   292
00A8   293          TST_TABLE <P>,ENQW,CR,<B>,,,BLKAST_M
00B0   294
00B0   295          TST_TABLE <Q>,ENQW,CW,<B>
00B8   296
00B8   297          TST_TABLE <Q>,ENQW,PR,<B>,,CONVERT
00C0   298
00C0   299          TST_TABLE <Q>,ENQW,EX,<B>,,CONVERT,INCOMP_M
00C8   300
00C8   301          TST_TABLE <P>,ENQW,CW,<C>,,,BLKAST_M
00D0   302
00D0   303          TST_TABLE <Q>,ENQW,PR,<C>,,,INCOMP_M
00D8   304
00D8   305          TST_TABLE <Q>,DEQ,,<C>
00E0   306
00E0   307          TST_TABLE <P>,ENQW,CW,<C>,,,BLKAST_M
00E8   308
00E8   309          TST_TABLE <Q>,ENQW,CR,<C>
00F0   310
00F0   311          TST_TABLE <Q>,ENQW,EX,<C>,,CONVERT,INCOMP_M
00F8   312
00F8   313          TST_TABLE <P>,ENQW,PR,<D>,,,BLKAST_M
0100   314
0100   315          TST_TABLE <Q>,ENQW,CW,<D>,,,INCOMP_M
0108   316
0108   317          TST_TABLE <Q>,DEQ,,<D>
0110   318
0110   319          TST_TABLE <P>,ENQW,PR,<D>,,,BLKAST_M
0118   320
0118   321          TST_TABLE <Q>,ENQW,EX,<D>,,,INCOMP_M
0120   322
0120   323          TST_TABLE <P>,ENQW,PW,<E>,,,BLKAST_M
0128   324
0128   325          TST_TABLE <Q>,ENQW,CR,<E>
0130   326
0130   327          TST_TABLE <Q>,ENQW,PW,<E>,,CONVERT,INCOMP_M
0138   328
0138   329          TST_TABLE <Q>,DEQ,,<E>
0140   330
0140   331          TST_TABLE <P>,ENQW,EX,<F>,,,BLKAST_M
0148   332
0148   333          TST_TABLE <Q>,ENQW,CR,<F>,,,INCOMP_M
0150   334
0150   335          TST_TABLE <Q>,DEQ,,<F>
0158   336
0158   337          TST_TABLE <P>,ENQW,EX,<F>,,,BLKAST_M
0160   338
0160   339          TST_TABLE <Q>,ENQW,EX,<F>,,,INCOMP_M
0168   340 ;
0168   341 ; Test of value block
0168   342 ;
0168   343          TST_TABLE <P>,ENQW,CR,<V>,,VALBLK
0170   344
0170   345          TST_TABLE <Q>,ENQW,CR,<V>,,VALBLK
0178   346
```

```
0178   347          TST_TABLE <P>,ENQW,PW,<V>,,<VALBLK!LCK$M_CONVERT>,VALBLK_M
0180   348
0180   349          TST_TABLE <P>,ENQW,CR,<V>,,<VALBLK!LCK$M_CONVERT>,
0188   350
0188   351          TST_TABLE <Q>,ENQW,PW,<V>,,<VALBLK!LCK$M_CONVERT>,VALBLK_M
0190   352 ;
0190   353 ; Test of tree structured lock and LCK$M_NOQUEUE flag
0190   354 ;
0190   355          TST_TABLE <P>,ENQW,CW,<U>
0198   356
0198   357          TST_TABLE <P>,ENQW,PW,<X>,<U>
01A0   358
01A0   359          TST_TABLE <P>,ENQW,EX,<Z>,<X>
01A8   360
01A8   361          TST_TABLE <Q>,ENQW,CR,<U>
01B0   362
01B0   363          TST_TABLE <Q>,ENQW,PR,<X>,<U>,NOQUEUE,NOCAST_M
01B8   364 ;
01B8   365 ; Test of LCK$M_SYNCSTS flag
01B8   366 ;
01B8   367          TST_TABLE <P>,ENQW,PW,<S>,,SYNCSTS,<SYNCST_M!BLKAST_M>
01C0   368
01C0   369          TST_TABLE <Q>,ENQ,EX,<S>,,SYNCSTS
01C8   370 ;
01C8   371 ; Test of Local deadlock detection
01C8   372 ;
01C8   373          TST_TABLE <P>,ENQW,EX,<L>
01D0   374
01D0   375          TST_TABLE <Q>,ENQW,EX,<M>
01D8   376
01D8   377          TST_TABLE <P>,ENQ,EX,<M>,,,DEADLK_M
01E0   378
01E0   379          TST_TABLE <Q>,ENQ,EX,<L>,,,DEADLK_M!NOWAIT_M
01E8   380
01E8   381          TST_TABLE <R>,ENQW,NL,<N>
01F0   382
01F0   383          TST_TABLE <R>,ENQW,EX,<N>
01F8   384
01F8   385          TST_TABLE <R>,ENQW,EX,<N>,,,VICTIM_M
0200   386 ;
0200   387 ; Tell all detached processes to terminate.  Generate a termination message for
0200   388 ; all processes.  The driver process is smart enough to ignore the message
0200   389 ; because it must wait for all other processes to finish.
0200   390 ;
0200   391 .MACRO  X       PROCESS
0200   392          TST_TABLE <PROCESS>,ENDTEST,,,,,NOWAIT_M
0200   393 .ENDM   X       ; For termination list
0200   394          PROC_NAMES                          ; Generate list for termination msgs
0218   395
0218   396 TABLE_END:                                  ; End of Test table
00000000 0218   397          .LONG   0
021C   398 TO_BE_FILLED:                               ; For patch use
0000022C 021C   399          .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   450
                                        02DC   451 SUBPROC_STRING:                          ; Mailbox code to tell driven proc...
20 3A 63 6F 72 50 000002E4'010E0000'    02DC   452          .ASCID  /Proc: /                ; ...which one it is
                                        02EA   453                                          ; THIS STRING MUST FIT INTO A...
                                        02EA   454                                          ; ...MAILBOX USED FOR PASSING...
                                        02EA   455                                          ; ...COMMANDS TO DRIVEN PROCESSES!
                                        02EA   456
                                        02EA   457 ILL_PROC_NAME:                           ; Driven process name is not in our table
61 67 65 6C 6C 49 000002F2'010E0000'    02EA   458          .ASCID  /Illegal name for cooperating detached process: !AD./
63 20 72 6F 66 20 65 6D 61 65 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   459
                                        0325   460 ERR_IN_TABLE:
6E 72 65 74 6E 49 0000032D'010E0000'    0325   461          .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   462
                                        036E   463 SNDMBX_ERRMSG:
6E 64 6C 75 6F 43 00000376'010E0000'    036E   464          .ASCID  /Couldn't send mailbox to cooperating detached process./
6C 6E 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   465
                                        03AC   466 CAST_ERRMSG:
2C 72 6F 72 72 45 000003B4'010E0000'    03AC   467          .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   468
                                        03D7   469 BAST_ERRMSG:
2C 72 6F 72 72 45 000003DF'010E0000'    03D7   470          .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   471
                                        0400   472 LKVAL_ERRMSG:
76 20 6B 63 6F 4C 00000408'010E0000'    0400   473          .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   474
                                        0435   475 NOSYNCH_ERRMSG:
20 57 51 4E 45 24 0000043D'010E0000'    0435   476          .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   477
                                        0465   478 CASTSYNCH_ERRMSG:
20 57 51 4E 45 24 0000046D'010E0000'    0465   479          .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   480            <13><10>/       should not be free when LCK$M_SYNCSTS was set./
68 77 20 65 65 72 66 20 65 62 20 74  04AE
4E 59 53 5F 4D 24 4B 43 4C 20 6E 65  04BA
74 65 73 20 73 61 77 20 53 54 53 43  04C6
                                 2E  04D2
                                     04D3   481
                                     04D3   482 SYNCH_ERRMSG:
20 57 51 4E 45 24 000004DB'010E0000' 04D3   483         .ASCID  /$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   484            <13><10>/       SS$_SYNCSTS and delivered no completion AST./
76 69 6C 65 64 20 64 6E 61 20 53 54  0521
70 6D 6F 63 20 6F 6E 20 64 65 72 65  052D
   2E 54 53 41 20 6E 6F 69 74 65 6C  0539
                                     0544   485
                                     0544   486 NOSYQUEUE_ERRMSG:
20 57 51 4E 45 24 0000054C'010E0000' 0544   487         .ASCID  \$ENQW did not queue a request for a resource which was in\-
75 65 75 71 20 74 6F 6E 20 64 69 64  0552
20 74 73 65 75 71 65 72 20 61 20 65  055E
72 75 6F 73 65 72 20 61 20 72 6F 66  056A
73 61 77 20 68 63 69 68 77 20 65 63  0576
                           6E 69 20  0582
75 6F 68 74 09 2F 21 2C 65 73 75 20  0585   488            \ use,!/        though LCK$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   489
                                     05BA   490 PAR_ERRMSG:
6F 20 51 4E 45 24 000005C2'010E0000' 05BA   491         .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   492
                                     05F6   493 VICTIM_ERRMSG:
20 67 6E 6F 72 57 000005FE'010E0000' 05F6   494         .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   495            \!/     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   496
                                     0651   497 DEQALL_ERRMSG:
6F 20 51 45 44 24 00000659'010E0000' 0651   498         .ASCID  \$DEQ of all locks returned !XL instead of SS$_NORMAL.\
20 73 6B 63 6F 6C 20 6C 6C 61    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    499
                                                068E    500 DEADLK_ERRMSG:
6F 6C 64 61 65 44 00000696'010E0000'            068E    501         .ASCID  \Deadlock was not detected/resolved in !UL seconds.\
64 20 74 6F 6E 20 73 61 77 20 68 63             069C
6F 73 65 72 2F 64 65 74 63 65 74 65             06A8
20 4F 55 21 20 6E 69 20 64 65 76 6C             06B4
                  2E 73 64 6E 6F 63 65 73       06C0
                                                06C8    502
                                                06C8    503 NODLOCK_ERRMSG:
61 63 6F 6C 6C 41 000006D0'010E0000'            06C8    504         .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 20 63 63 73 20 65 63 72                06EE
            65 74 69 70 73 65 20 64             06FA
20 5B 63 6F 6C 64 61 65 64 09 0A 0D             0703    505                 <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    506
                                                0728    507 CASTPAR_ERRMSG:
6F 6C 70 6D 6F 43 00000730'010E0000'            0728    508         .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    509
                                                0760    510 NOTRMB_ERRMSG:
20 72 6F 72 72 45 00000768'010E0000'            0760    511         .ASCID  /Error reading termination mailbox from cooperating process./
6D 72 65 74 20 67 6E 69 6D 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    512
                                                07A3    513 MSGTYP_ERRMSG:
6F 61 73 73 65 4D 000007AB'010E0000'            07A3    514         .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    515
                                                07E6    516 TERMBX_ERRMSG:
6E 69 6D 72 65 54 000007EE'010E0000'            07E6    517         .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    518                 \!/     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    519
                                                0852    520 DETPRC_ERRMSG:
20 72 6F 72 72 45 0000085A'010E0000'            0852    521         .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    522
                                      0898    523 FILE:                                   ; Fills in RMS_ERR_STRING
       65 6C 69 66 000008A0'010E0000' 0898    524         .ASCID  /file/
                                      08A4    525 RECORD:                                 ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 000008AC'010E0000' 08A4    526         .ASCID  /record/
                                      08B2    527 RMS_ERR_STRING:                         ; Announces an RMS error
41 21 20 53 4D 52 000008BA'010E0000' 08B2    528         .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    529
                                      08D3    530 RMS_ERRMSG:
72 65 20 53 4D 52 000008DB'010E0000' 08D3    531         .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    532
                                      090C    533 COPY_LOG_MSG:                           ; Introduces subprocess log file
6F 20 79 70 6F 43 00000914'010E0000' 090C    534         .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    535
                                      093D    536 FORCEX_MSG:
73 65 63 6F 72 50 00000945'010E0000' 093D    537         .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    538
                                      0969    539 DUMP_MSG:
73 65 63 6F 72 50 00000971'010E0000' 0969    540         .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    541                 \!/     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    542
                                      09DA    543 NONE:                                   ; !AD string for $FAO
                   65 6E 6F 6E        09DA    544         .ASCII  /none/
                      00000004        09DE    545 NONE_LENGTH = .-NONE
                                      09DE    546
                                      09DE    547 TEST_CODES:                             ; List of SS we perform
                      000009EA'       09DE    548         .ADDRESS ENQ_CODE               ; Ordering and content are dependent...
                      000009EF'       09E2    549         .ADDRESS ENQW_CODE              ; ...on the definitions in the...
                      000009F5'       09E6    550         .ADDRESS DEQ_CODE               ; ...Equated Symbols section
                                      09EA    551 ENQ_CODE:
                51 4E 45 24 00'       09EA    552         .ASCIC  /$ENQ/
                               04     09EA
                                      09EF    553 ENQW_CODE:
             57 51 4E 45 24 00'       09EF    554         .ASCIC  /$ENQW/
```

```
                 05  09EF
                     09F5    555 DEQ_CODE:
51 45 44 24 00'      09F5    556         .ASCIC  /$DEQ/
                 04  09F5
                     09FA    557
                     09FA    558 LOCK_MODES:                         ; List of $ENQ lock modes
        00000A16'    09FA    559         .ADDRESS NLMODE_CODE        ; Ordering and content are dependent...
        00000A19'    09FE    560         .ADDRESS CRMODE_CODE        ; ...on the definitions supplied...
        00000A1C'    0A02    561         .ADDRESS CWMODE_CODE        ; ...by the $LCKDEF macro
        00000A1F'    0A06    562         .ADDRESS PRMODE_CODE
        00000A22'    0A0A    563         .ADDRESS PWMODE_CODE
        00000A25'    0A0E    564         .ADDRESS EXMODE_CODE
        00000A28'    0A12    565         .ADDRESS NOMODE_CODE        ; This one is a dummy for $DEQ
                     0A16    566 NLMODE_CODE:
           4C 4E 00' 0A16    567         .ASCIC  /NL/
                 02  0A16
                     0A19    568 CRMODE_CODE:
           52 43 00' 0A19    569         .ASCIC  /CR/
                 02  0A19
                     0A1C    570 CWMODE_CODE:
           57 43 00' 0A1C    571         .ASCIC  /CW/
                 02  0A1C
                     0A1F    572 PRMODE_CODE:
           52 50 00' 0A1F    573         .ASCIC  /PR/
                 02  0A1F
                     0A22    574 PWMODE_CODE:
           57 50 00' 0A22    575         .ASCIC  /PW/
                 02  0A22
                     0A25    576 EXMODE_CODE:
           58 45 00' 0A25    577         .ASCIC  /EX/
                 02  0A25
                     0A28    578 NOMODE_CODE:
              6F 6E 00' 0A28    579         .ASCIC  /no/
                 02  0A28
                     0A2B    580
                     0A2B    581 GETSYI_ITMLST:                      ; $GETSYI item list for...
              1067 0006  0A2B    582         .WORD   6,SYI$_SCSNODE      ; ...my node's name in a cluster
    00000000'0000009A'   0A2F    583         .ADDRESS SCSNODE,0
              00000000   0A37    584         .LONG   0
                     0A3B    585
                     0A3B    586 ; Set up data structures so that mailboxes which serve as SYS$INPUT to
                     0A3B    587 ; the processes we create are created in our group logical name table.
                     0A3B    588
                     0A3B    589 LNMPRCDIR:                          ; Table name to force mbx logicals...
52 50 24 4D 4E 4C 00000A43'010E0000'  0A3B    590         .ASCID  /LNM$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    591
                     0A58    592 LNMTMPMBX:                          ; Logical name which tells $CREMBX...
45 54 24 4D 4E 4C 00000A60'010E0000'  0A58    593         .ASCID  /LNM$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    594
                     0A75    595 LNMITMLST:                          ; $CRELNM ITMLST naming where mbx...
              0002 0010' 0A75    596         .WORD   LNMGRPLEN,LNM$_STRING   ; ...logical name is to be defined
    00000000'000000D2'   0A79    597         .ADDRESS LNMGRPNUM,0
              00000000   0A81    598         .LONG   0
```

```
                        0A85   600          .SBTTL  Read/Write Data
                    00000000   601          .PSECT  RWDATA,WRT,NOEXE,PAGE
                        0000   602
                        0000   603 DRIVEN_DESC:                                ; Name of driven process. Driven procs...
                   0000 000B'  0000   604          .WORD   TEST_NAME_LEN+1,0     ; ...always have qualifier char at end
                    00000011'  0004   605          .ADDRESS TEST_NAME_I
                        0008   606 TEST_NAME_D:                                ; Name of the test - .ASCID version
                   0000 000A'  0008   607          .WORD   TEST_NAME_LEN,0
                    00000011'  000C   608          .ADDRESS TEST_NAME_I
                        0010   609 ; THESE NEXT ITEMS MUST REMAIN CONTIGUOUS!
                        0010   610 TEST_NAME_C:                                ; Name of the test - .ASCIC version
                          0A'  0010   611          .BYTE   TEST_NAME_LEN
                        0011   612 TEST_NAME_I:                                ; Name of the test - .ASCII version
 5F 30 30 4B 43 4F 4C 54 45 55  0011   613          .ASCII  /UETLOCK00_/
                    0000000A   001B   614 TEST_NAME_LEN = .-TEST_NAME_I        ; Length in chars of our test name
                        001B   615 PROC_QUALIFIER:                             ; Driven process qualifier appended...
                    0000001C   001B   616          .BLKB   1                   ; ...to test name
                        001C   617 ; End of contiguous items
                        001C   618
                        001C   619 MBXCHANS:                                   ; Mailbox channels to communicate with detac
                    00000020   001C   620          .BLKW   PROC_COUNT
                        0020   621
                        0020   622 PROCIDS:                                    ; Detached processes id
                    00000028   0020   623          .BLKL   PROC_COUNT
                    00000000   0028   624          .LONG   0                   ; End of the proc id table
                        002C   625
                        002C   626 LOG_FILE_DESC:                              ; Skeleton name for driven proc log files
                   0000 000B'  002C   627          .WORD   LOG_FILE_LEN,0
                    00000034'  0030   628          .ADDRESS LOG_FILE_QUAL
                        0034   629 ; THE FOLLOWING ITEMS MUST REMAIN CONTIGUOUS!
                        0034   630 LOG_FILE_QUAL:                              ; Character to distinguish between procs
                          00   0034   631          .BYTE   0
 47 4F 4C 2E 54 53 45 54 4B 4C  0035   632          .ASCII  /LKTEST.LOG/
                    0000000B   003F   633 LOG_FILE_LEN = .-LOG_FILE_QUAL
                        003F   634 ; End of items which must remain contiguous.
                        003F   635
                        003F   636 ENQLST:                                     ; ENQ(W) QIO arguments list
                               637          $ENQ    RESNAM = RESR_DESC
                        006F   638
                        006F   639 DEQLST:                                     ; DEQ QIO argument list
                               640          $DEQ
                        0083   641
                        0083   642 TST_COMMAND:                                ; Test command from the test table
            00000000 00000000  0083   643          .LONG   0,0
                        008B   644
                        008B   645 TSTFLG:                                     ; Tells actions after calling sys svc
                        0000   008B   646          .WORD   0
                        008D   647
                        008D   648 RESR_DESC:                                  ; Resource name descriptor
                    00000011'  008D   649          .LONG   RESR_LEN
                    00000095'  0091   650          .ADDRESS .+4                ; Note:  We assume all the following...
                 24 50 54 45 55  0095   651          .ASCII  'UETP$'             ; (Facility name, for convention's sake)
                        009A   652 SCSNODE:                                    ; (Node name, so unique within cluster)
                    000000A0   009A   653          .BLKB   6
                          5F   00A0   654          .ASCII  "_"                 ; (Separator)
                        00A1   655 RESR:                                       ; (Resource name)
              4B 43 4F 4C 3F   00A1   656          .ASCII  /?LOCK/             ; ...are contiguous
```

```
                    00000011  00A6   657 RESR_LEN = .-RESR_DESC-8
                              00A6   658
                              00A6   659 DLOCK_TIME:                             ; Delta time for $SETIMR to prevent...
                    000000AA  00A6   660         .BLKL   1                      ; ...deadlock hangs
                    FFFFFFFF  00AA   661         .LONG   -1
                              00AE   662
                              00AE   663 GETUIC:                                 ; Get UIC
                        0004  00AE   664         .WORD   4                      ; Length of recieve buffer
                        0304  00B0   665         .WORD   JPI$_UIC               ; Request UIC
                    000000CA' 00B2   666         .LONG   LK_UIC                 ; Buffer address
                    00000000  00B6   667         .LONG   0                      ; No return length
                        0004  00BA   668         .WORD   4                      ; Get our base priority
                        0309  00BC   669         .WORD   JPI$_PRIB
                    000000CE' 00BE   670         .ADDRESS BASE_PRI
                    00000000  00C2   671         .LONG   0
                    00000000  00C6   672         .LONG   0                      ; End of list
                              00CA   673
                              00CA   674 LK_UIC:                                 ; Put UIC here
                    00000000  00CA   675         .LONG   0
                              00CE   676
                              00CE   677 BASE_PRI:                               ; Parent process' base priority
                    00000000  00CE   678         .LONG   0
                              00D2   679
                              00D2   680 LNMGRPNUM:                              ; Equivalence name giving the table...
5F 50 55 4F 52 47 24 4D 4E 4C 00D2   681         .ASCII  /LNM$GROUP_/           ; ...name in which mbx logical names...
                    0000000A  00DC   682         LNMGRP = .-LNMGRPNUM
                20'20'20'20'20'20'20' 00DC   683         .BYTE   ^A/ /[6]
                    00000010  00E2   684         LNMGRPLEN = .-LNMGRPNUM        ; ...are to be put
                              00E2   685
                              00E2   686 LNMGRPNIL:                              ; Descriptor to convert...
                  0000 0006   00E2   687         .WORD   LNMGRPLEN-LNMGRP,0
                   000000DC'  00E6   688         .ADDRESS LNMGRPNUM+LNMGRP      ; ...UIC to text
                              00EA   689
                              00EA   690 DIBBUF_DESC:                            ; Device information buffer descriptor
                  0000 0074   00EA   691         .WORD   DIB$K_LENGTH,0
                   000000F2'  00EE   692         .ADDRESS DIBBUF
                              00F2   693 DIBBUF:                                 ; Device information block
                    00000166  00F2   694         .BLKB   DIB$K_LENGTH
                              0166   695
                              0166   696 MBX_DESC:                               ; Mailbox logical name descriptor
                    00000006' 0166   697         .LONG   MBXNAM_LEN
                    0000016E' 016A   698         .ADDRESS MBXNAM
                              016E   699 MBXNAM:                                 ; Mailbox logical name
58 42 4D 4B 4C 3F             016E   700         .ASCII  /?LKMBX/
                    00000006  0174   701 MBXNAM_LEN = .-MBXNAM
                              0174   702
                              0174   703 MBX_IOSB:                              ; Mailbox QIO IO status block
00000000 00000000             0174   704         .LONG   0,0
                              017C   705
                              017C   706 EX_MBXCHAN:                            ; Termination mailbox channel
                        0000  017C   707         .WORD   0
                              017E   708 EXIT_MSG:                              ; Buffer for termination mailbox msg
                    000001D2  017E   709         .BLKB   ACC$K_TERMLEN
                              01D2   710 EX_PROC_CNT:                           ; Exit process count
                    00000002  01D2   711         .LONG   PROC_COUNT
                              01D6   712
                              01D6   713 TTCHAN:                                ; Terminal channel
```

```
                    00000000 01D6   714          .LONG   0
                             01DA   715
                             01DA   716 TYPE:
                    00000003 01DA   717          .LONG   3                      ; System service type
                             01DE   718
                             01DE   719 GLBFLG:
                        0000 01DE   720          .WORD   0                      ; Holds flags which remain thru test
                             01E0   721
                             01E0   722 LOCFLG:
                          00 01E0   723          .BYTE   0                      ; Local flag - flags for one TST_TABLE event
                             01E1   724
                             01E1   725 EF_STATE:
                    00000000 01E1   726          .LONG   0                      ; Event flag longword
                             01E5   727
                             01E5   728 SAVED_VAL:
00000000 00000000            01E5   729          .LONG   0,0                    ; Saved value of first quad of value block
                             01ED   730 LKID_ADDR:
                    00000000 01ED   731          .LONG   0                      ; Lock id address
                             01F1   732 LKSB_ADDR:
                    00000000 01F1   733          .LONG   0                      ; Lock status block address
                             01F5   734 LKSBS:
                    00000465 01F5   735          .BLKQ   <3*26>                 ; Buffer for LKSB's
                             0465   736
                             0465   737 EXIT_DESC:
                    00000000 0465   738          .LONG   0                      ; Exit handler descrip
                    00000CAA' 0469   739          .ADDRESS EXIT_HANDLER
                    00000001 046D   740          .LONG   1
                    00000511' 0471   741          .ADDRESS STATUS
                             0475   742
                             0475   743 MSG_BLOCK:                             ; Message block for GETMSG
                    00000479 0475   744          .BLKB   4
                             0479   745
                             0479   746 FAO_BUF:                               ; FAO output string descriptor
                        0000 0084   747          .WORD   TEXT_BUFFER,0
                     00000489' 047D   748          .ADDRESS BUFFER
                             0481   749
                             0481   750 BUFFER_PTR:                           ; Tex  buffer descriptor
                        0000 0084   751          .WORD   TEXT_BUFFER,0
                     00000489' 0485   752          .ADDRESS BUFFER
                             0489   753
                             0489   754 BUFFER:                               ; FAO output and other misc. buffer
                     0000050D 0489   755          .BLKB   TEXT_BUFFER
                             050D   756
                             050D   757 ARG_COUNT:                           ; Argument count
                    00000000 050D   758          .LONG   0
                             0511   759
                             0511   760 STATUS:                              ; Final status code
                    00000000 0511   761          .LONG   0
                             0515   762
                             0515   763 QUAD_STATUS:                         ; IO status block for misc. sys. svcs.
00000000 00000000            0515   764          .QUAD   0
                             051D   765
                             051D   766 LOG_MSGPTR:                          ; $FAO arg for $PUTMSG when copying...
                        0000 0000   767          .WORD   0,0                    ; ...the driven proc log file...
                     00000525' 0521   768          .ADDRESS RMS_BUFFER            ; ...to our SYS$OUTPUT
                             0525   769
                             0525   770 RMS_BUFFER:                           ; Log file record buffer
```

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

```
05A9    773          .SBTTL  RMS-32 Data Structures
05A9    774
05A9    775          .ALIGN  LONG
05AC    776
05AC    777 LOG_FAB:
05AC    778          $FAB -
05AC    779          FNA = LOG_FILE_QUAL,-
05AC    780          FNS = LOG_FILE_LEN,-
05AC    781          ORG = <SEQ>,-
05AC    782          MRS = TEXT_BUFFER
05FC    783
05FC    784 LOG_RAB:
05FC    785          $RAB -
05FC    786          FAB = LOG_FAB,-
05FC    787          UBF = RMS_BUFFER+INDENT,-
05FC    788          USZ = TEXT_BUFFER
0640    789
0640    790 INPUT_FAB:                                  ; Reads SYS$INPUT
0640    791          $FAB -
0640    792          FNM = <SYS$INPUT:>
0690    793
0690    794 INPUT_RAB:                                  ; Reads SYS$INPUT
0690    795          $RAB -
0690    796          FAB = INPUT_FAB,-
0690    797          UBF = TST_COMMAND,-
0690    798          USZ = COMMAND_SIZE
```

```
                        06D4    800          .SBTTL  TESTLOCK - Initialization
                    00000000    801          .PSECT  TESTLOCK,EXE,NOWRT,PAGE
                        0000    802          .DEFAULT DISPLACEMENT,WORD
                        0000    803
                        0000    804 ;+
                        0000    805 ;       This module serves as both the controlling (driver) and controlled
                        0000    806 ;       (driven) process of the Lock Manager test.  Initialize and set up
                        0000    807 ;       those things which are common to both processes - overhead, determining
                        0000    808 ;       our function, timers and interprocess communication.
                        0000    809 ;-
                    0000 0000    810 UETLOCK00::.ENTRY  UETLOCK00,^M<>                   ; Entry mask
                        0002    811
                6D  0A2F'CF  DE 0002    812          MOVAL   SSERROR,(FP)                ; Declare exception handler
                        0007    813          $SETSFM_S ENBFLG = #1               ; Enable system service failure mode
                        0010    814          $DCLEXH_S DESBLK = EXIT_DESC        ; Declare exit handler
                        001B    815 ;
                        001B    816 ; Determine if we are the master (driver) process or one of the slave (driven)
                        001B    817 ; processes.  If we are a driven process, determine which one.  Use that info
                        001B    818 ; to form our process name and announce us to the world.
                        001B    819 ;
                        001B    820          $OPEN   FAB = INPUT_FAB             ; See what sort of thing SYS$INPUT is
                     76 50  E9  0026    821          BLBC    R0,10$                  ; If we've no SYS$INPUT, we are driver
                        14  E1  0029    822          BBC     #DEV$V_MBX,-            ; If SYS$INPUT is not a mailbox...
                 70 0680'CF     002B    823                  INPUT_FAB+FAB$L_DEV,10$ ; ...we are driver
                        002F    824          $CONNECT RAB = INPUT_RAB,-          ; Check further to see if we're driven
                        002F    825                   ERR = RMS_ERROR
                        003E    826          $GET    RAB = INPUT_RAB,-
                        003E    827                  ERR = RMS_ERROR
                    02DC'CF  29 004D    828          CMPC3   SUBPROC_STRING,-            ; See if mailbox message...
                    02E4'CF     0051    829                  SUBPROC_STRING+8,-      ; ...is the one we expect...
                    0083'CF     0054    830                  TST_COMMAND             ; ...to tell us which process we are
                        46  12  0057    831          BNEQ    10$                     ; BR if no match - we're driver
            001B'CF     63  90  0059    832          MOVB    (R3),PROC_QUALIFIER     ; We're driven proc - copy which one
                    0008'CF B6  005E    833          INCW    TEST_NAME_D             ; Attach it to our name...
                    0010'CF 96  0062    834          INCB    TEST_NAME_C             ; ...in all places we need it
                01DE'CF     01  88  0066    835          BISB2   #DRIVEN_M,GLBFLG        ; Remember that we're a driven process
                            006B    836                                              ; We do only what we're told to do
            02A9'CF     02  63  3A  006B    837          LOCC    (R3),#PROC_COUNT,PROCS  ; Have we a legal process name?
                        34  12  0071    838          BNEQ    20$                     ; BR if so - we're all set up
                            0073    839          $FAO_S  CTRSTR = ILL_PROC_NAME,- ; Bad name - quit
                            0073    840                  OUTLEN = BUFFER_PTR,-
                            0073    841                  OUTBUF = FAO_BUF,-
                            0073    842                  P1     = #1,-
                            0073    843                  P2     = #PROC_QUALIFIER
                    0481'CF     DF  008E    844          PUSHAL  BUFFER_PTR
                            01  DD  0092    845          PUSHL   #1
                 00741132 8F    DD  0094    846          PUSHL   #UETP$_TEXT!STS$K_ERROR
                            03  DD  009A    847          PUSHL   #3
                        0B63    31  009C    848          BRW     ERROR_EXIT
                            009F    849 10$:
                    0008'CF B7  009F    850          DECW    TEST_NAME_D             ; Fix up our name...
                    0010'CF 97  00A3    851          DECB    TEST_NAME_C             ; ...wherever it is needed
                            00A7    852          ;BICB2  #DRIVEN_M,GLBFLG        ; Remember that we're a driver process
                            00A7    853                                          ; We tell other processes what to do
                            00A7    854 20$:
                            00A7    855          $SETPRN_S PRCNAM = TEST_NAME_D  ; Set our process name
                            00B2    856
```

```
                022C'CF    DF   00B2    857          PUSHAL  BEGUN_ADDR
                0010'CF    DF   00B6    858          PUSHAL  TEST_NAME_C
                      02   DD   00BA    859          PUSHL   #2                              ; Argument count
             007480D9 8F   DD   00BC    860          PUSHL   #UETP$_SATSMS!STS$K_SUCCESS ; Set the begin message code
00000000'GF      04   FB   00C2    861          CALLS   #4,G^LIB$SIGNAL                 ; Print the UETP begin time message
          01DE'CF    04   A8   00C9    862          BISW2   #BEGIN_MSGM,GLBFLG              ; Indicate that sentinel was given
                          00CE    863 ;
                          00CE    864 ; Set timer to catch hangs waiting for another process or for any hangs we
                          00CE    865 ; may encounter ourself.
                          00CE    866 ;
                          00CE    867          $SETIMR_S DAYTIM = THREE_MIN,-  ; Set 3 minute timer
                          00CE    868                    ASTADR = TIME_OUT,-
                          00CE    869                    REQIDT = #1
                          00E1    870 ;
                          00E1    871 ; Enable control C handler if run from terminal.
                          00E1    872 ;
                      02  E1   00E1    873          BBC     S^#DEV$V_TRM,-                  ; BR if SYS$INPUT is NOT a terminal
                66 0680'CF        00E3    874                  INPUT_FAB+FAB$L_DEV,50$
                          00E7    875          $GETDVI_S DEVNAM = SYS$INPUT,-      ; Get the name of...
                          00E7    876                    EFN    = #SS_SYNCH_EFN,- ; ...device which may abort test
                          00E7    877                    ITMLST = INPUT_ITMLST,-
                          00E7    878                    IOSB   = QUAD_STATUS
                45 0515'CF    E9   0103    879          BLBC    QUAD_STATUS,50$                 ; Avoid CTRL/C handler if any error
                          0108    880          $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C AST handling
                          0108    881                    CHAN   = TTCHAN
                          0119    882          $QIOW_S CHAN     = TTCHAN,-     ; Enable CTRL/C AST's...
                          0119    883                  FUNC     = #IO$_SETMODE!IO$M_CTRLCAST,-
                          0119    884                  P1       = CCASTHAND
                    0008'CF    DF   013A    885          PUSHAL  TEST_NAME_D                     ; ...and tell the user...
                          01   DD   013E    886          PUSHL   #1                              ; ...
                 0074832B 8F   DD   0140    887          PUSHL   #UETP$_ABORTC!STS$K_SUCCESS ; ...how to abort gracefully...
00000000'GF      03   FB   0146    888          CALLS   #3,G^LIB$SIGNAL                 ; ...
                          014D    889 ;
                          014D    890 ; Create and associate with a common event flag block.  This will be used to
                          014D    891 ; signal that a particular operation of a driven process has been completed.
                          014D    892 ;
                          014D    893 50$:
                          014D    894          $ASCEFC_S EFN = #DONE_CEF,-      ; Create and assoc common EF cluster
                          014D    895                    NAME = LK_CEF_DESC
                          0162    896 ;
                          0162    897 ; The logical name MODE is available to knowledgeable users to invoke special
                          0162    898 ; actions in this test.
                          0162    899 ;
                          0162    900          $TRNLOG_S LOGNAM = MODE,-       ; If MODE translates...
                          0162    901                    RSLLEN = BUFFER_PTR,-
                          0162    902                    RSLBUF = FAO_BUF
0489'CF    504D5544 8F   D1   017B    903          CMPL    #^A/DUMP/,BUFFER                ; ...to "DUMP"...
                      05   12   0184    904          BNEQ    60$
             01DE'CF    02   A8   0186    905          BISW2   #DUMP_M,GLBFLG                  ; ...then we type a running log
                          018B    906 60$:
                          018B    907          $GETSYI_S ITMLST = GETSYI_ITMLST ; Get my node's node name
                          01A0    908 ;
                          01A0    909 ; All things common to the driver and driven processes have been done.  Split
                          01A0    910 ; up and get on to the main work of the test.
                          01A0    911 ;
          03 01DE'CF    00   E1   01A0    912          BBC     #DRIVEN_V,GLBFLG,DRIVER ; BR if we are the driver process
                    010F   31   01A6    913          BRW     DRIVEN                          ; We are a driven process
```

```
                        01A9   915          .SBTTL  TESTLOCK - Driver Process
                        01A9   916 ;+
                        01A9   917 ;        This is the controlling process of the lock manager UETP test.  It
                        01A9   918 ;        creates detached, "cooperating" process(es), then, based on the entries
                        01A9   919 ;        in TST_TABLE, either calls Lock Manager system services or sends mail
                        01A9   920 ;        to (one or more) cooperating process(es) to have the process(es) call
                        01A9   921 ;        a Lock Manager system service.  Results are checked.
                        01A9   922 ;-
                        01A9   923 DRIVER:
                        01A9   924          $CREMBX_S CHAN = EX_MBXCHAN,-   ; Create and associate termination mbx
                        01A9   925                    MAXMSG = #ACC$K_TERMLEN,-
                        01A9   926                    BUFQUO = #256
                        01C6   927          $GETCHN_S CHAN = EX_MBXCHAN,-   ; Get channel infomation - we need...
                        01C6   928                    PRIBUF = DIBBUF_DESC  ; ...the mailbox unit number
                        01DC   929          $GETJPI_S ITMLST = GETUIC       ; Get our UIC and our base priority
              02    DD  01F1   930          PUSHL   #2                      ; We'll convert a 2-byte...
     7E 00E2'CF    3C  01F3   931          MOVZWL  LNMGRPNIL,-(SP)         ; ...integer to ASCII...
        00E2'CF    DF  01F8   932          PUSHAL  LNMGRPNIL
        00CC'CF    DF  01FC   933          PUSHAL  LK_UIC+2                ; ...using group number of...
00000000'GF    04    FB  0200   934          CALLS   #4,G^OTS$CVT_L_TO       ; ...the process which runs us
                        0207   935          $CRELNM_S TABNAM = LNMPRCDIR,-  ; Force SYS$INPUT mbx name...
                        0207   936                    LOGNAM = LNMTMPMBX,-  ; ...to appear in...
                        0207   937                    ITMLST = LNMITMLST    ; ...a group logical name table
              00D9    30  021E   938          BSBW    CREATE_PROCS            ; Go subroutine to create detached procs
                        0221   939 ;
                        0221   940 ; Have an outstanding read to the termination mailbox, should any detached
                        0221   941 ; process end
                        0221   942 ;
                        0221   943          $QIO_S  CHAN = EX_MBXCHAN,-     ; Read termination mailbox
                        0221   944                  FUNC   #IO$_READVBLK,-
                        0221   945                  ASTADR = EX_MBX_AST,-
                        0221   946                  IOSB = MBX_IOSB,-
                        0221   947                  P1 = EXIT_MSG,-
                        0221   948                  P2 = #ACC$K_TERMLEN
                        024C   949 ;
                        024C   950 ; For each line in TEST_TABLE, either execute the command directly or send a
                        024C   951 ; message to one of the driven processes.  Wait for the driven process to
                        024C   952 ; complete the command, if necessary.
                        024C   953 ;
     59 0000'CF    DE  024C   954          MOVAL   TEST_TABLE,R9           ; Address of beginning of test table
                        0251   955 GET_COMMAND:
   0083'CF    69    7D  0251   956          MOVQ    (R9),TST_COMMAND        ; Get an entry of the test table
02A8'CF 0083'CF    91  0256   957          CMPB    TST_COMMAND,ALL_PROCS   ; Is it for self?
              3A    13  025D   958          BEQL    20$                     ; BR if yes
                        025F   959
                        025F   960          $CLREF_S EFN = #DONE_CEF        ; Clear CEF which says command is done
              011A    30  026C   961          BSBW    SEND_MSG                ; Send command to the detached process
 0F 0089'CF    04    E1  026F   962          BBC     #DEADLK_V,TST_COMMAND+6,10$ ; BR if not deadlock special case
                        0275   963          $WFLAND_S EFN  = #DONE_CEF,-    ; Deadlock situation has been set up...
                        0275   964                    MASK = #DLMASK        ; ...we wait until it's resolved
                        0284   965 10$:
 19 0089'CF    06    E0  0284   966          BBS     #NOWAIT_V,TST_COMMAND+6,30$ ; BR if no_wait_done_cef bit set
                        028A   967          $WAITFR_S EFN = #DONE_CEF       ; Wait until another process finish
                        0297   968                                          ; executing the test command
              0A    11  0297   969          BRB     30$                     ; We've finished this table entry
                        0299   970 20$:
   0084'CF    03    91  0299   971          CMPB    #ENDTEST,TST_COMMAND+1  ; Is this endtest for driver process?
```

```
                     03   13  029E   972          BEQL    30$                     ; BR if so - we'll really end when...
                                 02A0   973                                      ; ...the test table is finished
                   014F   30  02A0   974          BSBW    EXECUTE                 ; Go execute the test command
                                 02A3   975 30$:
               59  08 A9   DE  02A3   976          MOVAL   COMMAND_SIZE(R9),R9     ; Get address of next test table entry
                     69   D5  02A7   977          TSTL    (R9)                    ; Is it end of the table?
                     03   13  02A9   978          BEQL    WAIT_PRCS               ; BR if yes
                   FFA3   31  02AB   979          BRW     GET_COMMAND             ; Next entry
                                 02AE   980
                                 02AE   981 WAIT_PRCS:
                                 02AE   982          $HIBER_S                    ; Hibernate until created procs exit
                   0021   31  02B5   983          BRW     SUC_EXIT                ; The test is over, let's go home
```

```
                     02B8   985          .SBTTL  TESTLOCK - Driven Process
                     02B8   986 ;+
                     02B8   987 ;         This is the controlled process of the Lock Manager UETP test.  It
                     02B8   988 ;         is created by te driver process, then waits for commands from the
                     02B8   989 ;         driver process to tell it how to call Lock Manager system services.
                     02B8   990 ;         In some cases, results are checked to see if they are what we expect.
                     02B8   991 ;         Note that SYS$INPUT is already defined as the way commands are passed
                     02B8   992 ;         by the driver program.
                     02B8   993 ;-
                     02B8   994 DRIVEN:
                     02B8   995 GET_NEXTMSG:
                     02B8   996           $GET    RAB = INPUT_RAB,-       ; Read a command
                     02B8   997                   ERR = RMS_ERROR
           0128   30 02C7   998           BSBW    EXECUTE                 ; Perform it
                     02CA   999
                     02CA  1000           $SETEF_S EFN = #DONE_CEF        ; Tell the driver process we're done
             DF    11 02D7  1001           BRB     GET_NEXTMSG
```

```
                     02D9  1003          .SBTTL  Clean Up and Exit
                     02D9  1004 ;+
                     02D9  1005 ;       Arrive here from either the driver or the driven process.  Process
                     02D9  1006 ;       exit or failure is the same in either case.
                     02D9  1007 ;-
                     02D9  1008 SUC_EXIT:                                ; Successfully exit
10000000'8F   DO     02D9  1009          MOVL    #SS$_NORMAL!STS$M_INHIB_MSG,-
    0511'CF          02DF  1010                  STATUS
                     02E2  1011          $EXIT_S STATUS                  ; Exit with status
                     02ED  1012
                     02ED  1013 ; To use FAIL_OUT, push to the stack the address of an error message and
                     02ED  1014 ; BRW FAIL_OUT.
                     02ED  1015 FAIL_OUT:                                ; Failure exit
          01   DD    02ED  1016          PUSHL   #1                      ; Arg count
00741132 8F   DD    02EF  1017          PUSHL   #UETP$_TEXT!STS$K_ERROR ; UETP error code
          03   DD    02F5  1018          PUSHL   #3                      ; Argument count
        0908   31    02F7  1019          BRW     ERROR_EXIT              ; Error exit
```

```
                        02FA  1021          .SBTTL  Create Detached Processes and Mailboxes
                        02FA  1022 ;++
                        02FA  1023 ; FUNCTIONAL DESCRIPTION:
                        02FA  1024 ;       Create detached processes and associated mailboxes routine.  Such
                        02FA  1025 ;       processes are called driven processes because they only perform what
                        02FA  1026 ;       this, the driver process, tells them to do.  Detached processes with
                        02FA  1027 ;       the same UIC are created according to the process indicators (Q, R,
                        02FA  1028 ;       etc.) stored at addresses beginning PROCS.  The created processes
                        02FA  1029 ;       will run the same image as this, but will be able to distinguish itself
                        02FA  1030 ;       from this process by the content of the mailbox which is its SYS$INPUT.
                        02FA  1031 ;       There are mailboxes associated with each created process as
                        02FA  1032 ;       communication channels.
                        02FA  1033 ;
                        02FA  1034 ; CALLING SEQUENCE:
                        02FA  1035 ;       BSBW    CREATE_PROCS
                        02FA  1036 ;
                        02FA  1037 ; INPUT PARAMETERS:
                        02FA  1038 ;       NONE
                        02FA  1039 ;
                        02FA  1040 ; IMPLICIT INPUTS:
                        02FA  1041 ;       Process name table, PROCS
                        02FA  1042 ;
                        02FA  1043 ; OUTPUT PARAMETERS:
                        02FA  1044 ;
                        02FA  1045 ;       NONE
                        02FA  1046 ; IMPLICIT OUTPUTS:
                        02FA  1047 ;       Exit if error
                        02FA  1048 ;
                        02FA  1049 ; COMPLETION CODES:
                        02FA  1050 ;       NONE
                        02FA  1051 ;
                        02FA  1052 ; SIDE EFFECTS:
                        02FA  1053 ;       Creates detached, cooperating processes
                        02FA  1054 ;--
                        02FA  1055 CREATE_PROCS:
      02DC'CF    28     02FA  1056          MOVC3   SUBPROC_STRING,-        ; Initialize mailbox message...
      02E4'CF            02FE  1057                  SUBPROC_STRING+8,-      ; ...that will tell driven proc...
      0083'CF            0301  1058                  TST_COMMAND             ; ...which one it is
      56    53    D0     0304  1059          MOVL    R3,R6                   ; Save this pointer into TST_COMMAND
52    02A9'CF    DE     0307  1060          MOVAL   PROCS,R2                ; Addr of process name indicators
53    001C'CF    DE     030C  1061          MOVAL   MBXCHANS,R3             ; Addr of associated mailbox channel
54    0020'CF    DE     0311  1062          MOVAL   PROCIDS,R4              ; Addr of process ids
         55      D4     0316  1063          CLRL    R5                      ; Indexes into process qualifier table
                        0318  1064 10$:
001B'CF   6245    90     0318  1065          MOVB    (R2)[R5],PROC_QUALIFIER ; Set up process name
016E'CF   6245    90     031E  1066          MOVB    (R2)[R5],MBXNAM         ; Set up assoc mailbox logical name
0034'CF   6245    90     0324  1067          MOVB    (R2)[R5],LOG_FILE_QUAL  ; Set up log file name
     66   6245    90     032A  1068          MOVB    (R2)[R5],(R6)           ; Set up command to say "driven proc"
                        032E  1069
                        032E  1070          $CREMBX_S CHAN = (R3)[R5],-     ; Create mailbox and assign channel
                        032E  1071                    MAXMSG = #COMMAND_SIZE,- ; Maximum message length in bytes
                        032E  1072                    BUFQUO = #512,-       ; Buffer quota
                        032E  1073                    LOGNAM = MBX_DESC     ; Logical name "Q"LKMBX,"R"LKMBX, etc.
                        034A  1074
                        034A  1075          $CREPRC_S PIDADR = (R4)[R5],-   ; Create detached process
                        034A  1076                    IMAGE = LKTEST_DESC,- ; Image
                        034A  1077                    OUTPUT = LOG_FILE_DESC,- ; SYS$OUTPUT
```

```
                          034A  1078                    INPUT  = MBX_DESC,-   ; SYS$INPUT
                          034A  1079                    PRCNAM = DRIVEN_DESC,- ; Proc name
                          034A  1080                    UIC = LK_U.C,-        ; Same UIC as this controller
                          034A  1081                    BASPRI = BASE_PRI,-   ; Same base priority
                          034A  1082                    MBXUNT = DIBB0F+DIB$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  1090          .SBTTL  Determine Which Driven Process Gets Command
                     0389  1091 ;++
                     0389  1092 ; FUNCTIONAL DESCRIPTION:
                     0389  1093 ;       Send message to cooperative processes routine
                     0389  1094 ;
                     0389  1095 ; CALLING SEQUENCE:
                     0389  1096 ;       BSBW    SEND_MSG
                     0389  1097 ;
                     0389  1098 ; INPUT PARAMETERS:
                     0389  1099 ;       NONE
                     0389  1100 ;
                     0389  1101 ; IMPLICIT INPUTS:
                     0389  1102 ;       Message is in TST_COMMAND
                     0389  1103 ;
                     0389  1104 ; OUTPUT PARAMETERS:
                     0389  1105 ;       NONE
                     0389  1106 ;
                     0389  1107 ; IMPLICIT OUTPUTS:
                     0389  1108 ;       Mailbox message sent to the designated process
                     0389  1109 ;
                     0389  1110 ; COMPLETION CODES:
                     0389  1111 ;       In STATUS if error
                     0389  1112 ;
                     0389  1113 ; SIDE EFFECTS:
                     0389  1114 ;       NONE
                     0389  1115 ;--
                     0389  1116
                     0389  1117 SEND_MSG:
              55  D4 0389  1118          CLRL    R5                      ; Index into driven processes
                     038B  1119 10$:                                     ; Which process?
02A9'CF45  0083'CF  91 038B  1120          CMPB    TST_COMMAND,PROCS[R5]   ; Compare process indicator
                 08  13 0393  1121          BEQL    20$                     ; BR if equal
           F2 55    02  F2 0395  1122          AOBLSS  #PROC_COUNT,R5,10$      ; Try again if more process indicators
              0325'CF  DF 0399  1123          PUSHAL  ERR_IN_TABLE            ; Error message
                 FF4D  31 039D  1124          BRW     FAIL_OUT                ; Failure out
                     03A0  1125 20$:
      7E   001C'CF45  3C 03A0  1126          MOVZWL  MBXCHANS[R5],-(SP)
        03AC'CF    01  FB 03A6  1127          CALLS   #1,MBX_QIO              ; Send the message
                     05  03AB  1128          RSB
```

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

```
                        03F2  1180          .SBTTL  Execute Test Commands in the Test Table
                        03F2  1181 ;++
                        03F2  1182 ; FUNCTIONAL DESCRIPTION:
                        03F2  1183 ;       Decode the test command and set up the service parameters accordingly,
                        03F2  1184 ;       Execute the service (ENQ,ENQW,DEQ) requested
                        03F2  1185 ;
                        03F2  1186 ; CALLING SEQUENCE:
                        03F2  1187 ;       BSBW    EXECUTE
                        03F2  1188 ;
                        03F2  1189 ; INPUT PARAMETERS:
                        03F2  1190 ;       NONE
                        03F2  1191 ;
                        03F2  1192 ; IMPLICIT INPUTS:
                        03F2  1193 ;       Test command stored in TST_COMMAND quadword
                        03F2  1194 ;
                        03F2  1195 ; OUTPUT PARAMETERS:
                        03F2  1196 ;       NONE
                        03F2  1197 ;
                        03F2  1198 ; IMPLICIT OUTPUTS:
                        03F2  1199 ;       Address of LKSB in R3
                        03F2  1200 ;
                        03F2  1201 ; COMPLETION CODES:
                        03F2  1202 ;       In STATUS if error
                        03F2  1203 ;
                        03F2  1204 ; SIDE EFFECTS:
                        03F2  1205 ;       Exit if error,
                        03F2  1206 ;       BSBW CHECK_UP if ENQW to check correctness
                        03F2  1207 ;       An AST may be generated for some locks
                        03F2  1208 ;--
                        03F2  1209
                        03F2  1210 EXECUTE:
      01E0'CF    94     03F2  1211          CLRB    LOCFLG                  ; Clear all local flags
 56   0083'CF    DE     03F6  1212          MOVAL   TST_COMMAND,R6          ; Addr of the test command
           86    95     03FB  1213          TSTB    (R6)+                   ; Skip the process indicator
      01DA'CF    86     9A    03FD  1214          MOVZBL  (R6)+,TYPE              ; Get type of service, ENQ, ENQW or DEQ
      01DA'CF    03     D1    0402  1215          CMPL    #ENDTEST,TYPE           ; Is it end of test for driven process?
           03    12     0407  1216          BNEQ    10$                     ; BR if not
         FECD    31     0409  1217          BRW     SUC_EXIT                ; We're done if it is
                        040C  1218 10$:
05 01DE'CF    01    E1     040C  1219          BBC     #DUMP_V,GLBFLG,20$      ; BR if we don't dump every command
   052F'CF    00    FB     0412  1220          CALLS   #0,DUMP_COMMAND         ; Type every command to SYS$OUTPUT
                        0417  1221 20$:
 57   003F'CF    DE     0417  1222          MOVAL   ENQLST,R7               ; Assume ENQ(W), Addr of ENQ arg list
        04 A7    D4     041C  1223          CLRL    ENQ$_EFN(R7)            ; Assume EFN = 0
     08 A7    86     9A    041F  1224          MOVZBL  (R6)+,ENQ$_LKMODE(R7)   ; Get lock mode
        52    86     9A    0423  1225          MOVZBL  (R6)+,R2                ; Get resource indicator
      00A1'CF    52     90    0426  1226          MOVB    R2,RESR                 ; Set resource name
52    00000041    8F     C2    042B  1227          SUBL2   #^A/A/,R2               ; Index of associated LKSB
        52    18     C4    0432  1228          MULL2   #LKSB_SIZE,R2           ; Displacement of the LKSB from LKSBS
 53   01F5'C2    DE     0435  1229          MOVAL   LKSBS(R2),R3            ; Address of associated LKSB
      01F1'CF    53     D0    043A  1230          MOVL    R3,LKSB_ADDR            ; Store the addr of LKSB
01ED'CF    04 A3    DE     043F  1231          MOVAL   4(R3),LKID_ADDR         ; Address of the LOCK ID
 02   01DA'CF    D1     0445  1232          CMPL    TYPE,#DEQ               ; Is it DEQ?
           03    12     044A  1233          BNEQ    30$                     ; BR if not
         00CD    31     044C  1234          BRW     DEQS                    ; Yes, go dequeue
                        044F  1235 30$:
     0C A7    53    D0     044F  1236          MOVL    R3,ENQ$_LKSB(R7)        ; Address of LKSB
```

```
                54    86    9A  0453  1237          MOVZBL  (R6)+,R4                        ; Parent lock indicator
                  18  A7    D4  0456  1238          CLRL    ENQ$_PARID(R7)                  ; Assume no parent lock
                20    54    91  0459  1239          CMPB    R4,#^A/ /                       ; Is there parent lock?
                      14    13  045C  1240          BEQL    40$                             ; BR if no
      54  00000041  8F    C2  045E  1241          SUBL2   #^A/A/,R4                       ; Index of LKSB for parent lock
                54    18    C4  0465  1242          MULL2   #24,R4                          ; Displacement of LKSB of parent lock
           54   01F5'C4    DE  0468  1243          MOVAL   LKSBS(R4),R4                    ; Address of LKSB of parent lock
        18 A7     04  A4    D0  046D  1244          MOVL    4(R4),ENQ$_PARID(R7)            ; Parent lock id
                              0472  1245 40$:
           10  A7    86    9A  0472  1246          MOVZBL  (R6)+,ENQ$_FLAGS(R7)            ; Get enq flag
        0D 10  A7    03    E1  0476  1247          BBC     #LCK$V_SYNCSTS,ENQ$_FLAGS(R7),50$ ; BR if not SYNCSTS flag test
           04  A7    01    D0  047B  1248          MOVL    #SYNC_EF,ENQ$_EFN(R7)           ; Set up EFN for SYNCSTS test
                              047F  1249          $CLREF_S EFN = #SYNC_EF                 ; We need to initialize the efn for it
                              0488  1250 50$:
     1C  A7   07D6'CF    DE  0488  1251          MOVAL   COMP_AST,ENQ$_ASTADR(R7) ; Addr of completion AST
     20  A7   01ED'CF    D0  048E  1252          MOVL    LKID_ADDR,ENQ$_ASTPRM(R7) ; Set AST param to be addr of LKID
        008B'CF    86    B0  0494  1253          MOVW    (R6)+,TSTFLG                    ; Get test flag
     03 008B'CF    05    E1  0499  1254          BBC     #NOCAST_V,TSTFLG,60$            ; BR if no_completion_ast bit not set
                 1C  A7    D4  049F  1255          CLRL    ENQ$_ASTADR(R7)                 ; No completion AST
                              04A2  1256 60$:
                 24  A7    D4  04A2  1257          CLRL    ENQ$_BLKAST(R7)                 ; Assume blocking ast not requested
     06 008B'CF    00    E1  04A5  1258          BBC     #BLKAST_V,TSTFLG,70$            ; BR if block_ast bit not set
     24  A7   0807'CF    DE  04AB  1259          MOVAL   BLOCK_AST,ENQ$_BLKAST(R7) ; Set address of blocking AST routine
                              04B1  1260 70$:
     46 008B'CF    04    E1  04B1  1261          BBC     #DEADLK_V,TSTFLG,80$            ; BR if not deadlock special case
                              04B7  1262          $CLREF_S EFN = #DLDET_CEF               ; Deadlock must be detected and...
                              04C4  1263          $CLREF_S EFN = #DLRES_CEF               ; ...resolved. One flag per condition
     1C  A7   083F'CF    DE  04D1  1264          MOVAL   DLOCK_AST,ENQ$_ASTADR(R7) ; Deadlock gets special AST
         FBD3E280  8F    C5  04D7  1265          MULL3   #-10000000*<PROC_COUNT+5>,- ; To prevent hanging forever...
     51  00000000'GF         04DD  1266                  G^LCK$GL_WAITTIME,R1            ; ...while waiting for...
        00A6'CF    51    D0  04E3  1267          MOVL    R1,DLOCK_TIME                   ; ...deadlock detection....
                              04E8  1268          $SETIMR_S DAYTIM = DLOCK_TIME,-         ; ...we set a timer...
                              04E8  1269                    ASTADR = DLOCK_TO_AST,-       ; ...which will get us out...
                              04E8  1270                    REQIDT = LKID_ADDR            ; ...if need be
                              04FD  1271 80$:
           0D 01DA'CF    E8  04FD  1272          BLBS    TYPE,ENQWS                      ; Dispatch for correct system service
                              0502  1273                                                  ; (Correctness of TYPE guaranteed by...
                              0502  1274                                                  ; ...assembly time checks)
                              0502  1275 ENQS:
                              0502  1276          $ENQ_G  ENQLST                          ; Enqueue
                    028E    30  050B  1277          BSBW    CHK_SS                          ; Check R0 to see if we queued up OK
                        05    050E  1278          RSB                                     ; Return
                              050F  1279 ENQWS:
                              050F  1280          $ENQW_G ENQLST                          ; Enqueue and wait for EFN
                    008C    30  0518  1281          BSBW    CHECK_UP                        ; Go check final status
                        05    051B  1282          RSB                                     ; Return
                              051C  1283 DEQS:
  0073'CF    04  A3    D0  051C  1284          MOVL    4(R3),DEQLST+DEQ$_LKID          ; Lock id of the lock to be dequeued
                              0522  1285          $DEQ_G  DEQLST                          ; Dequeue
                    026E    30  052B  1286          BSBW    CHK_SS                          ; Check R0 to see if we dequeued OK
                        05    052E  1287          RSB                                     ; Return
```

```
                     052F  1289          .SBTTL  Type the Current Command
                     052F  1290 ;++
                     052F  1291 ; FUNCTIONAL DESCRIPTION:
                     052F  1292 ;       Type the current command to be executed.
                     052F  1293 ;
                     052F  1294 ; CALLING SEQUENCE:
                     052F  1295 ;       CALLS   #0,DUMP_COMMAND
                     052F  1296 ;
                     052F  1297 ; INPUT PARAMETERS:
                     052F  1298 ;       NONE
                     052F  1299 ;
                     052F  1300 ; IMPLICIT INPUTS:
                     052F  1301 ;       TST_COMMAND filled with next command to execute
                     052F  1302 ;
                     052F  1303 ; OUTPUT PARAMETERS:
                     052F  1304 ;       NONE
                     052F  1305 ;
                     052F  1306 ; IMPLICIT OUTPUTS:
                     052F  1307 ;       NONE
                     052F  1308 ;
                     052F  1309 ; COMPLETION CODES:
                     052F  1310 ;       NONE
                     052F  1311 ;
                     052F  1312 ; SIDE EFFECTS:
                     052F  1313 ;       Message to SYS$OUTPUT
                     052F  1314 ;--
                     052F  1315
                     052F  1316 DUMP_COMMAND:
                0FFC 052F  1317          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0531  1318
51   0084'CF    9A   0531  1319          MOVZBL  TST_COMMAND+1,R1        ; Get the type of SS call
52   0085'CF    9A   0536  1320          MOVZBL  TST_COMMAND+2,R2        ; Get the kind of lock we tried
     51    02   D1   053B  1321          CMPL    #DEQ,R1                 ; Were we doing a $DEQ?
           03   12   053E  1322          BNEQ    10$                     ; BR if not
     52    06   D0   0540  1323          MOVL    #LCK$K_EXMODE+1,R2      ; No mode for $DEQ, say that
                     0543  1324 10$:
     53    01   D0   0543  1325          MOVL    #1,R3                   ; Assume that we've a parent lock
54   0087'CF    DE   0546  1326          MOVAL   TST_COMMAND+4,R4
     64    20   91   054B  1327          CMPB    #^A/ /,(R4)             ; But have we?
           08   12   054E  1328          BNEQ    20$                     ; BR if we have
     53    04   D0   0550  1329          MOVL    #NONE_LENGTH,R3         ; We have none...
54   09DA'CF    DE   0553  1330          MOVAL   NONE,R4                 ; ...supply appropriate message
                     0558  1331 20$:
                     0558  1332          $FAO_S  CTRSTR = DUMP_MSG,-     ; Format the TEST_TABLE entry...
                     0558  1333                  OUTLEN = BUFFER_PTR,-   ; ...into something humanly readable
                     0558  1334                  OUTBUF = FAO_BUF,-
                     0558  1335                  P1     = #1,-
                     0558  1336                  P2     = #TST_COMMAND+0,-
                     0558  1337                  P3     = TEST_CODES[R1],-
                     0558  1338                  P4     = #1,-
                     0558  1339                  P5     = #TST_COMMAND+3,-
                     0558  1340                  P6     = LOCK_MODES[R2],-
                     0558  1341                  P7     = TST_COMMAND+5,-
                     0558  1342                  P8     = R3,-
                     0558  1343                  P9     = R4,-
                     0558  1344                  P10    = TST_COMMAND+6,-
                     0558  1345                  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  1352          .SBTTL  Check Correctness of the ENQW Test
                        05A7  1353 ;++
                        05A7  1354 ; FUNCTIONAL DESCRIPTION:
                        05A7  1355 ;       Check correctness of enqdeq service routine.
                        05A7  1356 ;
                        05A7  1357 ; CALLING SEQUENCE:
                        05A7  1358 ;       BSBW    CHECK_UP
                        05A7  1359 ;
                        05A7  1360 ; INPUT PARAMETERS:
                        05A7  1361 ;       NONE
                        05A7  1362 ;
                        05A7  1363 ; IMPLICIT INPUTS:
                        05A7  1364 ;       Status code in R0, LKSB
                        05A7  1365 ;       Address of LKSB in R3
                        05A7  1366 ;       Address of $ENQW argument list in R7
                        05A7  1367 ;
                        05A7  1368 ; OUTPUT PARAMETERS:
                        05A7  1369 ;       NONE
                        05A7  1370 ;
                        05A7  1371 ; IMPLICIT OUTPUTS:
                        05A7  1372 ;       Error message if error
                        05A7  1373 ;
                        05A7  1374 ; COMPLETION CODES:
                        05A7  1375 ;       in STATUS if error
                        05A7  1376 ;
                        05A7  1377 ; SIDE EFFECTS:
                        05A7  1378 ;       BSBW    CHK_XXXX for further check
                        05A7  1379 ;--
                        05A7  1380
                        05A7  1381 CHECK_UP:
03 10 A7   03   E1      05A7  1382          BBC     #LCK$V_SYNCSTS,ENQ$_FLAGS(R7),10$ ; BR if not SYNCSTS test
           00D6   31    05AC  1383          BRW     CHK_SYNCSTS                 ; Check SYNCSTS flag test
                        05AF  1384 10$:
03 10 A7   02   E1      05AF  1385          BBC     #LCK$V_NOQUEUE,ENQ$_FLAGS(R7),20$ ; BR if NOQUEUE not specfied
           0173   31    05B4  1386          BRW     CHK_NOQUEUE                 ; Check NOQUEUE flag test
                        05B7  1387 20$:
03 008B'CF 07   E1      05B7  1388          BBC     #VICTIM_V,TSTFLG,30$        ; BR if not dlock test w/ known victim
           01A3   31    05BD  1389          BRW     CHK_DEADLOCK                ; Check for deadlock - return via subr
                        05C0  1390 30$:
           01D9   30    05C0  1391          BSBW    CHK_SS                      ; Check R0 = #SS$_NORMAL
           01F3   30    05C3  1392          BSBW    CHK_LKSB                    ; Check LKSB status code = #SS$_NORMAL
                        05C6  1393
03 008B'CF 05   E0      05C6  1394          BBS     #NOCAST_V,TSTFLG,40$        ; BR if no_completion_ast specified
           0012   30    05CC  1395          BSBW    CHK_CAST                    ; Go check completion ast delivered
                        05CF  1396 40$:
03 008B'CF 01   E1      05CF  1397          BBC     #INCOMP_V,TSTFLG,50$        ; Incompatible flag set?
           0017   30    05D5  1398          BSBW    CHK_BLOCKAST                ; Go check blocking ast delivered
                        05D8  1399 50$:
03 10 A7   00   E1      05D8  1400          BBC     #LCK$V_VALBLK,ENQ$_FLAGS(R7),60$ ; BR if not value block test
           002A   30    05DD  1401          BSBW    CHK_VALBLK                  ; Go check value block test
                        05E0  1402 60$:
                   05   05E0  1403          RSB
```

```
                     05E1  1405          .SBTTL  Check Completion AST
                     05E1  1406 ;++
                     05E1  1407 ; FUNCTIONAL DESCRIPTION:
                     05E1  1408 ;       Check completion ast delivered routine.
                     05E1  1409 ;
                     05E1  1410 ; CALLING SEQUENCE:
                     05E1  1411 ;       BSBW    CHK_CAST
                     05E1  1412 ;
                     05E1  1413 ; INPUT PARAMETERS:
                     05E1  1414 ;       NONE
                     05E1  1415 ;
                     05E1  1416 ; IMPLICIT INPUTS:
                     05E1  1417 ;       NONE
                     05E1  1418 ;
                     05E1  1419 ; OUTPUT PARAMETERS:
                     05E1  1420 ;       NONE
                     05E1  1421 ;
                     05E1  1422 ; IMPLICIT OUTPUTS:
                     05E1  1423 ;       Exit message if error
                     05E1  1424 ;
                     05E1  1425 ; COMPLETION CODES:
                     05E1  1426 ;       in STATUS if error
                     05E1  1427 ;
                     05E1  1428 ; SIDE EFFECTS:
                     05E1  1429 ;       Exit if error
                     05E1  1430 ;--
                     05E1  1431
                     05E1  1432 CHK_CAST:
07 01E0'CF   00  E4  05E1  1433          BBSC    #C_AST_V,LOCFLG,10$     ; BR if comp_ast bit set (AST deliv.)
        03AC'CF  DF  05E7  1434          PUSHAL  CAST_ERRMSG             ; Error message
           FCFF  31  05EB  1435          BRW     FAIL_OUT                ; Failure exit
                     05EE  1436 10$:
                 05  05EE  1437          RSB                             ; Return
```

```
                        .SBTTL  Check Blocking AST
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;       Check blocking ast delivered routine.  The delivery of blocking AST
                ;       to another is indicated by setting a common EF BAST_CEF.
                ;
                ; CALLING SEQUENCE:
                ;       BSBW    CHK_BLOCKAST
                ;
                ; INPUT PARAMETERS:
                ;       NONE
                ;
                ; IMPLICIT INPUTS:
                ;       NONE
                ;
                ; OUTPUT PARAMETERS:
                ;       NONE
                ;
                ; IMPLICIT OUTPUTS:
                ;       Error message if error
                ;
                ; COMPLETION CODES:
                ;       In STATUS if error
                ;
                ; SIDE EFFECTS:
                ;       NONE
                ;--

                CHK_BLOCKAST:
                        $WAITFR_S EFN = #BAST_CEF       ; Wait for blocking ast delivered
                        $CLREF_S EFN = #BAST_CEF        ; Reset the blocking ast CEF flag
                        RSB                             ; Return
```

```
                     060A  1472          .SBTTL  Check Lock Value Block
                     060A  1473 ;++
                     060A  1474 ; FUNCTIONAL DESCRIPTION:
                     060A  1475 ;       Check value block test routine.
                     060A  1476 ;
                     060A  1477 ; CALLING SEQUENCE:
                     060A  1478 ;       BSBW    CHK_VALBLK
                     060A  1479 ;
                     060A  1480 ; INPUT PARAMETERS:
                     060A  1481 ;       NONE
                     060A  1482 ;
                     060A  1483 ; IMPLICIT INPUTS:
                     060A  1484 ;       Address of $ENQ argument list in R7
                     060A  1485 ;
                     060A  1486 ; OUTPUT PARAMETERS:
                     060A  1487 ;       NONE
                     060A  1488 ;
                     060A  1489 ; IMPLICIT OUTPUTS:
                     060A  1490 ;       Error message if error
                     060A  1491 ;
                     060A  1492 ; COMPLETION CODES:
                     060A  1493 ;       In STATUS if error
                     060A  1494 ;
                     060A  1495 ; SIDE EFFECTS:
                     060A  1496 ;       NONE
                     060A  1497 ;--
                     060A  1498
                     060A  1499 CHK_VALBLK:
   58   01F1'CF   D0  060A  1500          MOVL    LKSB_ADDR,R8                    ; Addr of LKSB
 07 10 A7    01   E0  060F  1501          BBS     #LCK$V_CONVERT,ENQ$_FLAGS(R7),10$ ; BR if not a new lock
01E5'CF    08 A8   7D  0614  1502          MOVQ    8(R8),SAVED_VAL                 ; Save the low quadword of lock value bk
                   05  061A  1503          RSB                                     ; Return
                       061B  1504 10$:                                             ; Not new lock
01 008B'CF    02   E0  061B  1505          BBS     #VALBLK_V,TSTFLG,20$            ; BR if VALBLK test flag set
                   05  0621  1506          RSB                                     ; Return
                       0622  1507 20$:
                       0622  1508          $READEF_S EFN = #CMP_VAL,-              ; Read the CEF flag
                       0622  1509                    STATE = EF_STATE
11 01E1'CF    02   E0  0633  1510          BBS     #CMP_VAL_V,EF_STATE,30$ ; BR if CMP_VAL flag set
                       0639  1511          $SETEF_S EFN = #CMP_VAL                 ; Set the common EF CMP_VAL
         08 A8   D6  0646  1512          INCL    8(R8)                           ; Increment the user lock value block
                   05  0649  1513          RSB                                     ; Return
                       064A  1514 30$:
      01E5'CF   D6  064A  1515          INCL    SAVED_VAL                       ; Increment the saved value
01E5'CF    08 A8   D1  064E  1516          CMPL    8(R8),SAVED_VAL                 ; Is the lock value OK?
             0E    12  0654  1517          BNEQ    40$                             ; Error if not equal
                       0656  1518          $CLREF_S EFN = #CMP_VAL                 ; Reset the CEF
                   05  0663  1519          RSB                                     ; Return
                       0664  1520 40$:
                       0664  1521          $FAO_S  CTRSTR = LKVAL_ERRMSG,- ; Value block error message
                       0664  1522                  OUTLEN = BUFFER_PTR,-
                       0664  1523                  OUTBUF = FAO_BUF,-
                       0664  1524                  P1     = 08(R8),-
                       0664  1525                  P2     = SAVED_VAL
      0481'CF   DF  067E  1526          PUSHAL  BUFFER_PTR
         FC68   31  0682  1527          BRW     FAIL_OUT
                       0685  1528
```

```
                     0685  1530          .SBTTL  Check SYNCSTS Flag Routine
                     0685  1531 ;++
                     0685  1532 ; FUNCTIONAL DESCRIPTION:
                     0685  1533 ;       Check LCK$M_SYNCSTS flag test routine
                     0685  1534 ;
                     0685  1535 ; CALLING SEQUENCE:
                     0685  1536 ;       BSBW    CHK_SYNCSTS
                     0685  1537 ;
                     0685  1538 ; INPUT PARAMETERS:
                     0685  1539 ;       NONE
                     0685  1540 ;
                     0685  1541 ; IMPLICIT INPUTS:
                     0685  1542 ;       R0 is the returned status from $ENQW
                     0685  1543 ;       LOCFLG synchronizes us with other test routines
                     0685  1544 ;       Test flag in TSTFLG
                     0685  1545 ;
                     0685  1546 ; OUTPUT PARAMETERS:
                     0685  1547 ;       NONE
                     0685  1548 ;
                     0685  1549 ; IMPLICIT OUTPUTS:
                     0685  1550 ;       Error message if error
                     0685  1551 ;
                     0685  1552 ; COMPLETION CODES:
                     0685  1553 ;       In STATUS if error
                     0685  1554 ;
                     0685  1555 ; SIDE EFFECTS:
                     0685  1556 ;       NONE
                     0685  1557 ;--
                     0685  1558 CHK_SYNCSTS:
56 008B'CF    03     E1    0685  1559          BBC     #SYNCST_V,TSTFLG,30$    ; BR if SYNCH flag cleared
   0000'8F    50     B1    068B  1560          CMPW    R0,#SS$_SYNCH           ; Compare R0 status code
              28     13    0690  1561          BEQL    10$                     ; BR if equal
              50     DD    0692  1562          PUSHL   R0                      ; Save the status...
                           0694  1563          $FAO_S  CTRSTR = NOSYNCH_ERRMSG,- ; ...and give a useful error
                           0694  1564                  OUTLEN = BUFFER_PTR,-
                           0694  1565                  OUTBUF = FAO_BUF,-
                           0694  1566                  P1     = R0
         0481'CF     DF    06A9  1567          PUSHAL  BUFFER_PTR
              01     DD    06AD  1568          PUSHL   #1
     00741132 8F     DD    06AF  1569          PUSHL   #UETP$_TEXT!STS$K_ERROR
              04     DD    06B5  1570          PUSHL   #4
            0548     31    06B7  1571          BRW     ERROR_EXIT
                           06BA  1572 10$:
07 01E0'CF    00     E1    06BA  1573          BBC     #C_AST_V,LOCFLG,20$     ; Should no comp ast delivered
         0465'CF     DF    06C0  1574          PUSHAL  CASTSYNCH_ERRMSG        ; Error message
            FC26     31    06C4  1575          BRW     FAIL_OUT                ; Failure exit
                           06C7  1576 20$:
                           06C7  1577          $READEF_S EFN = #SYNC_EF,-      ; Read EF
                           06C7  1578                    STATE = EF_STATE
4F 01E1'CF    01     E1    06D4  1579          BBC     #SYNC_EF,EF_STATE,50$   ; Error if SYNC EFN set
         04D3'CF     DF    06DA  1580          PUSHAL  SYNCH_ERRMSG            ; Error message
            FC0C     31    06DE  1581          BRW     FAIL_OUT                ; Failure out
                           06E1  1582 30$:
   0000'8F    50     B1    06E1  1583          CMPW    R0,#SS$_NORMAL          ; Check R0
              28     13    06E6  1584          BEQL    40$                     ; BR if equal
              50     DD    06E8  1585          PUSHL   R0                      ; Save the status...
                           06EA  1586          $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   0708  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  1602          .SBTTL  Check NOQUEUE Flag Routine
                        072A  1603 ;++
                        072A  1604 ; FUNCTIONAL DESCRIPTION:
                        072A  1605 ;       Check LCK$M_NOQUEUE flag function test routine
                        072A  1606 ;
                        072A  1607 ; CALLING SEQUENCE:
                        072A  1608 ;       BSBW    CHK_NOQUEUE
                        072A  1609 ;
                        072A  1610 ; INPUT PARAMETERS:
                        072A  1611 ;       NONE
                        072A  1612 ;
                        072A  1613 ; IMPLICIT INPUTS:
                        072A  1614 ;       Status code in R0
                        072A  1615 ;
                        072A  1616 ; OUTPUT PARAMETERS:
                        072A  1617 ;       NONE
                        072A  1618 ;
                        072A  1619 ; IMPLICIT OUTPUTS:
                        072A  1620 ;       Error message if error
                        072A  1621 ;
                        072A  1622 ; COMPLETION CODES:
                        072A  1623 ;       STATUS if error
                        072A  1624 ;
                        072A  1625 ; SIDE EFFECTS:
                        072A  1626 ;       NONE
                        072A  1627 ;--
                        072A  1628
                        072A  1629 CHK_NOQUEUE:
0000'8F    50    B1     072A  1630          CMPW    R0,#SS$_NOTQUEUED       ; Is R0 status code OK?
                01  12  072F  1631          BNEQ    10$                     ; BR if not
                    05  0731  1632          RSB                             ; Return
                        0732  1633 10$:
0511'CF    50    D0     0732  1634          MOVL    R0,STATUS               ; Save returned status as exit status
                        0737  1635          $FAO_S  CTRSTR = PAR_ERRMSG,-   ; Wrong status returned
                        0737  1636                  OUTLEN = BUFFER_PTR,-
                        0737  1637                  OUTBUF = FAO_BUF,-
                        0737  1638                  P1     = STATUS
     0511'CF       DD   074E  1639          PUSHL   STATUS
     0481'CF       DF   0752  1640          PUSHAL  BUFFER_PTR
           01      DD   0756  1641          PUSHL   #1
 00741132 8F       DD   0758  1642          PUSHL   #UETP$_TEXT!STS$K_ERROR
           04      DD   075E  1643          PUSHL   #4
         049F      31   0760  1644          BRW     ERROR_EXIT
```

```
                     0763  1646          .SBTTL  Check Deadlock Test
                     0763  1647 ;++
                     0763  1648 ; FUNCTIONAL DESCRIPTION:
                     0763  1649 ;       Check deadlock test routine.  This routine is executed only in those
                     0763  1650 ;       situations where a victim can be preselected and the results of
                     0763  1651 ;       requesting a lock predetermined.
                     0763  1652 ;
                     0763  1653 ; CALLING SEQUENCE:
                     0763  1654 ;       BSBW    CHK_DEADLOCK
                     0763  1655 ;
                     0763  1656 ; INPUT PARAMETERS:
                     0763  1657 ;       Status code in R0
                     0763  1658 ;
                     0763  1659 ; IMPLICIT INPUTS:
                     0763  1660 ;       R3 points to lock status block
                     0763  1661 ;
                     0763  1662 ; OUTPUT PARAMETERS:
                     0763  1663 ;       NONE
                     0763  1664 ;
                     0763  1665 ; IMPLICIT OUTPUTS:
                     0763  1666 ;       Error message if expected status not found
                     0763  1667 ;
                     0763  1668 ; COMPLETION CODES:
                     0763  1669 ;       NONE
                     0763  1670 ;
                     0763  1671 ; SIDE EFFECTS:
                     0763  1672 ;       Program exits if expected status not found
                     0763  1673 ;--
                     0763  1674
                     0763  1675 CHK_DEADLOCK:
0000'8F    63    B1  0763  1676          CMPW    (R3),#SS$_DEADLOCK      ; Is the status code OK?
          01    12  0768  1677          BNEQ    10$                     ; BR if not
                05  076A  1678          RSB
                    076B  1679 10$:
0511'CF    63    3C  076B  1680          MOVZWL  (R3),STATUS             ; Save returned status as exit status
                    0770  1681          $FAO_S  CTRSTR = VICTIM_ERRMSG,- ; Wrong status returned
                    0770  1682                  OUTLEN = BUFFER_PTR,-
                    0770  1683                  OUTBUF = FAO_BUF,-
                    0770  1684                  P1     = STATUS
     0511'CF    DD  0787  1685          PUSHL   STATUS
     0481'CF    DF  078B  1686          PUSHAL  BUFFER_PTR
           01    DD  078F  1687          PUSHL   #1
 00741132 8F    DD  0791  1688          PUSHL   #UETP$_TEXT!STS$K_ERROR
           04    DD  0797  1689          PUSHL   #4
         0466    31  0799  1690          BRW     ERROR_EXIT
```

```
                                079C  1692          .SBTTL  Check Status Code Subroutine
                                079C  1693 ;++
                                079C  1694 ; FUNCTIONAL DESCRIPTION:
                                079C  1695 ;       Subroutine to check normal status code in R0 (CHK_SS)
                                079C  1696 ;       and LKSB (CHK_LKSB), exit if error is found.
                                079C  1697 ;
                                079C  1698 ; CALLING SEQUENCE:
                                079C  1699 ;       BSBW    CHK_SS
                                079C  1700 ;       BSBW    CHK_LKSB
                                079C  1701 ;
                                079C  1702 ; INPUT PARAMETERS:
                                079C  1703 ;       NONE
                                079C  1704 ;
                                079C  1705 ; IMPLICIT INPUTS:
                                079C  1706 ;       Status code in R0 (CHK_SS), in LKSB (CHK_LKSB)
                                079C  1707 ;       Address of LKSB in R3 (CHK_LKSB)
                                079C  1708 ;
                                079C  1709 ; OUTPUT PARAMETERS:
                                079C  1710 ;       NONE
                                079C  1711 ;
                                079C  1712 ; IMPLICIT OUTPUTS:
                                079C  1713 ;       Exit if error
                                079C  1714 ;
                                079C  1715 ; COMPLETION CODES:
                                079C  1716 ;       In STATUS if error
                                079C  1717 ;
                                079C  1718 ; SIDE EFFECTS:
                                079C  1719 ;       Exit if error
                                079C  1720 ;--
                                079C  1721
                                079C  1722 CHK_SS:
0000'8F    50    B1              079C  1723          CMPW    R0,#SS$_NORMAL          ; Success ?
           01    12              07A1  1724          BNEQ    10$                     ; BR if not
                 05              07A3  1725          RSB                             ; Return
                                 07A4  1726 10$:
0511'CF    50    D0              07A4  1727          MOVL    R0,STATUS               ; Status code in STATUS
           02    F0              07A9  1728          INSV    #STS$K_ERROR,-          ; We'll force severity to be...
           00                    07AB  1729                  #STS$V_SEVERITY,-       ; ...an error, always
0511'CF    03                    07AC  1730                  #STS$S_SEVERITY,STATUS
     0511'CF    DD              07B0  1731          PUSHL   STATUS
           01    DD              07B4  1732          PUSHL   #1
         0449    31              07B6  1733          BRW     ERROR_EXIT
                                 07B9  1734
                                 07B9  1735
                                 07B9  1736 CHK_LKSB:
0000'8F    63    B1              07B9  1737          CMPW    (R3),#SS$_NORMAL        ; Status code in LKSB correct?
           01    12              07BE  1738          BNEQ    10$                     ; BR if not
                 05              07C0  1739          RSB
                                 07C1  1740 10$:
0511'CF    63    3C              07C1  1741          MOVZWL  (R3),STATUS             ; Error status code
           02    F0              07C6  1742          INSV    #STS$K_ERROR,-          ; We'll force severity to be...
           00                    07C8  1743                  #STS$V_SEVERITY,-       ; ...an error, always
0511'CF    03                    07C9  1744                  #STS$S_SEVERITY,STATUS
     0511'CF    DD              07CD  1745          PUSHL   STATUS
           01    DD              07D1  1746          PUSHL   #1
         042C    31              07D3  1747          BRW     ERROR_EXIT
```

```
                     07D6  1749          .SBTTL  Completion AST Routine
                     07D6  1750 ;++
                     07D6  1751 ; FUNCTIONAL DESCRIPTION:
                     07D6  1752 ;       ENQ(W) completion ast routine. This routine set a flag in LOCFLG
                     07D6  1753 ;       to indicate the completion ast being delivered and check the ast
                     07D6  1754 ;       parameter.
                     07D6  1755 ;
                     07D6  1756 ; CALLING SEQUENCE:
                     07D6  1757 ;       Called via AST when system service ENQ(W) complete
                     07D6  1758 ;
                     07D6  1759 ; INPUT PARAMETERS:
                     07D6  1760 ;       AST parameter = LKID_ADDR
                     07D6  1761 ;
                     07D6  1762 ; IMPLICIT INPUTS:
                     07D6  1763 ;       NONE
                     07D6  1764 ;
                     07D6  1765 ; OUTPUT PARAMETERS:
                     07D6  1766 ;       NONE
                     07D6  1767 ;
                     07D6  1768 ; IMPLICIT OUTPUTS:
                     07D6  1769 ;       Error message if error
                     07D6  1770 ;
                     07D6  1771 ; COMPLETION CODES:
                     07D6  1772 ;       In STATUS if error
                     07D6  1773 ;
                     07D6  1774 ; SIDE EFFECTS:
                     07D6  1775 ;       NONE
                     07D6  1776 ;--
                     07D6  1777
                     07D6  1778 COMP_AST:
                OFFC 07D6  1779          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     07D8  1780
   01E0'CF    01     88  07D8  1781          BISB2   #C_AST_M,LOCFLG         ; Set completion ast deliv flag
01ED'CF    04 AC     D1  07DD  1782          CMPL    4(AP),LKID_ADDR         ; AST parameter OK ?
               01     12  07E3  1783          BNEQ    10$                     ; Error if not equal
               04     04  07E5  1784          RET
                     07E6  1785 10$:
                     07E6  1786          $FAO_S  CTRSTR = CASTPAR_ERRMSG,- ; AST parameter error message
                     07E6  1787                  OUTLEN = BUFFER_PTR,-
                     07E6  1788                  OUTBUF = FAO_BUF,-
                     07E6  1789                  P1     = 04(AP),-
                     07E6  1790                  P2     = LKID_ADDR
         0481'CF     DF  0800  1791          PUSHAL  BUFFER_PTR
            FAE6     31  0804  1792          BRW     FAIL_OUT                ; Failure exit
```

```
                     0807  1794          .SBTTL  Blocking AST Routine
                     0807  1795 ;++
                     0807  1796 ; FUNCTIONAL DESCRIPTION:
                     0807  1797 ;       This blocking ast routine is called whenever the lock granted
                     0807  1798 ;       blocks another lock request. The routine set a common EF for another
                     0807  1799 ;       process to check the ast delivered. It then dequeue the lock such
                     0807  1800 ;       that another process can get the requested lock. In SYNCSTS flag test,
                     0807  1801 ;       The routine waits for a common EF before dequeue the lock.
                     0807  1802 ;
                     0807  1803 ; CALLING SEQUENCE:
                     0807  1804 ;       Called via blocking AST
                     0807  1805 ;
                     0807  1806 ; INPUT PARAMETERS:
                     0807  1807 ;       LKID_ADDR as AST parameter
                     0807  1808 ;
                     0807  1809 ; IMPLICIT INPUTS:
                     0807  1810 ;       NONE
                     0807  1811 ;
                     0807  1812 ; OUTPUT PARAMETERS:
                     0807  1813 ;       NONE
                     0807  1814 ;
                     0807  1815 ; IMPLICIT OUTPUTS:
                     0807  1816 ;       Error message if error found
                     0807  1817 ;
                     0807  1818 ; COMPLETION CODES:
                     0807  1819 ;       In STATUS if error
                     0807  1820 ;
                     0807  1821 ; SIDE EFFECTS:
                     0807  1822 ;       NONE
                     0807  1823 ;--
                     0807  1824
                     0807  1825 BLOCK_AST:
                OFFC 0807  1826          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0809  1827
                     0809  1828          $SETEF_S EFN = #BAST_CEF         ; Set a CEF BAST_CEF
 0D 008B'CF    08   E1  0816  1829          BBC     #SYNCST_M,TSTFLG,10$     ; BR if SYNCST flag clear
                     081C  1830          $WAITFR_S EFN = #UNLOCK_CEF      ; Wait for the unlock flag
                     0829  1831 10$:
    58  01ED'CF    D0  0829  1832          MOVL    LKID_ADDR,R8             ; Address of LKID
                     082E  1833          $DEQ_S  LKID = (R8)               ; Dequeue the lock
          FF5E    30  0838  1834          BSBW    CHK_SS                   ; Check R0
                  04  083E  1835          RET
```

```
                        083F  1837          .SBTTL  Deadlock Detection and Resolution Routine
                        083F  1838 ;++
                        083F  1839 ; FUNCTIONAL DESCRIPTION:
                        083F  1840 ;       This routine is specified as the completion AST routine when detecting
                        083F  1841 ;       and resolving a deadlock situation between processes each of which is
                        083F  1842 ;       equally likely to be the deadlock victim.
                        083F  1843 ;
                        083F  1844 ; CALLING SEQUENCE:
                        083F  1845 ;       Called via AST if deadlock is detected or resolved
                        083F  1846 ;
                        083F  1847 ; INPUT PARAMETERS:
                        083F  1848 ;       04(AP) is LKID_ADDR
                        083F  1849 ;
                        083F  1850 ; IMPLICIT INPUTS:
                        083F  1851 ;       Associated lock status block
                        083F  1852 ;
                        083F  1853 ; OUTPUT PARAMETERS:
                        083F  1854 ;       NONE
                        083F  1855 ;
                        083F  1856 ; IMPLICIT OUTPUTS:
                        083F  1857 ;       Error message if error found
                        083F  1858 ;
                        083F  1859 ; COMPLETION CODES:
                        083F  1860 ;       NONE
                        083F  1861 ;
                        083F  1862 ; SIDE EFFECTS:
                        083F  1863 ;       Program exits on some errors.  Common event flags signifying deadlock
                        083F  1864 ;       detection or resolution may be set.  The victim process dequeues its
                        083F  1865 ;       request for a deadlocked resource.  Timers to prevent permanent
                        083F  1866 ;       deadlock are cancelled.
                        083F  1867 ;--
                        083F  1868
                        083F  1869 DLOCK_AST:
                   0FFC 083F  1870          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0841  1871
 53    04 AC    04     C3  0841  1872          SUBL3   #4,04(AP),R3            ; Figure the LKSB address
    63    0000'8F     B1  0846  1873          CMPW    #SS$_DEADLOCK,(R3)      ; Are we the victim process?
                 32     12  084B  1874          BNEQ    10$                     ; BR if not
                        084D  1875          $SETEF_S EFN = #DLDET_CEF       ; Indicate that a victim has been found
                        085A  1876          $DEQ_S  FLAGS = #LCK$M_DEQALL   ; Dequeue all my locks, allowing...
                        0869  1877                                          ; ...other process(es) to succeed
 50    00000000'8F     D1  0869  1878          CMPL    #SS$_NORMAL,R0          ; Did we dequeue all locks?
                 60     12  0870  1879          BNEQ    30$                     ; BR if not
                        0872  1880          $CANTIM_S REQIDT = 04(AP)       ; Dlock was found, we don't need timer
                   04   087E  1881          RET                             ; We're done with deadlock detection
```

```
                        087F  1883 10$:
                FF37    30  087F  1884          BSBW    CHK_LKSB                ; See if we got our lock, broken dlock
                        0882  1885          $READEF_S EFN = #DLDET_CEF,-        ; Did we get it because...
                        0882  1886                  STATE = EF_STATE
     07 01E1'CF    04   E0  0893  1887          BBS     #DLDET_CEF-V,EF_STATE,20$ ; ...some other process got deadlock?
           06C8'CF    DF  0899  1888          PUSHAL  NODLOCK_ERRMSG
               FA4D   31  089D  1889          BRW     FAIL_OUT
                        08A0  1890 20$:
                        08A0  1891          $SETEF_S EFN = #DLRES_CEF           ; Indicate that deadlock has been resolved
                        08AD  1892          $DEQ_S  FLAGS = #LCK$M_DEQALL       ; Dequeue all my locks, allowing...
                        08BC  1893                                              ; ...other tests to run
     50   00000000'8F    D1  08BC  1894          CMPL    #SS$_NORMAL,R0          ; Did we dequeue all locks?
                   0D   12  08C3  1895          BNEQ    30$                     ; BR if not
                        08C5  1896          $CANTIM_S REQIDT = 04(AP)           ; Dlock was resolved, we don't need timer
                        04  08D1  1897          RET
                        08D2  1898
                        08D2  1899 30$:
          0511'CF    50   D0  08D2  1900          MOVL    R0,STATUS               ; Save returned status as exit status
                        08D7  1901          $FAO_S  CTRSTR = DEQALL_ERRMSG,-    ; Wrong status returned
                        08D7  1902                  OUTLEN = BUFFER_PTR,-
                        08D7  1903                  OUTBUF = FAO_BUF,-
                        08D7  1904                  P1     = STATUS
                0511'CF    DD  08EE  1905          PUSHL   STATUS
                0481'CF    DF  08F2  1906          PUSHAL  BUFFER_PTR
                      01   DD  08F6  1907          PUSHL   #1
           00741132 8F     DD  08F8  1908          PUSHL   #UETP$_TEXT!STS$K_ERROR
                      04   DD  08FE  1909          PUSHL   #4
                   02FF    31  0900  1910          BRW     ERROR_EXIT
```

```
                     0903  1912          .SBTTL  Deadlock Timeout AST Routine
                     0903  1913 ;++
                     0903  1914 ; FUNCTIONAL DESCRIPTION:
                     0903  1915 ;       This routine executes only if deadlock was not detected or resolved to
                     0903  1916 ;       this process's satisfaction in some reasonable time, a multiple of the
                     0903  1917 ;       SYSGEN parameter of the interval for deadlock detection.
                     0903  1918 ;
                     0903  1919 ; CALLING SEQUENCE:
                     0903  1920 ;       Called via $SETIMR AST
                     0903  1921 ;
                     0903  1922 ; INPUT PARAMETERS:
                     0903  1923 ;       04(AP) is LKID_ADDR
                     0903  1924 ;
                     0903  1925 ; IMPLICIT INPUTS:
                     0903  1926 ;       NONE
                     0903  1927 ;
                     0903  1928 ; OUTPUT PARAMETERS:
                     0903  1929 ;       NONE
                     0903  1930 ;
                     0903  1931 ; IMPLICIT OUTPUTS:
                     0903  1932 ;       Error message
                     0903  1933 ;
                     0903  1934 ; COMPLETION CODES:
                     0903  1935 ;       NONE
                     0903  1936 ;
                     0903  1937 ; SIDE EFFECTS:
                     0903  1938 ;       Common event flags specifying deadlock detection and resolution are
                     0903  1939 ;       set.  The resource requested by this process is dequeued.
                     0903  1940 ;--
                     0903  1941
                     0903  1942 DLOCK_TO_AST:
                0FFC 0903  1943          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0905  1944
             07   C5 0905  1945          MULL3   #PROC_COUNT+5,-         ; Figure how long...
51   00000000'GF     09 7  1946                  G^LCK$GL_WAITTIME,R1    ; ...we waited for deadlock
                     090D  1947          $FAO_S  CTRSTR = DEADLK_ERRMSG,- ; Tell the world that deadlock...
                     090D  1948                  OUTLEN = BUFFER_PTR,-    ; ...detection seemed to fail
                     090D  1949                  OUTBUF = FAO_BUF,-
                     090D  1950                  P1     = R1
         0481'CF     DF 0922  1951          PUSHAL  BUFFER_PTR
            F9C4     31 0926  1952          BRW     FAIL_OUT                ; We can't continue - fail with msg
```

```
                     0929  1954          .SBTTL  Termination Mailbox AST Routine
                     0929  1955 ;++
                     0929  1956 ; FUNCTIONAL DESCRIPTION:
                     0929  1957 ;       Receives the termination mailboxes from driven processes.
                     0929  1958 ;
                     0929  1959 ; CALLING SEQUENCE:
                     0929  1960 ;       Called via AST
                     0929  1961 ;
                     0929  1962 ; INPUT PARAMETERS:
                     0929  1963 ;       NONE
                     0929  1964 ;
                     0929  1965 ; IMPLICIT INPUTS:
                     0929  1966 ;       MBX_IOSB has the result of the read of a termination mailbox
                     0929  1967 ;       EXIT_MSG is the buffer into which the mail is written
                     0929  1968 ;
                     0929  1969 ; OUTPUT PARAMETERS:
                     0929  1970 ;       NONE
                     0929  1971 ;
                     0929  1972 ; IMPLICIT OUTPUTS:
                     0929  1973 ;       Error message if incorrect or inconsistent info in mailbox
                     0929  1974 ;
                     0929  1975 ; COMPLETION CODES:
                     0929  1976 ;       NONE
                     0929  1977 ;
                     0929  1978 ; SIDE EFFECTS:
                     0929  1979 ;       May cause program termination
                     0929  1980 ;
                     0929  1981 ;--
                     0929  1982
                     0929  1983 EX_MBX_AST:
                OFFC 0929  1984          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     092B  1985
      5E 0174'CF  E9 092B  1986          BLBC    MBX_IOSB,50$                ; BR if IO was not successful
   03   017E'CF  B1 0930  1987          CMPW    EXIT_MSG+ACC$W_MSGTYP,#MSG$_DELPROC ; Is it a termination msg?
           73  12 0935  1988          BNEQ    60$                         ; BR if not
   52   0020'CF  DE 0937  1989          MOVAL   PROCIDS,R2                  ; Addr of process ids
                     093C  1990 10$:
   62   0178'CF  D1 093C  1991          CMPL    MBX_IOSB+4,(R2)             ; Process id found?
           0B  13 0941  1992          BEQL    20$                         ; BR if yes
      52   04 A2  DE 0943  1993          MOVAL   4(R2),R2                    ; Next process id
           62  D5 0947  1994          TSTL    (R2)                        ; End of process id table?
           F1  12 0949  1995          BNEQ    10$                         ; Try next if more
         007F  31 094B  1996          BRW     70$                         ; Process id not found
                     094E  1997 20$:
00C0'8F  0182'CF  B1 094E  1998          CMPW    EXIT_MSG+ACC$L_FINALSTS,#SS$_NORMAL ; Deleted normally?
           03  13 0955  1999          BEQL    30$
         0095  31 0957  2000          BRW     80$                         ; BR if not
                     095A  2001 30$:
         01D2'CF  D7 095A  2002          DECL    EX_PROC_CNT                 ; Decrement exist proc count
           03  12 095E  2003          BNEQ    40$                         ; BR if there are more to wait for
         00B3  31 0960  2004          BRW     WAKE_UP                     ; Go wake up the controller process
                     0963  2005 40$:
                     0963  2006          $QIO_S  CHAN = EX_MBXCHAN,-          ; QIO read to mailbox for next process
                     0963  2007                  FUNC = #IO$_READVBLK,-
                     0963  2008                  ASTADR = EX_MBX_AST,-
                     0963  2009                  IOSB = MBX_IOSB,-
                     0963  2010                  P1 = EXIT_MSG,-
```

```
                        0963  2011                    P2 = #ACC$K_TERMLEN
                  04    098D  2012          RET                             ; Return
                        098E  2013 50$:
0511'CF  0174'CF  3C    098E  2014          MOVZWL  MBX_IOSB,STATUS         ; Use mailbox status as exit status
         0511'CF  DD    0995  2015          PUSHL   STATUS
         0760'CF  DF    0999  2016          PUSHAL  NOTRMB_ERRMSG
                01  DD  099D  2017          PUSHL   #1
          00741132  8F  DD    099F  2018    PUSHL   #UETP$_TEXT!STS$K_ERROR
                04  DD  09A5  2019          PUSHL   #4
              0258  31  09A7  2020          BRW     ERROR_EXIT
                        09AA  2021 60$:
    56  017E'CF  3C    09AA  2022          MOVZWL  EXIT_MSG+ACC$W_MSGTYP,R6 ; Complain about the kind of msg we got
                       09AF  2023          $FAO_S  CTRSTR = MSGTYP_ERRMSG,-
                       09AF  2024                  OUTLEN = BUFFER_PTR,-
                       09AF  2025                  OUTBUF = FAO_BUF,-
                       09AF  2026                  P1     = R6,-
                       09AF  2027                  P2     = #MSG$_DELPROC
         0481'CF  DF   09C6  2028          PUSHAL  BUFFER_PTR              ; Error in termination mailbox
            F920  31   09CA  2029          BRW     FAIL_OUT                ; Failure out
                       09CD  2030 70$:
                       09CD  2031          $FAO_S  CTRSTR = TERMBX_ERRMSG,- ; We got a message for the wrong proc
                       09CD  2032                  OUTLEN = BUFFER_PTR,-
                       09CD  2033                  OUTBUF = FAO_BUF,-
                       09CD  2034                  P1     = MBX_IOSB+4,-
                       09CD  2035                  P2     = EXIT_MSG+ACC$L_FINALSTS
         0481'CF  DF   09E8  2036          PUSHAL  BUFFER_PTR
            F8FE  31   09EC  2037          BRW     FAIL_OUT
                       09EF  2038 80$:
20 01E0'CF    01  E0   09EF  2039          BBS     #DLPRC_V,LOCFLG,90$     ; BR if the process is deleted by LOCK00
                       09F5  2040          $FAO_S  CTRSTR = DETPRC_ERRMSG,- ; Error in detached proccess
                       09F5  2041                  OUTLEN = BUFFER_PTR,-
                       09F5  2042                  OUTBUF = FAO_BUF,-
                       09F5  2043                  P1     = (R2),-
                       09F5  2044                  P2     = EXIT_MSG+ACC$L_FINALSTS
         0481'CF  DF   0A0E  2045          PUSHAL  BUFFER_PTR
            F8D8  31   0A12  2046          BRW     FAIL_OUT                ; Failure out
                       0A15  2047 90$:
                  04   0A15  2048          RET
                       0A16  2049 WAKE_UP:
                       0A16  2050          $WAKE_S                         ; Wake up the controller
                  04   0A21  2051          RET                             ; Return
```

```
                     0A22  2053          .SBTTL  Timer Expiration Routine
                     0A22  2054 ;++
                     0A22  2055 ; FUNCTIONAL DESCRIPTION:
                     0A22  2056 ;       This routine will be called only if the timer which was set to prevent
                     0A22  2057 ;       program hangs goes off.
                     0A22  2058 ;
                     0A22  2059 ; CALLING SEQUENCE:
                     0A22  2060 ;       Called via AST at $SETIMR expiration.
                     0A22  2061 ;
                     0A22  2062 ; INPUT PARAMETERS:
                     0A22  2063 ;       NONE
                     0A22  2064 ;
                     0A22  2065 ; IMPLICIT INPUTS:
                     0A22  2066 ;       NONE
                     0A22  2067 ;
                     0A22  2068 ; OUTPUT PARAMETERS:
                     0A22  2069 ;       NONE
                     0A22  2070 ;
                     0A22  2071 ; IMPLICIT OUTPUTS:
                     0A22  2072 ;       NONE
                     0A22  2073 ;
                     0A22  2074 ; COMPLETION CODES:
                     0A22  2075 ;       #SS$_TIMEOUT
                     0A22  2076 ;
                     0A22  2077 ; SIDE EFFECTS:
                     0A22  2078 ;       NONE
                     0A22  2079 ;
                     0A22  2080 ;--
                     0A22  2081
                     0A22  2082  IME_OUT:
                OFFC 0A22  2083          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0A24  2084
00000000'8F      DD  0A24  2085          PUSHL   #SS$_TIMEOUT                ; Push the signal name
          01     DD  0A2A  2086          PUSHL   #1                          ; Push the argument count total
         01D3    31  0A2C  2087          BRW     ERROR_EXIT                  ; Bail out completely
```

```
0A2F  2089          .SBTTL  System Service Exception Handler
0A2F  2090 ;++
0A2F  2091 ; FUNCTIONAL DESCRIPTION:
0A2F  2092 ;       This routine is executed if a software or hardware exception occurs or
0A2F  2093 ;       if a LIB$SIGNAL system service is used to output a message.
0A2F  2094 ;       Information about this method of handling messages and errors can be
0A2F  2095 ;       found in the VMS COMMON RUN-TIME manual and in the VMS SYSTEM SERVICE
0A2F  2096 ;       manual.
0A2F  2097 ;
0A2F  2098 ; CALLING SEQUENCE:
0A2F  2099 ;       Entered via an exception from the system
0A2F  2100 ;
0A2F  2101 ; INPUT PARAMETERS:
0A2F  2102 ;       ERROR_COUNT    = previous cumulative error count
0A2F  2103 ;
0A2F  2104 ;          AP ---->      +---------------+
0A2F  2105 ;                        |       2       |
0A2F  2106 ;                        |---------------|
0A2F  2107 ;                        | SIGNL ARY PNT |
0A2F  2108 ;                        |---------------|
0A2F  2109 ;                        | MECH  ARY PNT |  ---------
0A2F  2110 ;                        |---------------|          ^
0A2F  2111 ;                        |       4       |          |
0A2F  2112 ;                        |---------------|          |
0A2F  2113 ;                        | ESTABLISH FP  |          |
0A2F  2114 ;                        |---------------|          |
0A2F  2115 ;                        |     DEPTH     |  Mechanism Array
0A2F  2116 ;                        |---------------|          |
0A2F  2117 ;                        |       R0      |          |
0A2F  2118 ;                        |---------------|          |
0A2F  2119 ;                        |       R1      |          v
0A2F  2120 ;                        |---------------|  ---------
0A2F  2121 ;                        |       N       |          ^
0A2F  2122 ;                        |---------------|          |
0A2F  2123 ;                        | CONDITION NAME|          |
0A2F  2124 ;                        |---------------|          |
0A2F  2125 ;                        | N-3 ADDITIONAL|     Signal Array
0A2F  2126 ;                        | LONG WORD ARGS|          |
0A2F  2127 ;                        |---------------|          |
0A2F  2128 ;                        |       PC      |          |
0A2F  2129 ;                        |---------------|          |
0A2F  2130 ;                        |       PSL     |          v
0A2F  2131 ;                        +---------------+  ---------
0A2F  2132 ; IMPLICIT INPUTS:
0A2F  2133 ;       NONE
0A2F  2134 ;
0A2F  2135 ; OUTPUT PARAMETERS:
0A2F  2136 ;       NONE
0A2F  2137 ;
0A2F  2138 ; IMPLICIT OUTPUTS:
0A2F  2139 ;       NONE
0A2F  2140 ;
0A2F  2141 ; COMPLETION CODES:
0A2F  2142 ;       NONE
0A2F  2143 ;
0A2F  2144 ; SIDE EFFECTS:
0A2F  2145 ;       May branch to ERROR_EXIT
0A2F  2145 ;--
```

```
                        0A2F  2146
                        0A2F  2147 SSERROR:
                   0FFC 0A2F  2148          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0A31  2149
                        0A31  2150          $SETAST_S ENBFLG = #0               ; Disable AST delivery
                     01 DD    0A3A  2151          PUSHL   #1                      ; Assume ASTs were enabled
           50 00000000'8F D1  0A3C  2152          CMPL    #SS$_WASSET,R0          ; Were ASTs enabled?
                     02 13    0A43  2153          BEQL    10$                     ; BR if they were
                     6E D4    0A45  2154          CLRL    (SP)                    ; Set ASTs to remain disabled
                        0A47  2155 10$:
                        0A47  2156          $SETSFM_S ENBFLG = #0               ; Disable SS failure mode
                     01 DD    0A50  2157          PUSHL   #1                      ; Assume SS failure mode was enabled
           50 00000000'8F D1  0A52  2158          CMPL    #SS$_WASSET,R0          ; Was SS failure mode enabled?
                     02 13    0A59  2159          BEQL    20$                     ; BR if it was
                     6E D4    0A5B  2160          CLRL    (SP)                    ; Set SS failure mode to remain off
                        0A5D  2161 20$:
              56 04 AC D0     0A5D  2162          MOVL    CHF$L_SIGARGLST(AP),R6  ; Get the signal array pointer
                     10 F0    0A61  2163          CMPZV   #STS$V_FAC_NO,-         ; Is this a message from LIB$SIGNAL?
                     0C       0A63  2164                  #STS$S_FAC_NO,-
                  04 A6       0A64  2165                  CHF$L_SIG_NAME(R6),-
              00000074 8F     0A66  2166                  #UETP$_FACILITY
                     30 12    0A6B  2167          BNEQ    30$                     ; BR if this is not a UETP exception
                  66 02 C2    0A6D  2168          SUBL2   #2,CHF$L_SIG_ARGS(R6)   ; Drop the PC and PSL
                        0A70  2169          $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6) ; Print the message
                     01 BA    0A7F  2170          POPR    #^M<R0>                 ; Restore SS failure mode...
                        0A81  2171          $SETSFM_S ENBFLG = R0               ; ...
                     01 BA    0A8A  2172          POPR    #^M<R0>                 ; Restore AST enable...
                        0A8C  2173          $SETAST_S ENBFLG = R0               ; ...
           50 00000000'8F D0  0A95  2174          MOVL    #SS$_NORMAL,R0          ; Supply a standard status for exit
                     04       0A9C  2175          RET                             ; Return to the program
                        0A9D  2176 30$:
        00000000'C0'8F D1     0A9D  2177          CMPL    #SS$_SSFAIL,-           ; RMS fails come through as SS fails
                  04 A6       0AA3  2178                  CHF$L_SIG_NAME(R6)
                     26 12    0AA5  2179          BNEQ    40$                     ; BR if this can't be an RMS failure
                     10 F0    0AA7  2180          CMPZV   #STS$V_FAC_NO,-         ; Is it an RMS failure?
                     0C       0AA9  2181                  #STS$S_FAC_NO,-
                  08 A6       0AAA  2182                  CHF$L_SIG_ARG1(R6),-
                     01       0AAC  2183                  #RMS$_FACILITY
                     1E 12    0AAD  2184          BNEQ    40$                     ; BR if not
                     01 BA    0AAF  2185          POPR    #^M<R0>                 ; Restore SS failure mode...
                        0AB1  2186          $SETSFM_S ENBFLG = R0               ; ...
                     01 BA    0ABA  2187          POPR    #^M<R0>                 ; Restore AST enable...
                        0ABC  2188          $SETAST_S ENBFLG = R0               ; ...
              50 00000000'8F D0  0AC5  2189          MOVL    #SS$_NORMAL,R0          ; Supply a standard status for exit
                     04       0ACC  2190          RET                             ; RMS_ERROR gets to handle these
                        0ACD  2191 40$:
        0511'CF 04 A6 D0        0ACD  2192          MOVL    CHF$L_SIG_NAME(R6),STATUS ; Save the status
                     58 D4      0AD3  2193          CLRL    R8                      ; Assume for now it's not SS failure
  0511'CF 00000000'8F D1        0AD5  2194          CMPL    #SS$_SSFAIL,STATUS      ; But is it a System Service failure?
                     3B 12      0ADE  2195          BNEQ    60$                     ; BR if not - no special case message
                        0AE0  2196          $GETMSG_S MSGID = CHF$L_SIG_ARG1(R6),- ; Get SS failure code associated text
                        0AE0  2197                    MSGLEN = BUFFER_PTR,-
                        0AE0  2198                    BUFADR = FAO_BUF,-
                        0AE0  2199                    FLAGS  = #14,-
                        0AE0  2200                    OUTADR = MSG_BLOCK
                 0476'CF 95     0AF8  2201          TSTB    MSG_BLOCK+1             ; Get FAO arg count for SS failure code
                     17 13      0AFC  2202          BEQL    50$                     ; Don't use $GETMSG if no $FAO args...
```

```
                                                I 14

                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
                   >8   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  2221          .SBTTL  RMS Error Handler
                                0B2E  2222 ;++
                                0B2E  2223 ; FUNCTIONAL DESCRIPTION:
                                0B2E  2224 ;       This routine handles error returns from RMS calls.
                                0B2E  2225 ;
                                0B2E  2226 ; CALLING SEQUENCE:
                                0B2E  2227 ;       Called by RMS when a file processing error is found.
                                0B2E  2228 ;
                                0B2E  2229 ; INPUT PARAMETERS:
                                0B2E  2230 ;       The FAB or RAB associated with the RMS call.
                                0B2E  2231 ;
                                0B2E  2232 ; IMPLICIT INPUTS:
                                0B2E  2233 ;       NONE
                                0B2E  2234 ;
                                0B2E  2235 ; OUTPUT PARAMETERS:
                                0B2E  2236 ;       NONE
                                0B2E  2237 ;
                                0B2E  2238 ; IMPLICIT OUTPUTS:
                                0B2E  2239 ;       Error message
                                0B2E  2240 ;
                                0B2E  2241 ; COMPLETION CODES:
                                0B2E  2242 ;       NONE
                                0B2E  2243 ;
                                0B2E  2244 ; SIDE EFFECTS:
                                0B2E  2245 ;       Program may exit, depending on severity of the error.
                                0B2E  2246 ;
                                0B2E  2247 ;--
                                0B2E  2248
                                0B2E  2249 RMS_ERROR:
                        0FFC    0B2E  2250          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                0B30  2251
          56   04 AC    D0    0B30  2252          MOVL    4(AP),R6                    ; See whether we're dealing with...
               66   03   91    0B34  2253          CMPB    #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
                    16   12    0B37  2254          BNEQ    10$                         ; BR if it's a RAB
          57   0898'CF   DE    0B39  2255          MOVAL   FILE,R7                     ; FAB-specific code:  text string...
               58   56   D0    0B3E  2256          MOVL    R6,R8                       ; ...address of FAB...
                 0C A6   DD    0B41  2257          PUSHL   FAB$L_STV(R6)               ; ...STV field for error...
                 08 A6   DD    0B44  2258          PUSHL   FAB$L_STS(R6)               ; ...STS field for error...
     0511'CF     08 A6   D0    0B47  2259          MOVL    FAB$L_STS(R6),STATUS        ; ...and save the error code
                    15   11    0B4D  2260          BRB     COMMON                      ; FAB and RAB share other code
                                0B4F  2261 10$:
          57   08A4'CF   DE    0B4F  2262          MOVAL   RECORD,R7                   ; RAB-specific code:  text string...
            58   3C A6   D0    0B54  2263          MOVL    RAB$L_FAB(R6),R8            ; ...address of associated FAB...
                 0C A6   DD    0B58  2264          PUSHL   RAB$L_STV(R6)               ; ...STV field for error...
                 08 A6   DD    0B5B  2265          PUSHL   RAB$L_STS(R6)               ; ...STS field for error...
     0511'CF     08 A6   D0    0B5E  2266          MOVL    RAB$L_STS(R6),STATUS        ; ...and save the error code
                                0B64  2267 COMMON:
          5A   34 A8   9A    0B64  2268          MOVZBL  FAB$B_FNS(R8),R10           ; Get the file name size
                             0B68  2269          $FAO_S  CTRSTR = RMS_ERR_STRING,-   ; Common code, prepare error message...
                             0B68  2270                  OUTLEN = BUFFER_PTR,-
                             0B68  2271                  OUTBUF = FAO_BUF,-
                             0B68  2272                  P1     = R7,-
                             0B68  2273                  P2     = R10,-
                             0B68  2274                  P3     = FAB$L_FNA(R8)
               0481'CF    DF    0B82  2275          PUSHAL  BUFFER_PTR                  ; ...and arguments for ERROR_EXIT...
                    01    DD    0B86  2276          PUSHL   #1                          ; ...
          00741130 8F     DD    0B88  2277          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  2285          .SBTTL  CTRL/C Handler
                                0B9D  2286 ;++
                                0B9D  2287 ; FUNCTIONAL DESCRIPTION:
                                0B9D  2288 ;       This routine handles CTRL/C AST's
                                0B9D  2289 ;
                                0B9D  2290 ; CALLING SEQUENCE:
                                0B9D  2291 ;       Called via AST
                                0B9D  2292 ;
                                0B9D  2293 ; INPUT PARAMETERS:
                                0B9D  2294 ;       NONE
                                0B9D  2295 ;
                                0B9D  2296 ; IMPLICIT INPUTS:
                                0B9D  2297 ;       NONE
                                0B9D  2298 ;
                                0B9D  2299 ; OUTPUT PARAMETERS:
                                0B9D  2300 ;       NONE
                                0B9D  2301 ;
                                0B9D  2302 ; IMPLICIT OUTPUTS:
                                0B9D  2303 ;       NONE
                                0B9D  2304 ;
                                0B9D  2305 ; COMPLETION CODES:
                                0B9D  2306 ;       NONE
                                0B9D  2307 ;
                                0B9D  2308 ; SIDE EFFECTS:
                                0B9D  2309 ;       NONE
                                0B9D  2310 ;
                                0B9D  2311 ;--
                                0B9D  2312
                                0B9D  2313 CCASTHAND:
                        0FFC    0B9D  2314          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                0B9F  2315
          28 01DE'CF    00    E0  0B9F  2316          BBS     #DRIVEN_V,GLBFLG,30$    ; Skip this if we are a driven proc
             52   02A9'CF    DE  0BA5  2317          MOVAL   PROCS,R2                ; Addr of process indicator
             53   0020'CF    DE  0BAA  2318          MOVAL   PROCIDS,R3              ; Addr of detached process id
                       55   D4  0BAF  2319          CLRL    R5                      ; Index
                                0BB1  2320 10$:
                       6345 D5  0BB1  2321          TSTL    (R3)[R5]                ; Is the process exited ?
                         13 13  0BB4  2322          BEQL    20$                     ; BR if yes
                                0BB6  2323          $FORCEX_S PIDADR = (R3)[R5],-   ; Delete the process
                                0BB6  2324                    CODE = #0             ; Status is zero
             01E0'CF    02    88  0BC4  2325          BISB2   #DLPRC_M,LOCFLG         ; Set the delete process flag
                                0BC9  2326 20$:
              E4 55     02    F2  0BC9  2327          AOBLSS  #PROC_COUNT,R5,10$      ; Next if more
                                0BCD  2328
                                0BCD  2329 30$:
                 02BB'CF    DF  0BCD  2330          PUSHAL  CNTRLCMSG               ; Set message pointer
                         01 DD  0BD1  2331          PUSHL   #1                      ; Set arg count
                00741130 8F    DD  0BD3  2332          PUSHL   #UETP$_TEXT!STS$K_WARNING ; Set signal name
                         00 DD  0BD9  2333          PUSHL   #0                      ; Indicate an abnormal termination
                 0008'CF    DF  0BDB  2334          PUSHAL  TEST_NAME_D             ; ...
                         02 DD  0BDF  2335          PUSHL   #2                      ; ...
                007410E0 8F    DD  0BE1  2336          PUSHL   #UETP$_ABENDD!STS$K_WARNING ; ...
          00000000'GF   07    FB  0BE7  2337          CALLS   #7,G^LIB$SIGNAL         ; Output the message...
                                D0  0BEE  2338          MOVL    #<STS$K_SUCCESS!SS$_CONTROLC-- ; ...and exit status
                                0BEF  2339                  STS$K_SUCCESS+STS$K_WARNING>+-
     0511'CF    10000000'8F        0BEF  2340                  STS$M_INHIB_MSG,STATUS
                                0BF7  2341          $EXIT_S STATUS                  ; Terminate program cleanly
```

```
                     0C02  2343          .SBTTL  Error Exit
                     0C02  2344 ;++
                     0C02  2345 ; FUNCTIONAL DESCRIPTION:
                     0C02  2346 ;       This routine prints an error message and exits.
                     0C02  2347 ;
                     0C02  2348 ; CALLING SEQUENCE:
                     0C02  2349 ;       PUSHx error specific information on the stack
                     0C02  2350 ;       PUSHL current argument count
                     0C02  2351 ;       BRW   ERROR_EXIT
                     0C02  2352 ;
                     0C02  2353 ; INPUT PARAMETERS:
                     0C02  2354 ;       Arguments to LIB$SIGNAL, as above
                     0C02  2355 ;
                     0C02  2356 ; IMPLICIT INPUTS:
                     0C02  2357 ;       NONE
                     0C02  2358 ;
                     0C02  2359 ; OUTPUT PARAMETERS:
                     0C02  2360 ;       Message to SYS$OUTPUT (log file) and SYS$ERROR
                     0C02  2361 ;
                     0C02  2362 ; IMPLICIT OUTPUTS:
                     0C02  2363 ;       Program exit
                     0C02  2364 ;
                     0C02  2365 ; COMPLETION CODES:
                     0C02  2366 ;       NONE
                     0C02  2367 ;
                     0C02  2368 ; SIDE EFFECTS:
                     0C02  2369 ;       NONE
                     0C02  2370 ;
                     0C02  2371 ;--
                     0C02  2372
                     0C02  2373 ERROR_EXIT:
1C 01DE'CF    02 E0  0C02  2374          BBS     #BEGIN_MSGV,GLBFLG,5$   ; BR if already given beginning sentinel
        022C'CF    DF  0C08  2375          PUSHAL  BEGUN_ADDR              ; Not yet, so do it before giving error
        0010'CF    DF  0C0C  2376          PUSHAL  TEST_NAME_C
             02    DD  0C10  2377          PUSHL   #2
    007480D9 8F    DD  0C12  2378          PUSHL   #UETP$_SATSMS!STS$K_SUCCESS
00000000'GF    04  FB  0C18  2379          CALLS   #4,G^LIB$SIGNAL         ; Squirt out the message...
   01DE'CF    04  A8  0C1F  2380          BISW2   #BEGIN_MSGM,GLBFLG      ; ...and mark it as done
                     0C24  2381 5$:
40 01DE'CF    00 E0  0C24  2382          BBS     #DRIVEN_V,GLBFLG,30$    ; Skip this if we are a driven proc
   52 02A9'CF    DE  0C2A  2383          MOVAL   PROCS,R2                ; Addr of process indicator
   53 0020'CF    DE  0C2F  2384          MOVAL   PROCIDS,R3              ; Addr of detached process id
             55  D4  0C34  2385          CLRL    R5                      ; Index
                     0C36  2386 10$:
           6345  D5  0C36  2387          TSTL    (R3)[R5]                ; Is the process exited ?
             13  13  0C39  2388          BEQL    20$                     ; BR if yes
                     0C3B  2389          $FORCEX_S PIDADR = (R3)[R5],-   ; Delete the process
                     0C3B  2390                    CODE = #0             ; Status is zero
   01E0'CF    02  88  0C49  2391          BISB2   #DLPRC_M,LOCFLG         ; Set the delete process flag
                     0C4E  2392 20$:
     E4 55    02  F2  0C4E  2393          AOBLSS  #PROC_COUNT,R5,10$      ; Next if more
                     0C52  2394          $SCHDWK_S DAYTIM = TEN_SECONDS  ; Cheap way to allow process rundown...
                     0C63  2395          $HIBER_S                        ; ...for the ones we just zapped
                     0C6A  2396
                     0C6A  2397 30$:
        0511'CF    D5  0C6A  2398          TSTL    STATUS                  ; Was any exit status supplied?
             09  12  0C6E  2399          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
                           04  8E  C1  0C79  2402 40$:
          050D'CF                  DD  0C79  2403          ADDL3   (SP)+,#4,ARG_COUNT      ; Get total # args, pop partial count
                          00       DD  0C7F  2404          PUSHL   #0                      ; Push the time parameter
                    0008'CF        DF  0C81  2405          PUSHAL  TEST_NAME_D             ; Push test name...
                          02       DD  0C85  2406          PUSHL   #2                      ; ...arg count...
                007410E2 8F        DD  0C87  2407          PUSHL   #UETP$_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  2412          .SBTTL  Exit Handler
                        0CAA  2413 ;++
                        0CAA  2414 ; FUNCTIONAL DESCRIPTION:
                        0CAA  2415 ;       Output the log file of detached processes to SYS$OUTPUT and exit.
                        0CAA  2416 ;
                        0CAA  2417 ; CALLING SEQUENCE:
                        0CAA  2418 ;       Invoked automatically by $EXIT system service.
                        0CAA  2419 ;
                        0CAA  2420 ; INPUT PARAMETERS:
                        0CAA  2421 ;       NONE
                        0CAA  2422 ;
                        0CAA  2423 ; IMPLICIT INPUTS:
                        0CAA  2424 ;       STATUS contains the exit status
                        0CAA  2425 ;       Log files of cooperating detached processes
                        0CAA  2426 ;
                        0CAA  2427 ; OUTPUT PARAMETERS:
                        0CAA  2428 ;       NONE
                        0CAA  2429 ;
                        0CAA  2430 ; IMPLICIT OUTPUTS:
                        0CAA  2431 ;       Contents of cooperating processes written to SYS$OUTPUT
                        0CAA  2432 ;
                        0CAA  2433 ; COMPLETION CODES:
                        0CAA  2434 ;       NONE
                        0CAA  2435 ;
                        0CAA  2436 ; SIDE EFFECTS:
                        0CAA  2437 ;       Log files of cooperating detached processes are deleted
                        0CAA  2438 ;
                        0CAA  2439 ;--
                        0CAA  2440 EXIT_HANDLER:
                  0FFC  0CAA  2441          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0CAC  2442
                        0CAC  2443          $SETSFM_S ENBFLG = #0           ; We can't trap System Service errors
                        0CB5  2444          $SETAST_S ENBFLG = #0           ; ASTs could screw us up now
                        0CBE  2445
    17 01DE'CF    02    E0  0CBE  2446          BBS     #BEGIN_MSGV,GLBFLG,5$   ; BR if already given beginning sentinel
           022C'CF    DF  0CC4  2447          PUSHAL  BEGUN_ADDR              ; Not yet, so do it before giving error
           0010'CF    DF  0CC8  2448          PUSHAL  TEST_NAME_C
                      02  DD  0CCC  2449          PUSHL   #2
       007480D9    8F    DD  0CCE  2450          PUSHL   #UETP$_SATSMS!STS$K_SUCCESS
00000000'GF        04    FB  0CD4  2451          CALLS   #4,G^LIB$SIGNAL         ; Squirt out the message...
                        0CDB  2452 ;
                        0CDB  2453 5$:          BISW2   #BEGIN_MSGM,GLBFLG      ; ...and mark it as done
                        0CDB  2454
    03 01DE'CF    00    E0  0CDB  2455          BBS     #DRIVEN_V,GLBFLG,10$    ; Ignore log files if we're driven proc
             0071    30  0CE1  2456          BSBW    TYPE_LOG_FILES          ; Dump driven process log file(s)
                        0CE4  2457
                        0CE4  2458 10$:
           0511'CF    D5  0CE4  2459          TSTL    STATUS                  ; The only way we get no status...
                 35    12  0CE8  2460          BNEQ    EXIT_MESSAGE            ; ...is if we were $FORCEXed. BR if OK
                        0CEA  2461          $FAO_S  CTRSTR = FORCEX_MSG,-   ; Type a message to tell of our...
                        0CEA  2462                  OUTLEN = BUFFER_PTR,-   ; ...$FORCEX plight
                        0CEA  2463                  OUTBUF = FAO_BUF,-
                        0CEA  2464                  P1     = #TEST_NAME_D
           0481'CF    DF  0CD3  2465          PUSHAL  BUFFER_PTR
                      01  DD  0CD7  2466          PUSHL   #1
       00741132    8F    DD  0CD9  2467          PUSHL   #UETP$_TEXT!STS$K_ERROR
00000000'GF        03    FB  0CDF  2468          CALLS   #3,G^LIB$SIGNAL
```

```
                                0D1F  2469          MOVL    #UETP$_ABENDD!STS$K_ERROR!- ; Supply a default exit status
                                0D1F  2470                  STS$M_INHIB_MSG,STATUS
                                0D1F  2471
                                0D1F  2472 EXIT_MESSAGE:
                    0232'CF     DF  0D1F  2473          PUSHAL  END_ADDR                    ; Assume we completed successfully
                 05 0511'CF     E8  0D23  2474          BLBS    STATUS,10$                  ; But did we?
                6E  023D'CF     DE  0D28  2475          MOVAL   FAIL_ADDR,(SP)              ; Modify message if we failed
                                0D2D  2476 10$:
                    0010'CF     DF  0D2D  2477          PUSHAL  TEST_NAME_C
                        02      DD  0D31  2478          PUSHL   #2                          ; Argument count
                 00748009 8F    DD  0D33  2479          PUSHL   #UETP$_SATSMS               ; Set the end message code
                00  0511'CF     F0  0D39  2480          INSV    STATUS,#STS$V_SEVERITY,- ; Give the test status
                    6E  03          0D3E  2481                  #STS$S_SEVERITY,(SP)
                        04      DD  0D40  2482          PUSHL   #4
                    51  5E      D0  0D42  2483          MOVL    SP,R1
                                0D45  2484          $PUTMSG_S MSGVEC = (R1)             ; Print the ending message
                                04  0D54  2485          RET
```

```
                        0D55  2487 ;
                        0D55  2488 ; This routine does "typical" UETP log file output, i.e., it reads a file,
                        0D55  2489 ; searching for a keyword used as a beginning sentinel.  When the sentinel
                        0D55  2490 ; is found, all lines of the file up to, but not including, a line containing
                        0D55  2491 ; and ending sentinel are copied to SYS$OUTPUT.  An end of file counts as an
                        0D55  2492 ; ending sentinel.  The copying algorithm is done for all the driven
                        0D55  2493 ; processes started up by this driver.
                        0D55  2494 ;
                        0D55  2495 TYPE_LOG_FILES:
                  56    D4    0D55  2496         CLRL    R6                          ; Initialize pointer to table of...
                        0D57  2497                                                   ; ...qualifiers for driven proc names
                        0D57  2498 10$:
          0020'CF46     D5    0D57  2499         TSTL    PROCIDS[R6]                 ; Was this process ever started?
                  03    12    0D5C  2500         BNEQ    15$                         ; BR if it was
                 00CC   31    0D5E  2501         BRW     60$                         ; It wasn't. There can be no log to copy
                        0D61  2502 15$:
0034'CF   02A9'CF46     90    0D61  2503         MOVB    PROCS[R6],LOG_FILE_QUAL     ; Set up name for this log file
                        0D69  2504         $OPEN   FAB = LOG_FAB               ; Open the log file
                  012A  30    0D74  2505         BSBW    FAB_ERROR
                11 50   E9    0D77  2506         BLBC    R0,17$                      ; Don't try to read if we have error
                        0D7A  2507         $CONNECT RAB = LOG_RAB              ; Connect to log file
                  010B  30    0D85  2508         BSBW    RAB_ERROR
                03 50   E8    0D88  2509         BLBS    R0,20$                      ; BR if all OK
                        0D8B  2510 17$:
                  0089  31    0D8B  2511         BRW     50$                         ; Don't try to read if we have error
                        0D8E  2512 20$:
                        0D8E  2513         $GET    RAB = LOG_RAB               ; Read a record looking for sentinel
                31 50   E9    0D99  2514         BLBC    R0,30$                      ; BR if error
          54    022C'CF 9B    0D9C  2515         MOVZBW  BEGUN_ADDR,R4               ; Form MATCHC length argument
          022D'CF   54  39    0DA1  2516         MATCHC  R4,BEGUN_ADDR+1,-           ; Did we find a beginning sentinel?
0525'CF   0084  8F          0DA6  2517                 #TEXT_BUFFER,RMS_BUFFER
                  E0    12    0DAC  2518         BNEQ    20$                         ; Loop for next record if not
                        0DAE  2519         $GET    RAB = LOG_RAB               ; Anticipate the next record
          54    0232'CF 9B    0DB9  2520         MOVZBW  END_ADDR,R4                 ; Form MATCHC length argument
          0233'CF   54  39    0DBE  2521         MATCHC  R4,END_ADDR+1,-             ; Did we find an ending sentinel?
0525'CF   0084  8F          0DC3  2522                 #TEXT_BUFFER,RMS_BUFFER
                  4C    13    0DC9  2523         BEQL    50$                         ; BR if we did - nothing to copy
                  67    10    0DCB  2524         BSBB    TYPE_INTRO                  ; Introduce anything we might copy
                        0DCD  2525 30$:
                        0DCD  2526         $GET    RAB = LOG_RAB               ; Read a record to copy
                2D 50   E9    0DD8  2527         BLBC    R0,40$                      ; BR if error
          54    0232'CF 9B    0DDB  2528         MOVZBW  END_ADDR,R4                 ; Form MATCHC length argument
          0233'CF   54  39    0DE0  2529         MATCHC  R4,END_ADDR+1,-             ; Did we find an ending sentinel?
0525'CF   0084  8F          0DE5  2530                 #TEXT_BUFFER,RMS_BUFFER
                  2A    13    0DEB  2531         BEQL    50$                         ; BR if we did
          061E'CF   04  A1    0DED  2532         ADDW3   #INDENT,LOG_RAB+RAB$W_RSZ,- ; Set the size of our message
                051D'CF       0DF2  2533                 LOG_MSGPTR
                        0DF5  2534         $PUTMSG_S MSGVEC = LOG_MSGVEC       ; All driven proc msgs go to SYS$OUTPUT
                  C5    11    0E06  2535         BRB     30$                         ; Do the next record
                        0E08  2536 40$:
 00000000'8F      50    D1    0E08  2537         CMPL    R0,#RMS$_EOF                ; End of file ?
                  06    13    0E0F  2538         BEQL    50$                         ; BR if it is
                 007F   30    0E11  2539         BSBW    RAB_ERROR                   ; Give an error otherwise...
                16 50   E9    0E14  2540         BLBC    R0,60$                      ; ...and skip the rest of this one
                        0E17  2541 50$:
                        0E17  2542         $CLOSE  FAB = LOG_FAB               ; Close the log file, ignoring errors
                        0E22  2543         $ERASE  FAB = LOG_FAB               ; Delete the log file, ignoring errors
```

```
                OE2D  2544 60$:
FF24 56   01  01  F1  OE2D  2545        ACBL    #PROC_COUNT-1,#1,R6,10$ ; Loop through all driven processes
                  05  OE33  2546        RSB                             ; Return to mainline code
```

```
                        0E34  2548 TYPE_INTRO:
051D'CF   00000084 8F   D0  0E34  2549          MOVL    #TEXT_BUFFER,LOG_MSGPTR ; Set up...
      50   02A9'C6   9E   0E3D  2550          MOVAB   PROCS(R6),R0            ; ...a message...
                        0E42  2551          $FAO_S  CTRSTR = COPY_LOG_MSG,- ; ...to introduce the log file...
                        0E42  2552                  OUTLEN = LOG_MSGPTR,-
                        0E42  2553                  OUTBUF = LOG_MSGPTR,-
                        0E42  2554                  P1     = #LOG_FILE_DESC,-
                        0E42  2555                  P2     = #1,-
                        0E42  2556                  P3     = R0
                        0E5F  2557          $PUTMSG_S MSGVEC = LOG_MSGVEC   ; ...if there is stuff to copy
            02          90  0E70  2558          MOVB    #RAB$C_RFA,-            ; Now set up...
        061A'CF             0E72  2559                  LOG_RAB+RAB$B_RAC
                        0E75  2560          $FIND   RAB = LOG_RAB,-         ; ...so that we can...
                        0E75  2561                  ERR = RMS_ERROR
            00          90  0E84  2562          MOVB    #RAB$C_SEQ,-            ; ...reread that last record
        061A'CF             0E86  2563                  LOG_RAB+RAB$B_RAC
    20  00 8F          00   2C  0E89  2564          MOVC5   #0,#0,#^A/ /,-          ; Set up indentation to mark...
      0525'CF          04       0E8E  2565                  #INDENT,RMS_BUFFER      ; ...subprocess log from our messages
                            05  0E92  2566          RSB
                        0E93  2567
                        0E93  2568 RAB_ERROR:
              01 50     E9  0E93  2569          BLBC    R0,10$                  ; If there is no error...
                            05  0E96  2570          RSB                             ; ...we take no action
                        0E97  2571 10$:
        0608'CF         DD  0E97  2572          PUSHL   LOG_RAB+RAB$L_STV       ; Get the specific error
        0604'CF         DD  0E9B  2573          PUSHL   LOG_RAB+RAB$L_STS
                        0C  11  0E9F  2574          BRB     ALL_ERROR               ; Rest of code is same for FAB & RAB
                        0EA1  2575
                        0EA1  2576
                        0EA1  2577
                        0EA1  2578 FAB_ERROR:
              01 50     E9  0EA1  2579          BLBC    R0,10$                  ; If there is no error...
                            05  0EA4  2580          RSB                             ; ...we take no action
                        0EA5  2581 10$:
        05B8'CF         DD  0EA5  2582          PUSHL   LOG_FAB+FAB$L_STV       ; Get the specific error
        05B4'CF         DD  0EA9  2583          PUSHL   LOG_FAB+FAB$L_STS
                        0EAD  2584          ;BRB    ALL_ERROR               ; Rest of code is same for FAB & RAB
                        0EAD  2585
                        0EAD  2586
                        0EAD  2587
                        0EAD  2588 ALL_ERROR:
            50          DD  0EAD  2589          PUSHL   R0                      ; Register must be preserved
                        0EAF  2590          $FAO_S  CTRSTR = RMS_ERRMSG,-   ; RMS error message
                        0EAF  2591                  OUTLEN = BUFFER_PTR,-
                        0EAF  2592                  OUTBUF = FAO_BUF,-
                        0EAF  2593                  P1     = #LOG_FILE_DESC
            50 8ED0     0EC8  2594          POPL    R0                      ; Register must be preserved
        0481'CF         DF  0ECB  2595          PUSHAL  BUFFER_PTR
            01          DD  0ECF  2596          PUSHL   #1                      ; Arg count
  00741132 8F           DD  0ED1  2597          PUSHL   #UETP$_TEXT!STS$K_ERROR ; Message code
00000000'GF             05  FB  0ED7  2598          CALLS   #5,G^LIB$SIGNAL         ; Output - note that R0 is preserved
                            05  0EDE  2599          RSB                             ; Return with R0 intact
                        0EDF  2600
                        0EDF  2601          .END    UETLOCK00
```

```
$$.TAB                  = 00000690 R      03
$$.TABEND               = 000006D4 R      03
$$.TMP                  = 00000000
$$.TMP1                 = 00000002
$$.TMP2                 = 000000CF
$$.TMPX                 = 00000000 R      04
$$.TMPX1                = 0000000A
$$ARGS                  = 00000004
$$T1                    = 00000000
$$T2                    = 00000004
ACC$K_TERMLEN           = 00000054
ACC$L_FINALSTS          = 00000004
ACC$W_MSGTYP            = 00000000
ALL_ERROR                 00000EAD R      05
ALL_PROCS                 000002A8 R      02
ARG_COUNT                 00000500 R      03
BASE_PRI                  000000CE R      03
BAST_CEF                = 00000041
BAST_CEF_V              = 00000001
BAST_ERRMSG               000003D7 R      02
BEGIN_MSGM              = 00000004
BEGIN_MSGV              = 00000002
BEGUN_ADDR                0000022C R      02
BLKAST_M                = 00000001
BLKAST_V                = 00000000
BLOCK_AST                 00000807 R      05
BUFFER                    00000489 R      03
BUFFER_PTR                00000481 R      03
CASTPAR_ERRMSG            00000728 R      02
CASTSYNCH_ERRMSG          00000465 R      02
CAST_ERRMSG               000003AC R      02
CCASTHAND                 00000B9D R      05
CHECK_UP                  000005A7 R      05
CHF$L_SIGARGLST         = 00000004
CHF$L_SIG_ARG1          = 00000008
CHF$L_SIG_ARGS          = 00000000
CHF$L_SIG_NAME          = 00000004
CHK_BLOCKAST              000005EF R      05
CHK_CAST                  000005E1 R      05
CHK_DEADLOCK              00000763 R      05
CHK_LKSB                  000007B9 R      05
CHK_NOQUEUE               0000072A R      05
CHK_SS                    0000079C R      05
CHK_SYNCSTS               00000685 R      05
CHK_VALBLK                0000060A R      05
CMP_VAL                 = 00000042
CMP_VAL_V               = 00000002
CNTRLCMSG                 000002BB R      02
COMMAND_SIZE            = 00000008
COMMON                    00000B64 R      05
COMP_AST                  000007D6 R      05
COPY_LOG_MSG              0000090C R      02
CREATE_PROCS              000002FA R      05
CRMODE_CODE               00000A19 R      02
CWMODE_CODE               00000A1C R      02
C_AST_M                 = 00000001
C_AST_V                 = 00000000
DEADLK_ERRMSG             0000068E R      02
DEADLK_M                = 00000010
DEADLK_V                = 00000004
DEQ                     = 00000002
DEQ$_ACMODE             = 0000000C
DEQ$_FLAGS              = 00000010
DEQ$_LKID               = 00000004
DEQ$_NARGS              = 00000004
DEQ$_VALBLK             = 00000008
DEQALL_ERRMSG             00000651 R      02
DEQLST                    0000006F R      03
DEQS                      0000051C R      05
DEQ_CODE                  000009F5 R      02
DETPRC_ERRMSG             00000852 R      02
DEV$V_MBX               = 00000014
DEV$V_TRM               = 00000002
DIB$K_LENGTH            = 00000074
DIB$W_UNIT              = 0000000C
DIBBJF                    000000F2 R      03
DIBBUF_DESC               000000EA R      03
DLDET_CEF               = 00000044
DLDET_CEF_V             = 00000004
DLMASK                  = 00000030
DLOCK_AST                 0000083F R      05
DLOCK_TIME                000000A6 R      03
DLOCK_TO_AST              00000903 R      05
DLPRC_M                 = 00000002
DLPRC_V                 = 00000001
DLRES_CEF               = 00000045
DONE_CEF                = 00000040
DRIVEN                    000002B8 R      05
DRIVEN_DESC               00000000 R      03
DRIVEN_M                = 00000001
DRIVEN_V                = 00000000
DRIVER                    000001A9 R      05
DUMP_COMMAND              0000052F R      05
DUMP_M                  = 00000002
DUMP_MSG                  00000969 R      02
DUMP_V                  = 00000001
DVI$_DEVNAM             = 00000020
EF_STATE                  000001E1 R      03
ENDTEST                 = 00000003
END_ADDR                  00000232 R      02
ENQ                     = 00000000
ENQ$_ACMODE             = 00000028
ENQ$_ASTADR             = 0000001C
ENQ$_ASTPRM             = 00000020
ENQ$_BLKAST             = 00000024
ENQ$_EFN                = 00000004
ENQ$_FLAGS              = 00000010
ENQ$_LKMODE             = 00000008
ENQ$_LKSB               = 0000000C
ENQ$_NARGS              = 0000000B
ENQ$_PARID              = 00000018
ENQ$_PROT               = 0000002C
ENQ$_RESNAM             = 00000014
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                000001FE 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$SIGNAL              ********   X    05
LKID_ADDR               000001ED R      03
LKSB$                   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$WFLAND                      ********  GX    05
TABLE_END                       00000218  R     02
TEN_SECONDS                     00000271  R     02
TERMBX_ERRMSG                   000007E6  R     02
TEST_CODES                      000009DE  R     02
TEST_NAME_C                     00000010  R     03
TEST_NAME_D                     00000008  R     03
TEST_NAME_I                     00000011  R     03
TEST_NAME_LEN                 = 0000000A
TEST_TABLE                      00000000  R     02
TEXT_BUFFER                   = 00000C84
THREE_MIN                       00000279  R     02
TIME_OUT                        00000A22  R     05
TO_BE_FILLED                    0000021C  R     02
TSTFLG                          0000008B  R     03
TST_COMMAND                     00000083  R     03
TTCHAN                          000001D6  R     03
TYPE                            000001DA  R     03
TYPE_INTRO                      00000E34  R     05
TYPE_LOG_FILES                  00000D55  R     05
UETLOCK00                       00000000  RG    05
UETP                          = 00740000
UETP$_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
_255DUA28:[SHRLIB]UETP.MLB;1	1
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[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
```