```
UUU          UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP
UUU          UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP
UUU          UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP
UUU          UUU  EEE                    TTT         PPP          PPP
UUU          UUU  EEE                    TTT         PPP          PPP
UUU          UUU  EEE                    TTT         PPP          PPP
UUU          UUU  EEE                    TTT         PPP          PPP
UUU          UUU  EEE                    TTT         PPP          PPP
UUU          UUU  EEE                    TTT         PPP          PPP
UUU          UUU  EEEEEEEEEEEE           TTT         PPPPPPPPPPPP
UUU          UUU  EEEEEEEEEEEE           TTT         PPPPPPPPPPPP
UUU          UUU  EEEEEEEEEEEE           TTT         PPPPPPPPPPPP
UUU          UUU  EEE                    TTT         PPP
UUU          UUU  EEE                    TTT         PPP
UUU          UUU  EEE                    TTT         PPP
UUU          UUU  EEE                    TTT         PPP
UUU          UUU  EEE                    TTT         PPP
UUU          UUU  EEE                    TTT         PPP
UUU          UUU  EEE                    TTT         PPP
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT         PPP
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT         PPP
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT         PPP
```

FILEID**UETLPAK00


```
UU      UU  EEEEEEEEEE  TTTTTTTTTT  LL          PPPPPPPP      AAAAAA    KK      KK    000000      000000
UU      UU  EEEEEEEEEE  TTTTTTTTTT  LL          PPPPPPPP      AAAAAA    KK      KK    000000      000000
UU      UU  EE              TT      LL          PP      PP    AA    AA  KK      KK  00      00  00      00
UU      UU  EE              TT      LL          PP      PP    AA    AA  KK      KK  00      00  00      00
UU      UU  EE              TT      LL          PP      PP    AA    AA  KK    KK    00    0000  00    0000
UU      UU  EE              TT      LL          PP      PP    AA    AA  KK    KK    00    0000  00    0000
UU      UU  EEEEEEEE        TT      LL          PPPPPPPP      AA    AA  KKKKKK      00  00  00  00  00  00
UU      UU  EEEEEEEE        TT      LL          PPPPPPPP      AA    AA  KKKKKK      00  00  00  00  00  00
UU      UU  EE              TT      LL          PP            AAAAAAAAAA  KK    KK    0000    00  0000    00
UU      UU  EE              TT      LL          PP            AAAAAAAAAA  KK    KK    0000    00  0000    00
UU      UU  EE              TT      LL          PP            AA    AA  KK      KK  00      00  00      00      ....
UU      UU  EE              TT      LL          PP            AA    AA  KK      KK  00      00  00      00      ....
UUUUUUUUUU  EEEEEEEEEE      TT      LLLLLLLLLL  PP            AA    AA  KK      KK    000000      000000      ....
UUUUUUUUUU  EEEEEEEEEE      TT      LLLLLLLLLL  PP            AA    AA  KK      KK    000000      000000      ....

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

Table of contents

```
0000     1          .TITLE UETLPAK00 VAX/VMS UETP DEVICE TEST FOR THE LPA11-K
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 ;++
0000    30 ; FACILITY:
0000    31 ;       This module will be distributed with VAX/VMS under the [SYSTEST]
0000    32 ;       account.
0000    33 ;
0000    34 ; ABSTRACT:
0000    35 ;       This program tests an LPA11-K on a VAX/VMS system in all the LPA11-K's
0000    36 ;       modes of operation.
0000    37 ;
0000    38 ; ENVIRONMENT:
0000    39 ;       This program will run in user access mode, with ASTs enabled except
0000    40 ;       during error processing.  It is dependent on the LPA11-K microcode
0000    41 ;       loader process ([SYSMGR]LPA11STRT.COM) having already been started.
0000    42 ;       This program requires the following privileges and quotas:
0000    43 ;               ALTPRI (Not normally used.  See functional spec.)
0000    44 ;
0000    45 ;--
0000    46 ;
0000    47 ; AUTHOR: Richard Holstein,     CREATION DATE: August, 1981
0000    48 ;
0000    49 ; MODIFIED BY:
0000    50 ;
0000    51 ;       V03-004 RNH0007         Richard N. Holstein,    15-Feb-1984
0000    52 ;               Take advantage of the new UETP message codes.  Fix SSERROR
0000    53 ;               interaction with RMS_ERROR.
0000    54 ;
0000    55 ;       V03-003 RNH0006         Richard N. Holstein,    19-Dec-1983
0000    56 ;               Give correct sentinels to Test Controller.
0000    57 ;
```

```
0000     58 ;        V03-002 RNH0005         Richard N. Holstein,    10-Mar-1983
0000     59 ;                Allow for longer device names.  Be more cautious when boosting
0000     60 ;                priority.  Reload original microcode, if any.  Don't signal
0000     61 ;                ending message in EXIT_HANDLER.
0000     62 ;
0000     63 ;        V03-001 RNH0004         Ric ard N. Holstein,    15-Oct-1982
0000     64 ;                Miscellaneous fixes listed in the V3B UETP Workplan.
0000     65 ;
0000     66 ;        V02-004 MTR0001         Michael T. Rhodes,      17-Mar-1982
0000     67 ;                Fix continuation of .ASCID string defined at CONFIG_MSG:
0000     68 ;
0000     69 ;        V02-003 RNH0003         Richard N. Holstein,    24-Dec-1981
0000     70 ;                Miscellaneous cleanup.
0000     71 ;
0000     72 ;        V02-002 RNH0002         Richard N. Holstein,    12-Oct-1981
0000     73 ;                Do multiuser testing asynchronously to prevent buffer overruns
0000     74 ;                on fully laden LPA11-Ks.  Set SETPRI so we can change priority.
0000     75 ;
0000     76 ;        V02-001 RNH0001         Richard N. Holstein,    28-Aug-1981
0000     77 ;                Final testing and development for Field Test 1.
0000     78 ;
0000     79 ;**
```

```
                     0000     81          .SBTTL  Declarations
                     0000     82 ;
                     0000     83 ; INCLUDE FILES:
                     0000     84 ;
                     0000     85 ;        SYS$LIBRARY:LIB.MLB     for general definitions
                     0000     86 ;        SHRLIB$:UETP.MLB        for UETP definitions
                     0000     87
                     0000     88 ;
                     0000     89 ; MACROS:
                     0000     90 ;
                     0000     91          $CHFDEF                         ; Condition handler frame definitions
                     0000     92          $DEVDEF                         ; Device definitions
                     0000     93          $DIBDEF                         ; Device Information Block
                     0000     94          $DVIDEF                         ; $GETDVI ITMLST item codes
                     0000     95          $JPIDEF                         ; Job/process info
                     0000     96          $LADEF                          ; LPA11-K specific
                     0000     97          $PRVDEF                         ; Privileges
                     0000     98          $SHRDEF                         ; Shared messages
                     0000     99          $SSDEF                          ; System Service status codes
                     0000    100          $STSDEF                         ; Status return
                     0000    101          $UETUNTDEF                      ; UETP unit block offset definitions
                     000C    102          $UETPDEF                        ; UETP
                     0000    103 ;
                     0000    104 ; EQUATED SYMBOLS:
                     0000    105 ;
                     0000    106 ;   Facility number definitions:
00000001             0000    107              RMS$_FACILITY = 1
                     0000    108
                     0000    109 ;   SHR message definitions:
00740000             0000    110              UETP = UETP$_FACILITY@STS$V_FAC_NO ; Define the UETP facility code
007410E0             0000    111              UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
00741038             0000    112              UETP$_BEGIND = UETP!SHR$_BEGIND
00741080             0000    113              UETP$_ENDEDD = UETP!SHR$_ENDEDD
00741098             0000    114              UETP$_OPENIN = UETP!SHR$_OPENIN
00741130             0000    115              UETP$_TEXT   = UETP!SHR$_TEXT
                     0000    116
                     0000    117 ;   Internal flag bits...:
00000001             0000    118              TEST_OVERV   = 1                ; Set when test is over
00000002             0000    119              SAFE_TO_UPDV = 2                ; Set if it's safe to update UETINIDEV
00000003             0000    120              BEGIN_MSGV   = 3                ; Set if "BEGIN" msg has been printed
00000004             0000    121              ONE_SHOTV    = 4                ; Set if running in one-shot mode
00000005             0000    122              DUMP_MODEV   = 5                ; Set if additional error info printed
                     0000    123 ;   ...and corresponding masks:
00000002             0000    124              TEST_OVERM   = 1@TEST_OVERV
00000004             0000    125              SAFE_TO_UPDM = 1@SAFE_TO_UPDV
00000008             0000    126              BEGIN_MSGM   = 1@BEGIN_MSGV
00000010             0000    127              ONE_SHOTM    = 1@ONE_SHOTV
00000020             0000    128              DUMP_MODEM   = 1@DUMP_MODEV
                     0000    129
                     0000    130 ;   Miscellany:
00000020             0000    131              LC_BITM         = ^X20          ; Mask to convert lower case to upper
00000028             0000    132              REC_SIZE        = 40            ; UETINIDEV.DAT record size
0000012C             0000    133              TEXT_BUFFER     = 300           ; Internal text buffer size
00000004             0000    134              EFN2            = 4             ; EFN used for three minute timer
00000003             0000    135              SS_SYNCH_EFN    = 3             ; Synch miscellaneous system services
0000000F             0000    136              MAX_PROC_NAME   = 15            ; Longest possible process name
0000000A             0000    137              MAX_DEV_DESIG   = 10            ; Longest possible controller name
```

```
00000005  0000   138          MAX_UNIT_DESIG= 5                ; Longest possible unit number
```

```
                     0000   140 ;   LPA11-K specific definitions:
00000000             0000   141         JOB_PRIORITY  = 0                 ; Priority to run LPA11-K test if...
                     0000   142                                           ; ...(.GT. 0) and (.GT. base priority)
00000005             0000   143         JPI_EFN       = EFN2+1            ; EFN for $GETJPI completion
B7173A51             0000   144         CLOCK_SPEED   = ^F0.0002          ; LPA11-K clock A rate
00000032             0000   145         MGT_BUF_SIZE  = 50                ; Size of channel status area
00000007             0000   146         NUM_CHANNELS  = 7                 ; Channels to test on LPA11-K I/O bus
000000A3             0000   147         LA_K_OVERRUN  = ^O243             ; LPA11-K error code for data overrun
00000000             0000   148         SETIBF        = 0                 ; Code for LPA$SETIBF in ONCE_FOR_EACH
00000001             0000   149         SWEEP         = 1                 ; Code for sweep routines in ONCE_FOR_EACH
00000002             0000   150         RLSBUF        = 2                 ; Code for LPA$RLSBUF in ONCE_FOR_EACH
00000002             0000   151         LAST_FUNCTION = 2                 ; Maximum ONCE_FOR_EACH code value
00000018             0000   152         MU_TIME_OUT   = 24                ; Maximum times we'll poll SWEEP_COUNT...
                     0000   153                                           ; ...to see if multiuser sweeps have...
                     0000   154                                           ; ...finished. We actually wait for...
                     0000   155                                           ; ...MU_TIME_OUT*FIVE_SECONDS
                     0000   156
00000800             0000   157         WRITE_SIZE    = 2048              ; Data buffer size (write/read)
                     0000   158 .IIF NE WRITE_SIZE&1, .ERROR 0-
                     0000   159                                           ; WRITE_SIZE must be even for dedicated sweeps
                     0000   160 .IIF LT WRITE_SIZE-258, .ERROR 0-
                     0000   161                                           ; WRITE_SIZE must be .GE. 258 for dedicated transfers
                     0000   162 .IIF GE <WRITE_SIZE*NUM_CHANNELS*2>-32768, .ERROR 0-
                     0000   163                                           ; Aggregate buffer size exceeds 32768
                     0000   164
00000800             0000   165         RAMP_HEIGHT   = WRITE_SIZE       ; Size of AA11-K ramp data buffers
00000000             0000   166         DEVDEP_SIZE   = 0                 ; Device dependent part size of node
00000000             0000   167         BUFFER_SIZE   = 0                 ; We don't use this - allocated statically
                     0000   168
                     0000   169         PAGES = <<UETUNT$C_INDSIZ+-       ; Add together all of the pieces...
                     0000   170                  DEVDEP_SIZE+-            ; ...which make up a UETP unit block...
                     0000   171                  BUFFER_SIZE+-            ; ...to give to the $EXPREG service...
00000001             0000   172                  511>/512>                ; ...later
                     0000   173
                     0000   174 ;++
                     0000   175 ;       In order to allow for word size displacement when linking this image,
                     0000   176 ;       and to make for a nice layout of the program text, some mucking about
                     0000   177 ;       with .PSECTs has been done.  In particular, the BUFFERS .PSECT, because
                     0000   178 ;       of its attributes, is the first .PSECT to be loaded into virtual
                     0000   179 ;       memory, allowing word displacement to other .PSECTs (i.e., longword
                     0000   180 ;       displacement addressing mode is only needed to reference items in the
                     0000   181 ;       BUFFERS .PSECT).  BE CAREFUL when changing anything which might change
                     0000   182 ;       the order in which .PSECTs are loaded!
                     0000   183 ;--
```

```
                                0000   185          .SBTTL  Read-Only Data
                            00000000   186          .PSECT  RODATA,EXE,NOWRT,PAGE   ; EXE attribute changes .PSECT ordering!
                                0000   187
                                0000   188 ACNT_NAME:                               ; Process name on exit
53 45 54 53 59 53 00000008'010E0000'   0000   189          .ASCID  /SYSTEST/
                                  54   000E
                                000F   190
                                000F   191 TEST_NAME:                               ; This test name
41 50 4C 54 45 55 00000017'010E0000'   000F   192          .ASCID  /UETLPAK00/
                              30 30 4B   001D
                                0020   193
                                0020   194 SUPDEV_GBLSEC:                          ; How we access UETSUPDEV.DAT
50 55 53 54 45 55 00000028'010E0000'   0020   195          .ASCID  /UETSUPDEV/
                              56 45 44   002E
                                0031   196
                                0031   197 CONTROLLER:                             ; Logical name of controller
41 4E 4C 52 54 43 00000039'010E0000'   0031   198          .ASCID  /CTRLNAME/
                                 45 4D   003F
                                0041   199
                                0041   200 MODE:                                   ; Run mode logical name
      45 44 4F 4D 00000049'010E0000'   0041   201          .ASCID  /MODE/
                                004D   202
                                004D   203 NO_RMS_AST_TABLE:                       ; List of errors for which...
                        00000000'      004D   204          .LONG   RMS$_BLN        ; ...RMS cannot deliver an AST...
                        00000000'      0051   205          .LONG   RMS$_BUSY       ; ...even if one has an ERR= arg
                        00000000'      0055   206          .LONG   RMS$_CDA        ; Note that we can search table...
                        00000000'      0059   207          .LONG   RMS$_FAB        ; ...via MATCHC since <31:16>...
                        00000000'      005D   208          .LONG   RMS$_RAB        ; ...pattern can't be in <15:0>
                        00000014       0061   209 NRAT_LENGTH = .-NO_RMS_AST_TABLE
                                0061   210
                                0061   211 SYS$INPUT:                              ; Name of device from which...
4E 49 24 53 59 53 00000069'010E0000'   0061   212          .ASCID  /SYS$INPUT/     ; ...the test can be aborted
                              54 55 50   006F
                                0072   213
                                0072   214 INPUT_ITMLST:                           ; $GETDVI arg list for SYS$INPUT
                          0020 0040    0072   215          .WORD   64,DVI$_DEVNAM  ; We need the equivalence name
                  0000000C'00000014'   0076   216          .LONG   BUFFER,BUFFER_PTR
                          00000000     007E   217          .LONG   0               ; Terminate the list
                                0082   218
                                0082   219 CS1:                                   ; Device class and type control string
21 20 42 58 32 21 0000008A'010E0000'   0082   220          .ASCID  /!2XB !2XB /
                              20 42 58 32   0090
                                0094   221
                                0094   222 CS3:                                   ; Device class-only control string
2A 20 42 58 32 21 0000009C'010E0000'   0094   223          .ASCID  /!2XB **/
                                 2A   00A2
                                00A3   224
                                00A3   225 CNTRLCMSG:
65 74 72 6F 62 41 000000AB'010E0000'   00A3   226          .ASCID  \Aborted via a user CTRL/C\
72 65 73 75 20 61 20 61 69 76 20 64   00B1
                  43 2F 4C 52 54 43 20   00BD
                                00C4   227
                                00C4   228 NO_CTRLNAME:
6E 6F 63 20 6F 4E 000000CC'010E0000'   00C4   229          .ASCID  /No controller specified./
63 65 70 73 20 72 65 6C 6C 6F 72 74   00D2
                  2E 64 65 69 66 69   00DE
                                00E4   230
```

```
                                00E4   231 DEAD_CTRLNAME:
20 74 27 6E 61 43 00000EC'010E0000'  00E4   232         .ASCID  /Can't test controller !AS, marked as unusable in UETINIDEV.DAT./
6C 6F 72 74 6E 6F 63 20 74 73 65 74  00F2
72 61 6D 20 2C 53 41 21 20 72 65 6C  00FE
61 73 75 6E 75 20 73 61 20 64 65 6B  010A
4E 49 54 45 55 20 6E 69 20 65 6C 62  0116
         2E 54 41 44 2E 56 45 44 49  0122
                                012B   233
                                012B   234 NOUNIT_SELECTED:
69 6E 75 20 6F 4E 00000133'010E0000'  012B   235         .ASCID  /No units selected for testing./
20 64 65 74 63 65 6C 65 73 20 73 74  0139
2E 67 6E 69 74 73 65 74 20 72 6F 66  0145
                                0151   236
                                0151   237 ILLEGAL_REC:
61 67 65 6C 6C 49 00000159'010E0000'  0151   238         .ASCID  /Illegal record format in file UETINIDEV.DAT!/
72 6F 66 20 64 72 6F 63 65 72 20 6C  015F
20 65 6C 69 66 20 6E 69 20 74 61 6D  016B
41 44 2E 56 45 44 49 4E 49 54 45 55  0177
                        21 54       0183
                                0185   239
                                0185   240 PASS_MSG:
66 6F 20 64 6E 45 0000018D'010E0000'  0185   241         .ASCID  /End of pass !UL with !UL iterations at !%D./
69 77 20 4C 55 21 20 73 73 61 70 20  0193
61 72 65 74 69 20 4C 55 21 20 68 74  019F
44 25 21 20 74 61 20 73 6E 6F 69 74  01AB
                              2E    01B7
                                01B8   242
                                01B8   243 INIDEV_UPDERR:                          ; Error during exit handler
20 72 6F 72 72 45 000001C0'010E0000'  01B8   244         .ASCID  /Error updating UETINIDEV.DAT./
54 45 55 20 67 6E 69 74 61 64 70 75  01C6
   2E 54 41 44 2E 56 45 44 49 4E 49  01D2
                                01DD   245
                                01DD   246 THREEMIN:                               ; 3 minute delta time
                FFFFFFFF 94B62E00   01DD   247         .LONG   -10*1000*1000*180,-1
                                01E5   248
                                01E5   249 CONT_DESC:                              ; Descriptor used to convert controller...
                     0000 0028      01E5   250         .WORD   REC_SIZE,0           ; ...from lowercase to uppercase
                      00000014'     01E9   251         .ADDRESS BUFFER
                                01ED   252
                                01ED   253 FILE:                                   ; Fills in RMS_ERR_STRING
      65 6C 69 66 000001F5'010E0000'  01ED   254         .ASCID  /file/
                                01F9   255
                                01F9   256 RECORD:                                 ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 00000201'010E0000'  01F9   257         .ASCID  /record/
                                0207   258
                                0207   259 RMS_ERR_STRING:                         ; Announces an RMS error
41 21 20 53 4D 52 0000020F'010E0000'  0207   260         .ASCID  /RMS !AS error in file !AD/
66 20 6E 69 20 72 6F 72 72 65 20 53  0215
               44 41 21 20 65 6C 69  0221
                                0228   261
                                0228   262 PROMPT:
64 20 72 65 6C 6C 6F 72 74 6E 6F 43  0228   263         .ASCII  /Controller designation?: /
3A 3F 6E 6F 69 74 61 6E 67 69 73 65  0234
                              20    0240
                     00000019       0241   264         PMTSIZ = .-PROMPT
                                0241   265
                                0241   266 LPA11K_PRIORITY:                       ; Logical name allows user...
```

```
4B 31 31 41 50 4C 00000249'010E0000'  0241    267          .ASCID  /LPA11KPRIORITY/        ; ...to easily set job priority
                59 54 49 52 4F 49 52 50  024F
                                         0257    268
                                         0257    269 LITERAL_0:                           ; For miscellaneous subroutine calls
                              00000000   0257    270          .LONG   0
                                         025B    271
                                         025B    272 LITERAL_1:                           ; For miscellaneous subroutine calls
                              00000001   025B    273          .LONG   1
                                         025F    274
                                         025F    275 FIVE_SECONDS:                        ; 5-second delta time
                        FFFFFFFF FD050F80  025F    276          .LONG   -10000000*5,-1
                                         0267    277
                                         0267    278 NEEDED_PRIVS:                        ; Privileges needed to run our test
                        00000000 00002000  0267    279          .LONG   1@PRV$V_SETPRI,0
                                         026F    280
                                         026F    281 MU_MCODE:                            ; Each of these...
      72 65 73 75 69 74 6C 75 6D 00'     026F    282          .ASCIC  /multiuser/
                                   09    026F
                                         0279    283 AD_MCODE:                            ; ...may be used in messages...
6F 74 2D 65 75 67 6F 6C 61 6E 61 00'     0279    284          .ASCIC  /analogue-to-digital/
            6C 61 74 69 67 69 64 2D      0285
                                   13    0279
                                         028D    285 DA_MCODE:                            ; ...to distinguish the mcode...
2D 6F 74 2D 6C 61 74 69 67 69 64 00'     028D    286          .ASCIC  /digital-to-analogue/
            65 75 67 6F 6C 61 6E 61      0299
                                   13    028D
                                         02A1    287 FO_MCODE:                            ; ...or section of the test
72 65 76 6F 20 64 65 63 72 6F 66 00'     02A1    288          .ASCIC  /forced overrun/
                              6E 75 72   02AD
                                   0E    02A1
                                         02B0    289
                                         02B0    290 LOAD_BAD_ERR_MSG:                    ; Bad error during microcode loading
20 72 6F 72 72 45 000002B8'010E0000'     02B0    291          .ASCID  \Error loading !AC microcode,!/!_LPA11-K Status Out: %O!OB,\-
20 43 41 21 20 67 6E 69 64 61 6F 6C      02BE
2F 21 2C 65 64 6F 63 6F 72 63 69 6D      02CA
74 53 20 4B 2D 31 31 41 50 4C 5F 21      02D6
4F 25 20 3A 74 75 4F 20 73 75 74 61      02E2
                              2C 42 4F 21  02EE
74 75 4F 20 6C 6F 72 74 6E 6F 43 20      02F2    292                  \ Control Out: %O!OB,!/!_Maint Status high,low: %O!OB,%O!OB.\
5F 21 2F 21 2C 42 4F 21 4F 25 20 3A      02FE
73 75 74 61 74 53 20 74 6E 69 61 4D      030A
25 20 3A 77 6F 6C 2C 68 67 69 68 20      0316
   2E 42 4F 21 4F 25 2C 42 4F 21 4F      0322
                                         032D    293
                                         032D    294 LOAD_MISC_ERR_MSG:                   ; Misc error during microcode loading
20 72 6F 72 72 45 0C000335'010E0000'     032D    295          .ASCID  \Error loading !AC microcode.\
20 43 41 21 20 67 6E 69 64 61 6F 6C      033B
      2E 65 64 6F 63 6F 72 63 69 6D      0347
                                         0351    296
                                         0351    297 BADRATE_ERR_MSG:                     ; Error calculating clock A rate
20 74 27 6E 61 43 00000359'010E0000'     0351    298          .ASCID  /Can't compute clock A rate and preset value./
63 6F 6C 63 20 65 74 75 70 6D 6F 63      035F
64 6E 61 20 65 74 61 72 20 41 20 6B      036B
75 6C 61 76 20 74 65 73 65 72 70 20      0377
                              2E 65      0383
                                         0385    299
                                         0385    300 SETUP_ERR_MSG:                       ; Error setting up buffers
```

```
20 72 6F 72 72 45 0000038D'010E0000' 0385   301          .ASCID  /Error setting up !AC data buffers./
21 20 70 75 20 67 6E 69 74 74 65 73  0393
66 66 75 62 20 61 74 61 64 20 43 41  039F
                     2E 73 72 65     03AB
                                     03AF   302
                                     03AF   303 RELEASE_ERR_MSG:                        ; Error releasing buffers
20 72 6F 72 72 45 000003B7'010E0000' 03AF   304          .ASCID  /Error releasing !AC data buffers./
41 21 20 67 6E 69 73 61 65 6C 65 72  03BD
65 66 66 75 62 20 61 74 61 64 20 43  03C9
                        2E 73 72     03D5
                                     03D8   305
                                     03D8   306 SWEEP_ERR_MSG:                          ; Data not passed correctly
20 72 6F 72 72 45 000003E0'010E0000' 03D8   307          .ASCID  /Error starting up !AC sweeps./
20 70 75 20 67 6E 69 74 72 61 74 73  03E6
   2E 73 70 65 65 77 73 20 43 41 21  03F2
                                     03FD   308
                                     03FD   309 IWTBUF_MISC_ERR_MSG:                    ; Miscellaneous error
20 72 6F 72 72 45 00000405'010E0000' 03FD   310          .ASCID  /Error returning from !AC data transfer./
72 66 20 67 6E 69 6E 72 75 74 65 72  040B
20 61 74 61 64 20 43 41 21 20 6D 6F  0417
         2E 72 65 66 73 6E 61 72 74  0423
                                     042C   311
                                     042C   312 IWTBUF_BAD_ERR_MSG:                    ; Error code with add'l data in IOSB
20 72 6F 72 72 45 00000434'010E0000' 042C   313          .ASCID  \Error during !AC data transfer,!/!_LPA11-K Status Out: %O!OB,\-
64 20 43 41 21 20 67 6E 69 72 75 64  043A
72 65 66 73 6E 61 72 74 20 61 74 61  0446
4B 2D 31 31 41 50 4C 5F 21 2F 21 2C  0452
3A 74 75 4F 20 73 75 74 61 74 53 20  045E
                  2C 42 4F 21 4F 25  046A
74 75 4F 20 6C 6F 72 74 6E 6F 43 20  0471   314                  \ Control Out: %O!OB,!/!_Maint Status high,low: %O!OB,%O!OB.\
5F 21 2F 21 2C 42 4F 21 4F 25 20 3A  047D
73 75 74 61 74 53 20 74 6E 69 61 4D  0489
25 20 3A 77 6F 6C 2C 68 67 69 68 20  0495
   2E 42 4F 21 4F 25 2C 42 4F 21 4F  04A1
                                     04AC   315
                                     04AC   316 OVERRUN_ERR_MSG:                        ; Unexpected buffer overrun/underrun
20 72 6F 72 72 45 000004B4'010E0000' 04AC   317          .ASCID  \Error during !AC data transfer,!/!_LPA11-K Status Out: %O!OB,\-
64 20 43 41 21 20 67 6E 69 72 75 64  04BA
72 65 66 73 6E 61 72 74 20 61 74 61  04C6
4B 2D 31 31 41 50 4C 5F 21 2F 21 2C  04D2
3A 74 75 4F 20 73 75 74 61 74 53 20  04DE
                  2C 42 4F 21 4F 25  04EA
74 75 4F 20 6C 6F 72 74 6E 6F 43 20  04F1   318                  \ Control Out: %O!OB,!/!_Maint Status high,low: %O!OB,\-
5F 21 2F 21 2C 42 4F 21 4F 25 20 3A  04FD
73 75 74 61 74 53 20 74 6E 69 61 4D  0509
25 20 3A 77 6F 6C 2C 68 67 69 68 20  0515
                     2C 42 4F 21 4F  0521
66 75 42 09 2F 21 2E 42 4F 21 4F 25  0526   319                  \%O!OB.!/       Buffer overrun/underrun.  Try retesting with\-
2F 6E 75 72 72 65 76 6F 20 72 65 66  0532
54 20 20 2E 6E 75 72 72 65 64 6E 75  053E
67 6E 69 74 73 65 74 65 72 20 79 72  054A
                     68 74 69 77 20  0556
20 62 6F 6A 20 72 65 68 67 69 68 20  055B   320                  \ higher job priority.\
         2E 79 74 69 72 6F 69 72 70  0567
                                     0570   321
                                     0570   322 CLOCK_ERR_MSG:                          ; Clock didn't start correctly
20 72 6F 72 72 45 00000578'010E0000' 0570   323          .ASCID  /Error starting clock during !AS test./
```

```
6F 6C 63 20 67 6E 69 74 72 61 74 73  057E
41 21 20 67 6E 69 72 75 64 20 6B 63  058A
                2E 74 73 65 74 20 53  0596
                                      059D   324
                                      059D   325 AD11K_ERR_MSG:                       ; Incorrect a-to-d setup
20 72 6F 72 72 45 000005A5'010E0000'  059D   326         .ASCID  /Error setting !AC AD11-K channel info./
20 43 41 21 20 67 6E 69 74 74 65 73   05AB
6E 6E 61 68 63 20 4B 2D 31 31 44 41   05B7
                2E 6F 66 6E 69 20 6C 65  05C3
                                      05CB   327
                                      05CB   328 NO_OVRN_ERR_MSG:                    ; Couldn't force data overrun
20 64 6C 75 6F 43 000005D3'010E0000'  05CB   329         .ASCID  /Could not detect data overrun./
64 20 74 63 65 74 65 64 20 74 6F 6E   05D9
2E 6E 75 72 72 65 76 6F 20 61 74 61   05E5
                                      05F1   330
                                      05F1   331 NAKED_ERR_MSG:                      ; No LPA11-K peripherals outboard
76 65 64 20 6F 4E 000005F9'010E0000'  05F1   332         .ASCID  \No devices found on LPA11-K input/output bus.\
6F 20 64 6E 75 6F 66 20 73 65 63 69   05FF
6E 69 20 4B 2D 31 31 41 50 4C 20 6E   060B
62 20 74 75 70 74 75 6F 2F 74 75 70   0617
                        2E 73 75      0623
                                      0626   333
                                      0626   334 LALOADER_PROC:                      ; System mcode loader process name
44 41 4F 4C 41 4C 0000062E'010E0000'  0626   335         .ASCID  /LALOADER/
                              52 45   0634
                                      0636   336
                                      0636   337 LALOADER_IMAGE:                     ; System mcode loader process image
45 58 45 53 59 53 0000063E'010E0000'  0636   338         .ASCID  /SYSEXE]LALOADER.EXE/
58 45 2E 52 45 44 41 4F 4C 41 4C 5D   0644
                                 45   0650
                                      0651   339
                                      0651   340 NO_LALOADER:                       ; Warns of incorrect system setup
79 73 20 65 68 54 00000659'010E0000'  0651   341         .ASCID  \The system LPA11-K microcode loader process wasn't started,\-
4B 2D 31 31 41 50 4C 20 6D 65 74 73   065F
6C 20 65 64 6F 63 6F 72 63 69 6D 20   066B
73 65 63 6F 72 70 20 72 65 64 61 6F   0677
61 74 73 20 74 27 6E 73 61 77 20 73   0683
                        2C 64 65 74 72  068F
75 6F 63 09 0A 0D 2C 2E 65 2E 69 20   0694   342                 \ i.e.,\<13><10>\       couldn't see process LALOADER with\-
72 70 20 65 65 73 20 74 27 6E 64 6C   06A0
44 41 4F 4C 41 4C 20 73 73 65 63 6F   06AC
                  68 74 69 77 20 52 45  06B8
72 20 5D 34 2C 31 5B 3D 43 49 55 20   06BF   343                 \ UIC=[1,4] running\<13><10>-
                  0D 67 6E 69 6E 6E 75  06CB
20 65 67 61 6D 69 20 65 68 74 09 0A   06D2   344                 \       the image SYS$SYSTEM:LALOADER.EXE.\
4C 3A 4D 45 54 53 59 53 24 53 59 53   06DE
2E 45 58 45 2E 52 45 44 41 4F 4C 41   06EA
                                      06F6   345
                                      06F6   346 BAD_CASE_MSG:                      ; Impossible condition in ONCE_FOR_EACH
6E 72 65 74 6E 49 000006FE'010E0000'  06F6   347         .ASCID  /Internal consistency error:  !UB dispatch code./
6E 65 74 73 69 73 6E 6F 63 20 6C 61   0704
21 20 20 3A 72 6F 72 72 65 20 79 63   0710
20 68 63 74 61 70 73 69 64 20 42 55   071C
                        2E 65 64 6F 63  0728
                                      072D   348
                                      072D   349 COUNTED_OUT_MSG:                    ; Some multiuser sweep(s) hung
20 64 65 6D 69 54 00000735'010E0000'  072D   350         .ASCID  /Timed out waiting for multiuser sweeps to complete./
```

```
20 67 6E 69 74 69 61 77 20 74 75 6F  073B
65 73 75 69 74 6C 75 6D 20 72 6F 66  0747
20 6F 74 20 73 70 65 65 77 73 20 72  0753
         2E 65 74 65 6C 70 6D 6F 63  075F
                                     0768   351
                                     0768   352 CONFIG_MSG:                                ; Gives LPA11-K I/O bus configuration
2D 31 31 41 50 4C 00000770'010E0000' 0768   353         .ASCID  \LPA11-K configuration:  !UB AD11-K!AC, !UB AA11-K!AC, !UB \-
74 61 72 75 67 69 66 6E 6F 63 20 4B  0776
44 41 20 42 55 21 20 20 3A 6E 6F 69  0782
42 55 21 20 2C 43 41 21 4B 2D 31 31  078E
20 2C 43 41 21 4B 2D 31 31 41 41 20  079A
                  20 42 55 21        07A6
09 2F 21 43 41 21 4B 2D 31 31 52 44  07AA   354 \DR11-K!AC!/    Intersample time will be !AS seconds.\
20 65 6C 70 6D 61 73 72 65 74 6E 49  07B6
65 62 20 6C 6C 69 77 20 65 6D 69 74  07C2
73 64 6E 6F 63 65 73 20 53 41 21 20  07CE
                              2E     07DA
                                     07DB   355
                                     07DB   356 S_MSG:                                   ; This because $FAO !%S is too smart
                        73 27 00'    07DB   357         .ASCIC  \'s\
                              02     07DB
                                     07DE   358 NULL_MSG:                                ; Alternative to S_MSG
                              00'    07DE   359         .ASCIC  \\
                              00     07DE
                                     07DF   360
                                     07DF   361 MU_BEG_MSG:                              ; Tells when multiuser test begins
75 69 74 6C 75 4D 000007E7'010E0000' 07DF   362         .ASCID  /Multiuser test beginning./
67 65 62 20 74 73 65 74 20 72 65 73  07ED
               2E 67 6E 69 6E 6E 69  07F9
                                     0800   363
                                     0800   364 MU_END_MSG:                              ; Tells when multiuser test ends
75 69 74 6C 75 4D 00000808'010E0000' 0800   365         .ASCID  /Multiuser test ending./
64 6E 65 20 74 73 65 74 20 72 65 73  080E
                        2E 67 6E 69  081A
                                     081E   366
                                     081E   367 AD_BEG_MSG:                              ; Tells when AD11-K test begins
61 63 69 64 65 44 00000826'010E0000' 081E   368         .ASCID  /Dedicated A-to-D test beginning./
74 20 44 2D 6F 74 2D 41 20 64 65 74  082C
6E 69 6E 6E 69 67 65 62 20 74 73 65  0838
                              2E 67  0844
                                     0846   369
                                     0846   370 AD_END_MSG:                              ; Tells when AD11-K test ends
61 63 69 64 65 44 0000084E'010E0000' 0846   371         .ASCID  /Dedicated A-to-D test ending./
74 20 44 2D 6F 74 2D 41 20 64 65 74  0854
   2E 67 6E 69 64 6E 65 20 74 73 65  0860
                                     086B   372
                                     086B   373 DA_BEG_MSG:                              ; Tells when AA11-K test begins
61 63 69 64 65 44 00000873'010E0000' 086B   374         .ASCID  /Dedicated D-to-A test beginning./
74 20 41 2D 6F 74 2D 44 20 64 65 74  0879
6E 69 6E 6E 69 67 65 62 20 74 73 65  0885
                              2E 67  0891
                                     0893   375
                                     0893   376 DA_END_MSG:                              ; Tells when AA11-K test ends
61 63 69 64 65 44 0000089B'010E0000' 0893   377         .ASCID  /Dedicated D-to-A test ending./
74 20 41 2D 6F 74 2D 44 20 64 65 74  08A1
   2E 67 6E 69 64 6E 65 20 74 73 65  08AD
```

```
                        08B8   379          .SBTTL  Read/Write Data
                    00000000   380          .PSECT  RWDATA,WRT,NOEXE,PAGE   ; .PSECT name affects ordering!
                        0000   381
                        0000   382 TTCHAN:                                  ; Channel associated with ctrl. term.
                   0000 0000   383          .WORD   0
                        0002   384
                        0002   385 FLAG:                                    ; Miscellaneous flag bits
                   0000 0002   386          .WORD   0                       ; (See Equated Symbols for definitions)
                        0004   387
                        0004   388 FAO_BUF:                                 ; FAO output string descriptor
                0000 012C 0004   389          .WORD   TEXT_BUFFER,0
                 00000014' 0008   390          .ADDRESS BUFFER
                        000C   391
                        000C   392 BUFFER_PTR:                              ; Fake .ASCID buffer for misc. strings
                0000 012C 000C   393          .WORD   TEXT_BUFFER,0         ; A word for length, a word for desc.
                 00000014' 0010   394          .ADDRESS BUFFER
                        0014   395
                        0014   396 BUFFER:                                  ; FAO output and other misc. buffer
                 00000140 0014   397          .BLKB   TEXT_BUFFER
                        0140   398
                        0140   399 DEVDSC:                                  ; Device name descriptor
                0000 000A 0140   400          .WORD   MAX_DEV_DESIG,0
                 0000015F' 0144   401          .ADDRESS DEV_NAME
                        0148   402
                        0148   403 PROCESS_NAME:                            ; Process name
    4B 41 50 4C 00000150'010E0000' 0148   404          .ASCID  /LPAK/
                    0000000B 0154   405          PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
                    0000015F 0154   406          .BLKB   PROCESS_NAME_FREE
                        015F   407
                        015F   408 DEV_NAME:                                ; Device name buffer
                    0000016E 015F   409          .BLKB   MAX_DEV_DESIG+MAX_UNIT_DESIG
                    0000000F 016E   410          NAME_LEN = .-DEV_NAME
                        016E   411
                        016E   412 DIB:                                     ; Device Information Block
                0000 0074 016E   413          .WORD   DIB$K_LENGTH,0
                 00000176' 0172   414          .ADDRESS DIBBUF
                        0176   415 DIBBUF:
                    000001EA 0176   416          .BLKB   DIB$K_LENGTH
                        01EA   417
                        01EA   418 ERROR_COUNT:                             ; Cumulative error count at runtime
                    00000000 01EA   419          .LONG   0
                        01EE   420
                        01EE   421 STATUS:                                  ; Status value on program exit
                    0000000C 01EE   422          .LONG   0
                        01F2   423
                        01F2   424 QUAD_STATUS:                             ; IO status block for misc sys. svcs.
00000000 00000000       01F2   425          .QUAD   0
                        01FA   426
                        01FA   427 INADDRESS:                               ; $CRMPSC address storage
00000000 00000000       01FA   428          .LONG   0,0
                        0202   429 OUTADDRESS:
00000000 00000000       0202   430          .LONG   0,0
                        020A   431
                        020A   432 DEVNAM_LEN:                              ; Current device name length
                   0000 020A   433          .WORD   0
                        020C   434
                        020C   435 RANDOM1:                                 ; Random word #1
```

```
                AAAAAAAA  020C   436          .LONG   ^XAAAAAAAA
                          0210   437
                          0210   438 RANDOM2:                                ; Random word #2
                A72EA72E  0210   439          .LONG   ^XA72EA72E
                          0214   440
                          0214   441 ITERATION:                               ; # of times all tests were executed
                00000000  0214   442          .LONG   0
                          0218   443
                          0218   444 PASS:                                    ; Pass count
                00000000  0218   445          .LONG   0
                          021C   446
                          021C   447 MSG_BLOCK:                               ; Auxiliary $GETMSG info
                00000220  021C   448          .BLKB   4
                          0220   449
                          0220   450 EXIT_DESC:                              ; Exit handler descriptor
                00000000  0220   451          .LONG   0
                000010FF' 0224   452          .ADDRESS EXIT_HANDLER
                00000001  0228   453          .LONG   1
                000001EE' 022C   454          .ADDRESS STATUS
                          0230   455
                          0230   456 ARG_COUNT:                              ; Argument counter used by ERROR_EXIT
                00000000  0230   457          .LONG   0
                          0234   458
                          0234   459          .ALIGN QUAD                     ; For self-relative queue of unit blocks
                          0238   460
                          0238   461 UNIT_LIST:                              ; Head of unit block circular queue
    00000000 00000000     0238   462          .QUAD   0
                          0240   463
                          0240   464 NEW_NODE:                               ; Newly acquired node address
    00000000 00000000     0240   465          .QUAD   0
                          0248   466
                          0248   467 PRIB_ITMLST:                           ; Argument for $GETJPI's base priority
            0309 0004     0248   468          .WORD   4,JPI$_PRIB
             00000258'    024C   469          .ADDRESS BASE_PRIORITY
             00000000     0250   470          .LONG   0
            0000 0000     0254   471          .WORD   0,0
                          0258   472 BASE_PRIORITY:                          ; A place for our process base priority
             00000000     0258   473          .LONG   0
                          025C   474
                          025C   475 IMAGNAM_ITMLST:                         ; Argument for $GETJPI's image name
            0207 0040     025C   476          .WORD   64,JPI$_IMAGNAME
             0000026C'    0260   477          .ADDRESS MCODE_IMAGNAM
             000002AC'    0264   478          .ADDRESS MCODE_IMAGLEN
            0000 0000     0268   479          .WORD   0,0
                          026C   480 MCODE_IMAGNAM:                          ; A place for LALOADER's process name
             000002AC     026C   481          .BLKB   64
                          02AC   482 MCODE_IMAGLEN:                          ; A place to hold the process name's length
                 0000     02AC   483          .WORD   0
                          02AE   484
                          02AE   485 IMAGNAM_IOSB:                           ; Final $GETJPI status
    00000000 00000000     02AE   486          .QUAD   0
                          02B6   487
                          02B6   488          .ALIGN  LONG                    ; Required for data transfer
                          02B8   489 RAMP_DATA:                              ; Ramp data for AA11-K test
             00001288     02B8   490          .BLKW RAMP_HEIGHT               ; Forms a [1:N,0:1] word buffer
                          12B8   491 RAMP_DATA1:
             000022B8     12B8   492          .BLKW RAMP_HEIGHT
```

```
                     22B8   493
                     22B8   494 SWEEP_COUNT:                             ; Number of multiuser channels...
                00   22B8   495         .BYTE   0                        ; ...which successfully started sweeps
                     22B9   496
                     22B9   497 MCODE_TYPE:                              ; Holds mcode type constant
           00000000  22B9   498         .LONG   0
                     22BD   499
                     22BD   500 MCODE_ADDR:                              ; Holds address of mcode name
           00000000' 22BD   501         .ADDRESS 0
                     22C1   502
                     22C1   503 FLOAT_PTR:                               ; For FOR$CVT_D_TF arguments
           0000 0007 22C1   504         .WORD   7,0
            000022C9' 22C5   505         .ADDRESS .+4
            000022D0  22C9   506         .BLKB   7
                     22D0   507
                     22D0   508         .ALIGN  LONG                     ; LPA11-K requirement
                     22D0   509 TRANSFER_TABLE:                          ; Data Transfer Command Table
               010A  22D0   510         .WORD   ^X010A                   ;       mode (cf. LPA11-K User's Guide)
                 01  22D2   511         .BYTE   1                        ;       highest buffer available (0-7)
            000022D4  22D3   512         .BLKB   1                        ;       (unused)
            000022F8' 22D4   513         .ADDRESS USER_STATUS             ;       pointer to user status word
           00002000  22D8   514         .LONG   2*WRITE_SIZE*2           ;       overall data buffer length (bytes)
           00000000' 22DC   515         .ADDRESS LA_AK_BUF11             ;       overall data buffer address
           00000000  22E0   516         .LONG   0                        ;       random channel list length
           00000000' 22E4   517         .ADDRESS 0                       ;       random channel list address
               0064  22E8   518         .WORD   100                      ;       delay
                 00  22EA   519         .BYTE   0                        ;       start channel number
                 00  22EB   520         .BYTE   0                        ;       channel increment
               000A  22EC   521         .WORD   10                       ;       number of samples in sequence
               000A  22EE   522         .WORD   10                       ;       dwell
                 00  22F0   523         .BYTE   0                        ;       digital trigger channel
                 00  22F1   524         .BYTE   0                        ;       event mark channel
               FFFF  22F2   525         .WORD   ^XFFFF                   ;       digital trigger mask
               FFFF  22F4   526         .WORD   ^XFFFF                   ;       event mark mask
            000022F8  22F6   527         .BLKW   1                        ;       (unused)
                     22F8   528
                     22F8   529         .ALIGN  WORD                     ; LPA11-K requirement
                     22F8   530 USER_STATUS:                             ; User status word from transfer table
               0000  22F8   531         .WORD   0                        ; (cf. LPA11-K User's Guide, p.2-19)
                     22FA   532
                     22FA   533 AINTRVL:                                 ; Holds CLOCK A speed
   00000000 B7173A51 22FA   534         .QUAD   CLOCK_SPEED              ; "D_FLOATING" for FOR$CVT_D_TF
                     2302   535
                     2302   536 COMP_RATE:                               ; Receives computed actual CLOCK A...
   00000000 00000000 2302   537         .QUAD   0                        ; ...speed for debugging info
                     230A   538
                     230A   539 ;
                     230A   540 ; For a note concerning the names of the following items, see the text where
                     230A   541 ; the GEN_ARG_LIST macro is defined.
                     230A   542 ;
                     230A   543 LA_W_LBUF:                               ; Size in words of data buffers
               0800  230A   544         .WORD   WRITE_SIZE
                     230C   545
                     230C   546 LA_B_DRS:                                ; Number of DR11-K's on this LPA11-K
           00002310  230C   547         .BLKB   1+3                      ; (Count + filler to make PUSHL easy)
                     2310   548
                     2310   549 LA_B_ADS:                                ; Number of AD11-K's on this LPA11-K
```

```
00002314  2310   550          .BLKB   1+3                    ; (Count + filler to make PUSHL easy)
          2314   551
          2314   552 LA_B_DAS:                               ; Number of AA11-K's on this LPA11-K
00002318  2314   553          .BLKB   1+3                    ; (Count + filler to make PUSHL easy)
          2318   554
          2318   555 LA_W_NCHN:                              ; Number of AD11-K samples/sequence
0000231A  2318   556          .BLKW   1
          231A   557
          231A   558 LA_K_LAMSKB:                           ; LPA11-K select mask
00002322  231A   559          .BLKW   4
          2322   560
          2322   561 LA_W_NUM:                             ; LPA11-K number ($ DEFINE/SYS LPA11$'NUM)
00002324  2322   562          .BLKW   1
          2324   563
          2324   564 LA_L_IND:                             ; Return status from support routines
00002328  2324   565          .BLKL   1
          2328   566
          2328   567 LA_L_IERROR:                          ; More LPA11-K status
0000232C  2328   568          .BLKL   1
          232C   569
          232C   570 LA_W_IPRSET:                          ; Hardware clock preset value
00002330  232C   571          .BLKL   1
          2330   572
          2330   573 LA_L_IRATE:                           ; Clock rate.  Cf. I/O User's Guide
00002334  2330   574          .BLKL   1
          2334   575
          2334   576 LA_L_NBUF:                            ; Number of times to fill buffers/request
00002338  2334   577          .BLKL   1
          2338   578
          2338   579 LA_W_MODE:                            ; Sampling options.  Cf. I/O User's Guide
0000233A  2338   580          .BLKW   1
          233A   581
          233A   582 LA_W_DWELL:                           ; Number of hardware clock overflows between
0000233C  233A   583          .BLKW   1                    ; ...sample sequences in multirequest mode
          233C   584
          233C   585 LA_W_LDELAY:                          ; Delay from sweep start until first...
0000233E  233C   586          .BLKW   1                    ; ...sample is taken
          233E   587
          233E   588 LA_B_ICHN:                            ; First LPA11-K channel to be sampled
0000233F  233E   589          .BLKB   1
          233F   590
          233F   591 LA_L_DEVDEPEND:                       ; Device dependent longword from...
00002343  233F   592          .BLKL   1                    ; ...original $GETDEV
```

```
                2343   594          .SBTTL  General Data Buffers
           00000000    595          .PSECT  BUFFERS,WRT,NOEXE,PAGE  ; .PSECT name affects ordering!
                0000   596
                0000   597 ;
                0000   598 ; GEN_BUFFERS generates parallel sets of buffers of a given size and data type.
                0000   599 ; Each call to GEN_BUFFERS defines an inner macro, GENERATOR, with the
                0000   600 ; arguments specific to each kind of buffer, and then repeatedly calls the
                0000   601 ; GENERATOR macro to set up each buffer.  The GENERATOR macro is needed in
                0000   602 ; order to define labels for individual buffers.  The labels differ from one
                0000   603 ; another by the last character of their name, a digit.  For a note concerning
                0000   604 ; the names of the following items, see the text where the GEN_ARG_LIST macro
                0000   605 ; is defined.
                0000   606 ;
                000C   607 .MACRO  GEN_BUFFERS     BUF_TYPE,SIZE,DATA_TYPE,NUMBER
                0000   608
                0000   609          .MACRO  GENERATOR       COUNT   ; Defined specific to each buffer type
                0000   610                  .ALIGN  LONG            ; Required by LPA11-K support routines
                0000   611                  LA_AK_'BUF_TYPE''COUNT:
                0000   612                          .B[K'DATA_TYPE  SIZE
                0000   613                          .NOSHOW MEB     ; Keep listings neat
                0000   614          .ENDM   GENERATOR
                0000   615
                0000   616          N = 1
                0000   617          .REPEAT NUMBER
                0000   618                  GENERATOR       \N      ; Generate buffers
                0000   619                  N = N+1
                0000   620          .ENDR   ; NUMBER
                0000   621 .ENDM   GEN_BUFFERS
                0000   622
                0000   623 .SHOW   MEB     ; Show first expansion of the first macro call as an example
                0000   624
                0000   625 GEN_BUFFERS     BUF1,WRITE_SIZE,W,2     ; Data buffers on LPA11-K channel 1
                0000                       LA_AK_BUF1T:
           00001000 0000                           .BLKW   WRITE_SIZE
                2000   626
                2000   627 GEN_BUFFERS     BUF2,WRITE_SIZE,W,2     ; Data buffers on LPA11-K channel 2
                4000   628 GEN_BUFFERS     BUF3,WRITE_SIZE,W,2     ; Data buffers on LPA11-K channel 3
                6000   629 GEN_BUFFERS     BUF4,WRITE_SIZE,W,2     ; Data buffers on LPA11-K channel 4
                8000   630 GEN_BUFFERS     BUF5,WRITE_SIZE,W,2     ; Data buffers on LPA11-K channel 5
                A000   631 GEN_BUFFERS     BUF6,WRITE_SIZE,W,2     ; Data buffers on LPA11-K channel 6
                C000   632 GEN_BUFFERS     BUF7,WRITE_SIZE,W,2     ; Data buffers on LPA11-K channel 7
                E000   633 GEN_BUFFERS     IBUF,MGT_BUF_SIZE,L,NUM_CHANNELS ; Buffer management buffers
                E578   634 GEN_BUFFERS     BFNUM,1,[,NUM_CHANNELS  ; Points to next buffer to fill
```

```
                E594    636         .SBTTL  LPA11-K Subroutine Library Argument Lists
 00000?B8       08B8    637         .PSECT  RODATA,EXE,NOWRT,PAGE   ; EXE attribute changes .PSECT ordering!
                08B8    638
                08B8    639 ;
                08B8    640 ; GEN_ARG_LIST generates sets of argument lists for a given subroutine call.
                08B8    641 ; Generally the names of these argument lists correspond to the unique part
                08B8    642 ; of the name of the subroutine call which uses them.  Only those LPA11-K
                08B8    643 ; support routines which must be called once per LPA11-K I/O bus channel
                08B8    644 ; have their argument lists defined here.  They will be called by CALLG
                08B8    645 ; instructions because the lists are static, one need not take the time to
                08B8    646 ; set up the list each time the routine is called, and the code to do the
                08B8    647 ; calls will be simpler.  Other LPA11-K routines will use CALLS instructions.
                08B8    648 ;
                08B8    649 ; Each call to GEN_ARG_LIST defines an inner macro, GENERATOR, with the
                08B8    650 ; arguments specific to each kind of arg list, and then repeatedly calls the
                08B8    651 ; GENERATOR macro to set up each arg list.  The GENERATOR macro is needed in
                08B8    652 ; order to define labels for individual arg lists.  The labels differ from one
                08B8    653 ; another by the last character of their name, a digit.  Where an individual
                08B8    654 ; argument references storage defined by the GEN_BUFFERS macro, one must
                08B8    655 ; specify the argument name less the last character.  Instead of that last
                08B8    656 ; character, one gives the string, "'L", which will cause the last character
                08B8    657 ; of the argument list to be appended.  The listing file produced by assembling
                08B8    658 ; this file shows the expansion of the first argument of the first call of
                08B8    659 ; GEN_ARG_LIST (viz., SWEEP_ARGL_1).  One can see the symbol defined for the
                08B8    660 ; beginning of the arg list, the count of the arguments to follow, and the
                08B8    661 ; arguments themselves.
                08B8    662 ;
                08B8    663 ; The GENERATOR macro creates two kinds of items on an argument list:
                08B8    664 ; addresses and longwords.  It decides which to create based on the first three
                08B8    665 ; characters of the item to place on the arglist.  If the first three
                08B8    666 ; characters are "LA_", the decision is made that the item is an address of
                08B8    667 ; some named location in the code.  Otherwise, it is assumed that the item for
                08B8    668 ; the arglist is just a literal value to be placed in a longword.
                08B8    669 ;
                08B8    670 .MACRO  GEN_ARG_LIST    ARGL_TYPE,NUMBER,ARGS
                08B8    671
                08B8    672         .MACRO  GENERATOR       L,M     ; Defined specific to each arg list
                08B8    673                 ARGL_TYPE'_ARGL_'L:
                08B8    674                         .BYTE   M,0,0,0
                08B8    675                         .IRP    EACH_ARG,<ARGS>
                08B8    676                                 .IF EQ %LOCATE(<LA_>,EACH_ARG)
                08B8    677                                         .ADDRESS EACH_ARG
                08B8    678                                 .IFF
                08B8    679                                         .LONG   EACH_ARG
                08B8    680                                 .ENDC   ; IF EQ LOCATE
                08B8    681                         .ENDR   ; EACH_ARG
                08B8    682                         .NOSHOW MEB     ; Keep listings neat
                08B8    683         .ENDM   GENERATOR
                08B8    684
                08B8    685         M = 0
                08B8    686         .IRP    EACH_ARG,<ARGS>
                08B8    687                 M = M+1
                08B8    688         .ENDR   ; EACH_ARG
                08B8    689         N = 1
                08B8    690         .REPEAT NUMBER
                08B8    691                 GENERATOR       \N,\M   ; Generate each argument list
                08B8    692                 N = N+1
```

```
                     08B8    693          .ENDR   ; NUMBER
                     08B8    694 .ENDM    GEN_ARG_LIST
                     08B8    695
                     08B8    696 .SHOW    MEB     ; Show first expansion of the first macro call as an example
                     08B8    697
                     08B8    698 GEN_ARG_LIST     SWEEP,NUM_CHANNELS,-
                     08B8    699                  <LA_AK_IBUF'L,LA_W_LBUF,LA_L_NBUF,LA_W_MODE,LA_W_DWELL,-
                     08B8    700                  LA_A_IGTBUF_'L,LA_W_LDELAY,LA_B_ICHN,LA_W_NCHN,LA_L_IND>
                     08B8                         SWEEP_ARGL_T:
00 00 00 0A          08B8                                 .BYTE   10,0,0,0
   0000E000'         08BC                                         .ADDRESS LA_AK_IBUF1
   0000230A'         08C0                                         .ADDRESS LA_W_LBUF
   00002334'         08C4                                         .ADDRESS LA_L_NBUF
   00002338'         08C8                                         .ADDRESS LA_W_MODE
   0000233A'         08CC                                         .ADDRESS LA_W_DWELL
   00000C5D'         08D0                                         .ADDRESS LA_A_IGTBUF_1
   0000233C'         08D4                                         .ADDRESS LA_W_LDELAY
   0000233E'         08D8                                         .ADDRESS LA_B_ICHN
   00002318'         08DC                                         .ADDRESS LA_W_NCHN
   00002324'         08E0                                         .ADDRESS LA_L_IND
                     09EC    701                  ; Start an asynchronous sweep
                     09EC    702
                     09EC    703 GEN_ARG_LIST     SYSWP,2,-
                     09EC    704                  <LA_AK_IBUF'L,LA_W_LBUF,LA_L_NBUF,LA_W_MODE,LA_W_DWELL,-
                     09EC    705                  10+'L,LA_W_LDELAY,LA_B_ICHN,LA_W_NCHN,LA_L_IND>
                     0A44    706                  ; Start a synchronous sweep
                     0A44    707
                     0A44    708 GEN_ARG_LIST     SETIBF,NUM_CHANNELS,-
                     0A44    709                  <LA_AK_IBUF'L,LA_L_IND,LA_K_LAMSKB,LA_AK_BUF'L'1,LA_AK_BUF'L'2>
                     0AEC    710                  ; Set up buffer array for sweeps
                     0AEC    711
                     0AEC    712 GEN_ARG_LIST     IWTBUF,2,<LA_AK_IBUF'L,0,LA_AK_BFNUM'L>
                     0B0C    713                  ; Wait for the next free buffer
                     0B0C    714
                     0B0C    715 GEN_ARG_LIST     IGTBUF,NUM_CHANNELS,<LA_AK_IBUF'L,LA_AK_BFNUM'L>
                     0B60    716                  ; Tell me which buffer to process
                     0B60    717
                     0B60    718 GEN_ARG_LIST     RLSBUF,NUM_CHANNELS,<LA_AK_IBUF'L,LA_L_IND,LITERAL_0,LITERAL_1>
                     0BEC    719                  ; Put a buffer into the ready queue
```

```
                       0BEC    721          .SBTTL  RMS-32 Data Structures
                   00002343    722          .PSECT  RWDATA,WRT,NOEXE,PAGE   ; .PSECT name affects ordering!
                       2343    723          .ALIGN  LONG
                       2344    724
                       2344    725 SYSIN_FAB:                               ; Allocate FAB for SYS$INPUT
                       2344    726          $FAB-
                       2344    727          FNM = <SYS$INPUT>
                       2394    728
                       2394    729 SYSIN_RAB:                               ; Allocate RAB for SYS$INPUT
                       2394    730          $RAB-
                       2394    731          FAB = SYSIN_FAB,-
                       2394    732          ROP = PMT,-
                       2394    733          PBF = PROMPT,-
                       2394    734          PSZ = PMTSIZ,-
                       2394    735          UBF = DEV_NAME,-
                       2394    736          USZ = NAME_LEN
                       23D8    737
                       23D8    738 INI_FAB:                                 ; Allocate FAB for UETINIDEV
                       23D8    739          $FAB-
                       23D8    740          FAC = <GET,PUT,UPD>,-
                       23D8    741          RAT = CR,-
                       23D8    742          SHR = <GET,PUT,UPI>,-
                       23D8    743          FNM = <UETINIDEV.DAT>
                       2428    744
                       2428    745 INI_RAB:                                 ; Allocate RAB for UETINIDEV
                       2428    746          $RAB-
                       2428    747          FAB = INI_FAB,-
                       2428    748          RBF = BUFFER,-
                       2428    749          UBF = BUFFER,-
                       2428    750          USZ = REC_SIZE
                       246C    751
                       246C    752 DDB_RFA:                                 ; RFA storage for INI_RAB
                   00002472    753          .BLKB   6
                       2472    754
                       2472    755          .ALIGN  LONG
                       2474    756 SUP_FAB:                                 ; Allocate FAB for UETSUPDEV
                       2474    757          $FAB-
                       2474    758          FAC = GET,-
                       2474    759          SHR = <UPI,GET>,-
                       2474    760          RAT = CR,-
                       2474    761          FOP = UFO,-
                       2474    762          FNM = <UETSUPDEV.DAT>
```

```
                     24C4   764          .SBTTL  Test and Device Initialization
                 00000000   765          .PSECT  LPA11K,EXE,NOWRT,PAGE
                     0000   766
                     0000   767          .DEFAULT DISPLACEMENT,WORD
                     0000   768
                     0000   769 ;+
                     0000   770 ;       Start up the LPA11-K test. This entails some overhead necessary to cope
                     0000   771 ;       with both expected and unforseen conditions, figuring out just what
                     0000   772 ;       devices are to be tested, making sure we can test the indicated devices
                     0000   773 ;       and setting up writeable space for each device to be tested.
                     0000   774 ;-
                     0000   775
                0000 0000   776 .ENTRY UETLPAK00,^M<>                   ; Entry mask
                     0002   777
          6D  0EF6'CF  DE   0002   778          MOVAL   SSERROR,(FP)            ; Declare exception handler
                     0007   779          $SETSFM_S ENBFLG = #1           ; Enable system service failure mode
                     0010   780          $DCLEXH_S DESBLK = EXIT_DESC    ; Declare an exit handler
                     001B   781
                     001B   782          $OPEN   FAB = SYSIN_FAB,-       ; Open SYS$INPUT
                     001B   783                  ERR = RMS_ERROR
                     002A   784          $CONNECT RAB = SYSIN_RAB,-      ; Connect RAB to SYS$INPUT
                     002A   785                   ERR = RMS_ERROR
                  02 E1   0039   786          BBC     S^#DEV$V_TRM,-          ; BR if SYS$INPUT is NOT a terminal
            1E 2384'CF    003B   787                  SYSIN_FAB+FAB$L_DEV,10$
                     003F   788          $TRNLOG_S LOGNAM = CONTROLLER,- ; Allow terminal user to specify...
                     003F   789                    RSLLEN = DEVNAM_LEN,- ; ...a logical name...
                     003F   790                    RSLBUF = DEVDSC       ; ...for the controller to test
               01 50 D1   0058   791          CMPL    R0,#SS$_NORMAL          ; Was a controller specified?
                  2E 13   005B   792          BEQL    PROC_CONT_NAME          ; BR if it was - go process it
                     005D   793 10$:
                     005D   794          $GET    RAB = SYSIN_RAB,-       ; Read SYS$INPUT...
                     005D   795                  ERR = RMS_ERROR         ; ...for the controller name
            23B6'CF  B0   006C   796          MOVW    SYSIN_RAB+RAB$W_RSZ,-   ; Save the name length
                  020A'CF       0070   797                  DEVNAM_LEN
                     16 12   0073   798          BNEQ    PROC_CONT_NAME          ; BR if we got something
         01EE'CF     14 D0   0075   799          MOVL    #SS$_BADPARAM,STATUS    ; Save the exit status
                  00C4'CF  DF   007A   800          PUSHAL  NO_CTRLNAME             ; Prepare for message...
                     01 DD   007E   801          PUSHL   #1                      ; ...
          00741132   8F  DD   0080   802          PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...
                     03 DD   0086   803          PUSHL   #3                      ; ...
                   0FF4 31   0088   804          BRW     ERROR_EXIT              ; ...to tell of bad setup
                        008B   805
                        008B   806 PROC_CONT_NAME:
0140'CF           020A'CF  3C   008B   807          MOVZWL  DEVNAM_LEN,DEVDSC       ; Set the device name length
                  0140'CF  DF   0092   808          PUSHAL  DEVDSC                  ; Make sure...
                  0140'CF  DF   0096   809          PUSHAL  DEVDSC                  ; ...that the specified controller...
   00000000'GF    02 FB   009A   810          CALLS   #2,G^STR$UPCASE         ; ...is all uppercase for later comaparison
52    0140'CF     01 C1   00A1   811          ADDL3   #1,DEVDSC,R2            ; Estimate the eventual...
      0148'CF     52 A0   00A7   812          ADDW2   R2,PROCESS_NAME         ; ...process name length (incl. "_")
                  DE   00AC   813          MOVAL   PROCESS_NAME+8-         ; Locate first available byte...
                        00AD   814                  +MAX_PROC_NAME-         ; ...in process name handle...
      50    0154'CF       00AD   815                  -PROCESS_NAME_FREE,R0   ; ...for device name
                  08 C3   00B1   816          SUBL3   #PROCESS_NAME_FREE,-    ; Will the device name fit...
                  51 52   00B3   817                  R2,R1                   ; ...in the remaining space?
                  08 15   00B5   818          BLEQ    10$                     ; BR if it will
                  50 51   C2   00B7   819          SUBL2   R1,R0                   ; Overwrite handle otherwise...
      0148'CF     0F B0   00BA   820          MOVW    #MAX_PROC_NAME,PROCESS_NAME ; ...and define the maximum length
```

```
                     00BF    821 10$:
          80   5F 8F    90  00BF    822          MOVB    #^A/ /,(R0)+                    ; Separate handle from device name
      60  015F'CF  0140'CF  28  00C3    823          MOVC3   DEVDSC,DEV_NAME,(R0)    ; Concatenate handle with device name
                     7E    D4  00CB    824          CLRL    -(SP)                   ; Set the time stamp flag
                000F'CF    DF  00CD    825          PUSHAL  TEST_NAME               ; Set the test name
                     02    DD  00D1    826          PUSHL   #2                      ; Push the argument count
              00741039 8F    DD  00D3    827          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
          00000000'GF    04    FB  00D9    828          CALLS   #4,G^LIB$SIGNAL         ; Print the startup message
              0002'CF    08    A8  00E0    829          BISW2   #BEGIN_MSGM,FLAG        ; Set flag so we don't print it again
                                 00E5    830          $SETPRN_S PRCNAM = PROCESS_NAME ; Set the process name to UETLPAK00_x
                                 00F0    831
                     02    E1  00F0    832          BBC     S^#DEV$V_TRM,-          ; BR if SYS$INPUT is NOT a terminal
                  66 2384'CF     00F2    833                  SYSIN_FAB+FAB$L_DEV,20$
                                 00F6    834          $GETDVI_S DEVNAM = SYS$INPUT,-  ; Get the name of...
                                 00F6    835                    EFN    = #SS_SYNCH_EFN,- ; ...device which may abort test
                                 00F6    836                    ITMLST = INPUT_ITMLST,-
                                 00F6    837                    IOSB   = QUAD_STATUS
                  45 01F2'CF     E9  0112    838          BLBC    QUAD_STATUS,20$         ; Avoid CTRL/C handler if any error
                                 0117    839          $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C AST handler
                                 0117    840                    CHAN   = TTCHAN
                                 0128    841          $QIOW_S CHAN     = TTCHAN,-     ; Enable CTRL/C AST's...
                                 0128    842                  FUNC     = #IO$_SETMODE!IO$M_CTRLCAST,-
                                 0128    843                  P1       = CCASTHAND
                0148'CF    DF  0149    844          PUSHAL  PROCESS_NAME            ; ...and tell the user...
                     01    DD  014D    845          PUSHL   #1                      ; ...
              0074832B 8F    DD  014F    846          PUSHL   #UETP$_ABORTC!STS$K_SUCCESS ; ...how to abort gracefully...
          00000000'GF    03    FB  0155    847          CALLS   #3,G^LIB$SIGNAL         ; ...
                                 015C    848 20$:
                                 015C    849          $TRNLOG_S LOGNAM = MODE,-       ; Get the run mode
                                 015C    850                    RSLLEN = BUFFER_PTR,-
                                 015C    851                    RSLBUF = FAO_BUF
              0014'CF    20    8A  0175    852          BICB2   #LC_BITM,BUFFER         ; Convert to upper case
          0014'CF    4F 8F    91  017A    853          CMPB    #^A/O/,BUFFER           ; Is this a one shot?
                     05    12  0180    854          BNEQ    25$                     ; BR if not
              0002'CF    10    A8  0182    855          BISW2   #ONE_SHOTM,FLAG         ; Set flag for one-shot mode
                                 0187    856 25$:
      0014'CF    504D5544 8F    D1  0187    857          CMPL    #^A/DUMP/,BUFFER        ; Special dump mode info wanted?
                     05    12  0190    858          BNEQ    27$                     ; BR if not
              0002'CF    20    A8  0192    859          BISW2   #DUMP_MODEM,FLAG        ; Set flag for dump mode messages
                                 0197    860 27$:
```

```
                                0197   862 ;
                                0197   863 ; From UETINIDEV.DAT and UETSUPDEV.DAT, get information which gives controller
                                0197   864 ; and unit configuration and lets us know if the setup to run this test was
                                0197   865 ; done correctly.
                                0197   866 ;
                                0197   867         $OPEN   FAB = INI_FAB,-         ; Open file 'UETINIDEV.DAT"
                                0197   868                 ERR = RMS_ERROR
                                01A6   869         $CONNECT RAB = INI_RAB,-        ; Connect the RAB and FAB
                                01A6   870                  ERR = RMS_ERROR
                                01B5   871         $MGBLSC_S  INADR = INADDRESS,-  ; Connect to UETSUPDEV global section
                                01B5   872                    RETADR = OUTADDRESS,-
                                01B5   873                    GSDNAM = SUPDEV_GBLSEC,-
                                01B5   874                     FLAGS = #SEC$M_EXPREG
      00000978 8F    50    D1   01D4   875         CMPL    R0,#SS$_NOSUCHSEC       ; Was the section already there?
                       37    12   01DB   876         BNEQ    30$                     ; BR if it was...
                                01DD   877         $OPEN   FAB = SUP_FAB,-         ; ...else open "UETSUPDEV.DAT"
                                01DD   878                 ERR = RMS_ERROR
                                01EC   879         $CRMPSC_S CHAN = SUP_FAB+FAB$L_STV,- ; Create the global section
                                01EC   880                   INADR = INADDRESS,-
                                01EC   881                   RETADR = OUTADDRESS,-
                                01EC   882                   GSDNAM = SUPDEV_GBLSEC,-
                                01EC   883                    FLAGS = #SEC$M_EXPREG!SEC$M_GBL
                                0214   884 30$:
  56  0206'CF   0202'CF    C3   0214   885         SUBL3   OUTADDRESS,OUTADDRESS+4,R6 ; Compute global section length
                                021C   886
                                021C   887 FIND_IT:
                                021C   888         $GET    RAB = INI_RAB,-         ; Get the first record
                                021C   889                 ERR = RMS_ERROR
            01E5'CF    DF   022B   890         PUSHAL  CONT_DESC               ; Make sure...
            01E5'CF    DF   022F   891         PUSHAL  CONT_DESC               ; ...that the controller name...
      00000000'GF    02   FB   0233   892         CALLS   #2,G^STR$UPCASE         ; ...is all uppercase letters
       0014'CF    44 8F   91   023A   893         CMPB    #^A/D/,BUFFER           ; Is this a DDB?
                       27    13   0240   894         BEQL    10$                     ; Go on if not
       0014'CF    45 8F   91   0242   895         CMPB    #^A/E/,BUFFER           ; Is this the end of the file?
                       D2    12   0248   896         BNEQ    FIND_IT                 ; Continue on if not
            0140'CF    DF   024A   897         PUSHAL  DEVDSC                  ; Push device not supported message
            0148'CF    DF   024E   898         PUSHAL  PROCESS_NAME            ; Parameters on the stack
                       02    DD   0252   899         PUSHL   #2
         00748333 8F    DD   0254   900         PUSHL   #UETP$_DENOSU
                       02    F0   025A   901         INSV    #STS$K_ERROR,-          ; Set the severity code...
                       00         025C   902                 #STS$V_SEVERITY,-
                    6E    03         025D   903                 #STS$S_SEVERITY,(SP)
        01EE'CF     6E    D0   025F   904         MOVL    (SP),STATUS             ; ...and save it as the exit status
                       04    DD   0264   905         PUSHL   #4
                 0E16    31   0266   906         BRW     ERROR_EXIT              ; Exit in error
                             0269   907 10$:
015F'CF   001A'CF   020A'CF    29   0269   908         CMPC    DEVNAM_LEN,BUFFER+6,DEV_NAME ; Is this the right controller?
                       A7    12   0273   909         BNEQ    FIND_IT                 ; BR if not
     246C'CF   2438'CF    06   28   0275   910         MOVC3   #6,INI_RAB+RAB$W_RFA,DDB_RFA ; Save the Record File Address
          0018'CF    54 8F   91   027D   911         CMPB    #^A/T/,BUFFER+4         ; Can we test this controller?
                       2F    13   0283   912         BEQL    FOUND_IT                ; BR if we can...
                             0285   913         $FAO_S  CTRSTR = DEAD_CTRLNAME,- ; ...and yell at user if we can't
                             0285   914                 OUTLEN = BUFFER_PTR,-
                             0285   915                 OUTBUF = FAO_BUF,-
                             0285   916                 P1     = #DEVDSC
        01EE'CF    14   D0   029E   917         MOVL    #SS$_BADPARAM,STATUS    ; Set return status
            000C'CF    DF   02A3   918         PUSHAL  BUFFER_PTR              ; ...
```

```
                 01   DD  02A7   919          PUSHL   #1                              ; ...
          00741132 8F   DD  02A9   920          PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...
                 03   DD  02AF   921          PUSHL   #3                              ; ...
               0DCB   31  02B1   922          BRW     ERROR_EXIT                      ; We can't test what we can't test
                          02B4   923
                          02B4   924 FOUND_IT:
                          02B4   925          $GET    RAB = INI_RAB,-                 ; Get a record
                          02B4   926                  ERR = RMS_ERROR
             01E5'CF      DF  02C3   927          PUSHAL  CONT_DESC                       ; Make sure...
             01E5'CF      DF  02C7   928          PUSHAL  CONT_DESC                       ; ...that this line...
     00000000'GF    02   FB  02CB   929          CALLS   #2,G^STR$UPCASE         ; ...is all uppercase letters
      0014'CF    55 8F   91  02D2   930          CMPB    #^A/U/,BUFFER           ; Is this a UCB?
                 24   13  02D8   931          BEQL    30$                             ; BR if it is
      0014'CF    44 8F   91  02DA   932          CMPB    #^A/D/,BUFFER           ; Is this a DDB?
                 19   13  02E0   933          BEQL    20$                             ; BR if yes
      0014'CF    45 8F   91  02E2   934          CMPB    #^A/E/,BUFFER           ; Is this the end?
                 11   13  02E8   935          BEQL    20$                             ; BR if yes
                          02EA   936 10$:
             0151'CF      DF  02EA   937          PUSHAL  ILLEGAL_REC             ; Then this is an error in the record
                 01   DD  02EE   938          PUSHL   #1                              ; Push the error message
          00741132 8F   DD  02F0   939          PUSHL   #UETP$_TEXT!STS$K_ERROR ; Push the signal name
                 03   DD  02F6   940          PUSHL   #3                              ; Push the temp arg count
               0D84   31  02F8   941          BRW     ERROR_EXIT                      ; Finish for good
                          02FB   942 20$:
               0134   31  02FB   943          BRW     ALL_SET                         ; Found DDB or END
                          02FE   944 30$:
      0018'CF    54 8F   91  02FE   945          CMPB    #^A/T/,BUFFER+4         ; Is the unit testable?
                 AE   12  0304   946          BNEQ    FOUND_IT                        ; BR if not
             05   20   3B  0306   947          SKPC    #^A/ 7,#MAX_UNIT_DESIG,- ; Find out where unit number really is
             001A'CF      0309   948                  BUFFER+6
                 50   D7  030C   949          DECL    R0                              ; Units must all be at least one digit
           61    50   30  030E   950          SKPC    #^A/0/,R0,(R1)          ; Skip leading zeroes on the unit
                 50   D6  0312   951          INCL    R0                              ; Compensate for DECL above
     0140'CF     020A'CF  50  A1  0314   952          ADDW3   R0,DEVNAM_LEN,DEVDSC    ; Calculate device'unit string length
           52    020A'CF  3C  031C   953          MOVZWL  DEVNAM_LEN,R2           ; Offset to unit number in DEVDSC
      0000015F'E2  61   50  28  0321   954          MOVC3   R0,(R1),L^DEV_NAME(R2)  ; Append unit number to device
                          0329   955          $GETDEV_S DEVNAM = DEVDSC,-     ; Get the device characteristics
                          0329   956                    PRIBUF = DIB
           57    017A'CF  9A  033E   957          MOVZBL  DIBBUF+DIB$B_DEVCLASS,R7 ; Save the device class
           58    017B'CF  9A  0343   958          MOVZBL  DIBBUF+DIB$B_DEVTYPE,R8 ; Save the device type
                          0348   959          $FAO_S  CTRSTR = CS1,-
                          0348   960                  OUTBUF = FAO_BUF,-
                          0348   961                  P1     = R7,-
                          0348   962                  P2     = R8                     ; Make it into a string
0202'DF    56    0014'CF  06  39  035D   963          MATCHC  #6,BUFFER,R6,@OUTADDRESS ; Find the device class and type
                 1E   13  0366   964          BEQL    40$                             ; BR if it was found
                          0368   965          $FAO_S  CTRSTR = CS3,-          ; Try for full class support
                          0368   966                  OUTBUF = FAO_BUF,-
                          0368   967                      P1 = R7
0202'DF    56    0014'CF  06  39  037B   968          MATCHC  #6,BUFFER,R6,@OUTADDRESS ; Find the device class only
                 0D   12  0384   969          BNEQ    50$                             ; BR if not found
                          0386   970 40$:
           55    000F'CF  9A  0386   971          MOVZBL  TEST_NAME,R5            ; Get the test name length
      0017'CF    63   55  29  038B   972          CMPC3   R5,(R3),TEST_NAME+8     ; Are we the right test?
                 1F   13  0391   973          BEQL    60$                             ; BR if yes
                          0393   974 50$:
                0140'CF   DF  0393   975          PUSHAL  DEVDSC                  ; Push device not supported message
```

```
                0148'CF   DF  0397   976          PUSHAL  PROCESS_NAME              ; Parameters on the stack
                      02  DD  039B   977          PUSHL   #2                        ; Push the argument count
             00748333 8F  DD  039D   978          PUSHL   #UETP$_DENOSU
                      02  F0  03A3   979          INSV    #STS$K_ERROR,-
                      00       03A5   980                  #STS$V_SEVERITY,-
                   6E 03       03A6   981                  #STS$S_SEVERITY,(SP)     ; Set the severity code...
          01EE'CF   6E  D0  03A8   982          MOVL    (SP),STATUS               ; ...and save it as the exit status
                      04  DD  03AD   983          PUSHL   #4                        ; Push the partial arg count...
                    0CCD  31  03AF   984          BRW     ERROR_EXIT                ; ...and split this scene
                          03B2   985 60$:
                          03B2   986          $EXPREG_S PAGCNT = #PAGES,-           ; Get a new node of demand zero memory
                          03B2   987                    RETADR = NEW_NODE
      0238'CF   0240'DF   5D  03C3   988          INSQTI  @NEW_NODE,UNIT_LIST       ; Put the new node in the unit list
            56  0240'CF   D0  03CA   989          MOVL    NEW_NODE,R6               ; Save a copy of its address
              08 A6   01  90  03CF   990          MOVB    #1,UETUNT$B_TYPE(R6)      ; Set the structure type
                 01A4 8F  B0  03D3   991          MOVW    #UETUNT$C_INDSIZ+DEVDEP_SIZE,-
                   09 A6       03D7   992                  UETUNT$W_SIZE(R6)         ; Set the structure size
                      02  88  03D9   993          BISB2   #UETUNT$M_TESTABLE,-      ; By default, the unit is testable
                   0B A6       03DB   994                  UETUNT$B_FLAGS(R6)
            14 A6   0140'CF   90  03DD   995          MOVB    DEVDSC,UETUNT$T_FILSPC(R6) ; Set the device name size
      0144'DF   0140'CF   28  03E3   996          MOVC3   DEVDSC,@DEVDSC+4,-
                   15 A6       03EA   997                  UETUNT$T_FILSPC+1(R6)     ; Save the device name
      57  020A'CF   01  A3  03EC   998          SUBW3   #1,DEVNAM_LEN,R7          ; Pick off letter giving controller...
2322'CF 0144'DF47   41  8F  83  03F2   999          SUBB3   #^A/A/,@DEVDSC+4[R7],LA_W_NUM ; ...and convert to a number
             2322'CF  DF  03FC  1000          PUSHAL  LA_W_NUM                  ; Specify which LPA11-K we will use...
             231A'CF  DF  0400  1001          PUSHAL  LA_K_LAMSKB               ; ...
      00000000'GF    02  FB  0404  1002          CALLS   #2,G^LPA$LAMSKS           ; ...
             017E'CF  D0  040B  1003          MOVL    DIBBUF+DIB$L_DEVDEPEND,-  ; Save the state...
             233F'CF          040F  1004                  LA_L_DEVDEPEND            ; ...of the original microcode
                          0412  1005 ;
                          0412  1006 ; Fill the sets of buffers for the LPA11-K with random data.  There are seven
                          0412  1007 ; contiguous sets of two contiguous buffers each.
                          0412  1008 ;
   57   00003800 8F   D0  0412  1009          MOVL    #WRITE_SIZE*NUM_CHANNELS,R7 ; Total buffer size in longwords
   58   00000000'EF   DE  0419  1010          MOVAL   L^LA_AR_BUF11,R8          ; Point to first buffer
                          0420  1011 70$:
020C'CF   0210'CF   C0  0420  1012          ADDL2   RANDOM2,RANDOM1           ; Create a "random" longword
      88  020C'CF   D0  0427  1013          MOVL    RANDOM1,(R8)+             ; Stuff it into our buffer...
            F1 57   F5  042C  1014          SOBGTR  R7,70$                    ; ...until the buffer is full
                          042F  1015
              FE82  31  042F  1016          BRW     FOUND_IT                  ; Do the next UCB
```

```
                     0432  1018 ;
                     0432  1019 ; Arrive here when we have the device configuration.  In normal or loop forever
                     0432  1020 ; mode, set a timer far enough in the future such that we can do a reasonable
                     0432  1021 ; set of tests before the timer expires, but if our device gets hung, the
                     0432  1022 ; program won't waste too much time before noticing.  Let one-shot mode be a
                     0432  1023 ; special case.
                     0432  1024 ;
                     0432  1025 ALL_SET:
        0238'CF   D5 0432  1026         TSTL    UNIT_LIST                       ; Anything to test?
              16   12 0436  1027         BNEQ    10$                             ; BR if yes
        012B'CF   DF 0438  1028         PUSHAL  NOUNIT_SELECTED                 ; Else set up the error message...
              01   DD 043C  1029         PUSHL   #1                              ; ...argument count...
  00741132    8F   DD 043E  1030         PUSHL   #UETP$_TEXT!STS$K_ERROR         ; ...signal name...
              03   DD 0444  1031         PUSHL   #3                              ; ...and parameter count
    01EE'CF   14   D0 0446  1032         MOVL    #SS$_BADPARAM,STATUS            ; Set return status
          0C31   31   044B  1033         BRW     ERROR_EXIT                      ; ...and give up, complaining
                     044E  1034 10$:
    0002'CF   04   A8 044E  1035         BISW2   #SAFE_TO_UPDM,FLAG              ; OK safe to update UETINIDEV.DAT now
                     0453  1036         $GETJPI_S ITMLST = PRIB_ITMLST           ; Get & save our process' base priority
              56   D4 0468  1037         CLRL    R6                              ; Clear collector for changed priority
                     046A  1038         $TRNLOG_S LOGNAM = LPA11K_PRIORITY,-     ; See if the user...
                     046A  1039                   RSLLEN = BUFFER_PTR,-          ; ...wants to boost...
                     046A  1040                   RSLBUF = FAO_BUF               ; ...our base priority easily
  50  0629 8F   B1 0483  1041         CMPW    #SS$_NOTRAN,R0                  ; We'll try if user asked
              17   13 0488  1042         BEQL    20$                             ; BR if not requested this way
              00   DD 048A  1043         PUSHL   #0                              ; We have string, value goes to stack
              5E   DD 048C  1044         PUSHL   SP                              ; Value address for OTS$CVT below
        000C'CF   DF 048E  1045         PUSHAL  BUFFER_PTR                      ; String for OTS$CVT
00000000'GF   02   FB 0492  1046         CALLS   #2,G^OTS$CVT_TI_L               ; Convert translated string to number
              56 8ED0 0499  1047         POPL    R6                              ; Collect whatever value we got...
           02 50   E9 049C  1048         BLBC    R0,20$                          ; ...but ignore it if we got an error
              03   12 049F  1049         BNEQ    30$                             ; Runtime value has precedence, if we got it
                     04A1  1050 20$:
              56   00   D0 04A1  1051         MOVL    #JOB_PRIORITY,R6                ; Zero or no translation, try built in
                     04A4  1052 30$:
        0258'CF   56   D1 04A4  1053         CMPL    R6,BASE_PRIORITY                ; Do we want to boost our priority?
              25   15 04A9  1054         BLEQ    40$                             ; BR if we're as good as we can get
              56   10   D1 04AB  1055         CMPL    #16,R6                          ; Does user want a timesharing priority?
              20   18 04AE  1056         BGEQ    40$                             ; BR if not - we can't handle real time
                     04B0  1057         $SETPRV_S ENBFLG = #1,-                  ; Give us SETPRI privilege
                     04B0  1058                   PRVADR = NEEDED_PRIVS
                     04C1  1059         $SETPRI_S PRI = R6                       ; Boost our base priority
                     04D0  1060 40$:
                     04D0  1061         $GETJPIW_S ITMLST = IMAGNAM_ITMLST,-      ; We depend on system LPA11-K...
                     04D0  1062                   EFN     = #JPI_EFN,-          ; ...microcode loader process...
                     04D0  1063                   PRCNAM  = LALOADER_PROC,-     ; ...in order to load microcode
                     04D0  1064                   IOSB    = IMAGNAM_IOSB
02AE'CF   08E8 8F   B1 04E9  1065         CMPW    #SS$_NONEXPR,IMAGNAM_IOSB ; Is the mcode loader proc running?
              0F   13 04F0  1066         BEQL    50$                             ; BR if not
063E'CF   0636'CF   39 04F2  1067         MATCHC  LALOADER_IMAGE,LALOADER_IMAGE+8,- ; Proc name OK, check image name
026C'CF   02AC'CF         04F9  1068                 MCODE_IMAGLEN,MCODE_IMAGNAM
              1A   13 04FF  1069         BEQL    60$                             ; BR if correct image, too
                     0501  1070 50$:
        0651'CF   DF 0501  1071         PUSHAL  NO_LALOADER                     ; Someone forgot to do correct setup
              01   DD 0505  1072         PUSHL   #1                              ; We're unhappy about that
  00741132    8F   DD 0507  1073         PUSHL   #UETP$_TEXT!STS$K_ERROR         ; And we've got something to say!
              03   DD 050D  1074         PUSHL   #3
```

```
01EE'CF    000008EA 8F    D0  050F  1075          MOVL    #SS$_NONEXPR-STS$K_WARNING+STS$K_ERROR,STATUS
                    0B64  31  0518  1076          BRW     ERROR_EXIT
                          051B  1077 60$:
          08 0002'CF    04  E1  051B  1078          BBC     #ONE_SHOTV,FLAG,TIME_IT ; BR if not one-shot...
             0002'CF    02  A8  0521  1079          BISW2   #TEST_OVERM,FLAG        ; ...else end after one iteration...
                    0489  31  0526  1080          BRW     ONE_SHOT_TEST           ; ...and do a special test
                          0529  1081 TIME_IT:
                          0529  1082          $SETIMR_S DAYTIM = THREEMIN,-   ; Set timer AST to 3 minutes
                          0529  1083                    ASTADR = TIME_OUT,-
                          0529  1084                    EFN    = #EFN2
```

```
                                053C  1086          .SBTTL  Test the LPA11-K
                                053C  1087 RESTART:
                                053C  1088 ;+
                                053C  1089 ; At this point the device designation is in location DEV_NAME pointed to by
                                053C  1090 ; descriptor DEVDSC. The device is known to be supported by this test.
                                053C  1091 ;
                                053C  1092 ; The LPA11-K is capable of supporting up to eight different users at once
                                053C  1093 ; with the multiuser microcode loaded.  This test uses that microcode to test
                                053C  1094 ; as many devices as are present on the LPA bus.  There will be a request for
                                053C  1095 ; each device and each request will be double buffered.  Each individual
                                053C  1096 ; buffer will be WRITE_SIZE words long.  The requests will be started one
                                053C  1097 ; immediately after the other, with the result that there should be several
                                053C  1098 ; active requests at the same time.  Once the sweeps are all started, the
                                053C  1099 ; program will wait for all the requests to complete and test that they all
                                053C  1100 ; completed without an error.  No comparisons of data will be made, however.
                                053C  1101 ;-
                                053C  1102
                                053C  1103 MU_TEST:
      13 0002'CF    05    E1    053C  1104          BBC     #DUMP_MODEV,FLAG,10$    ; BR if not typing debug info
            07DF'CF    DF    0542  1105          PUSHAL  MU_BEG_MSG
                       01    DD    0546  1106          PUSHL   #1
                 00741133  8F    DD    0548  1107          PUSHL   #UETP$_TEXT!STS$K_INFO
      00000000'GF    03    FB    054E  1108          CALLS   #3,G^LIB$SIGNAL         ; Give some reassurance
                                0555  1109 10$:
                                0555  1110
   22BD'CF    026F'CF    DE    0555  1111          MOVAL   MU_MCODE,MCODE_ADDR     ; We'll be dealing with...
        22B9'CF    01    D0    055C  1112          MOVL    #LA$K_MRMCODE,MCODE_TYPE ; ...multiuser mcode
                                0561  1113
            2328'CF    DF    0561  1114          PUSHAL  LA_L_IERROR
            2324'CF    DF    0565  1115          PUSHAL  LA_L_IND
            2322'CF    DF    0569  1116          PUSHAL  LA_W_NUM
            22B9'CF    DF    056D  1117          PUSHAL  MCODE_TYPE
      00000000'GF    04    FB    0571  1118          CALLS   #4,G^LPA$LOADMC         ; Load multiuser microcode
                  074C    30    0578  1119          BSBW    LOADMC_CHECK            ; Check for errors
                                057B  1120 ;
                                057B  1121 ; Determine the configuration of the LPA11-K I/O bus before entering test mode.
                                057B  1122 ; We must redo the original $GETDEV for this LPA11-K now that we know for sure
                                057B  1123 ; that there is microcode loaded into it.  Devices present on the bus are made
                                057B  1124 ; known by a bit mask in the device dependent characteristics longword.
                                057B  1125 ;
                                057B  1126          $GETDEV_S DEVNAM = DEVDSC,-     ; Get the LPA11-K configuration
                                057B  1127                    PRIBUF = DIB
        57    017E'CF    D0    0590  1128          MOVL    DIBBUF+DIB$L_DEVDEPEND,R7 ; Save in register for easy access
58      57    02    05    EB    0595  1129          FFC     #LA$V_AD1,#2,R7,R8      ; Count each type ..
2310'CF       58    05    83    059A  1130          SUBB3   #LA$V_AD1,R8,LA_B_ADS   ; (AD11-K)
           05 57    07    E1    05A0  1131          BBC     #LA$V_DA,R7,20$         ; ...of device which can...
        2314'CF    01    D0    05A4  1132          MOVL    #1,LA_B_DAS             ; (AA11-K)
                                05A9  1133 20$:
58      57    05    08    EB    05A9  1134          FFC     #LA$V_DIO1,#5,R7,R8     ; ...be on the LPA11-K I/O bus
230C'CF       58    08    83    05AE  1135          SUBB3   #LA$V_DIO1,R8,LA_B_DRS  ; (DR11-K)
                  06BC    30    05B4  1136          BSBW    CONFIG_CHECK
                                05B7  1137
            0257'CF    9F    05B7  1138          PUSHAB  LITERAL_0
            232C'CF    3F    05BB  1139          PUSHAW  LA_W_IPRSET
            2330'CF    DF    05BF  1140          PUSHAL  LA_L_IRATE
            22FA'CF    7F    05C3  1141          PUSHAQ  AINTRVL
      00000000'GF    04    FB    05C7  1142          CALLS   #4,G^LPA$XRATE          ; Calculate the clock A rate
```

```
                2302'CF    50   D0  05CE  1143          MOVL    R0,COMP_RATE            ; Computed rate for debug info
                      0788   30  05D3  1144          BSBW    XRATE_CHECK
                             05D6  1145
           03 0002'CF    05   E0  05D6  1146          BBS     #DUMP_MODEV,FLAG,30$    ; BR if typing debug info
                      0082   31  05DC  1147          BRW     70$                     ; BR around debug typeout
                             05DF  1148 30$:
                        05   DD  05DF  1149          PUSHL   #5
                   22C1'CF   7F  05E1  1150          PUSHAQ  FLOAT_PTR
                   2302'CF   7F  05E5  1151          PUSHAQ  COMP_RATE
          00000000'GF    03   FB  05E9  1152          CALLS   #3,G^FOR$CVT_D_TF       ; Convert clock speed to text
              56   07DB'CF   DE  05F0  1153          MOVAL   S_MSG,R6                ; Assume for now...
              57   07DB'CF   DE  05F5  1154          MOVAL   S_MSG,R7                ; ...that we will not have...
              58   07DB'CF   DE  05FA  1155          MOVAL   S_MSG,R8                ; ...exactly one...
          2310'CF    01   91  05FF  1156          CMPB    #1,LA_B_ADS             ; ...AD11-K,...
                        05   12  0604  1157          BNEQ    40$                     ; ...
              56   07DE'CF   DE  0606  1158          MOVAL   NULL_MSG,R6             ; ...(but change the message if we do)
                             060B  1159 40$:
          2314'CF    01   91  060B  1160          CMPB    #1,LA_B_DAS             ; ...AA11-K,...
                        05   12  0610  1161          BNEQ    50$                     ; ...
              57   07DE'CF   DE  0612  1162          MOVAL   NULL_MSG,R7             ; ...(but change the message if we do)
                             0617  1163 50$:
          230C'CF    01   91  0617  1164          CMPB    #1,LA_B_DRS             ; ...or DR11-K...
                        05   12  061C  1165          BNEQ    60$                     ; ...
              58   07DE'CF   DE  061E  1166          MOVAL   NULL_MSG,R8             ; ...(change the message if we do)
                             0623  1167 60$:
                             C623  1168          $FAO_S  CTRSTR = CONFIG_MSG,-   ; Form configuration message
                             0623  1169                  OUTLEN = BUFFER_PTR,-
                             0623  1170                  OUTBUF = FAO_BUF,-
                             0623  1171                  P1     = LA_B_ADS,-
                             0623  1172                  P2     = R6,-
                             0623  1173                  P3     = LA_B_DAS,-
                             0623  1174                  P4     = R7,-
                             0623  1175                  P5     = LA_B_DRS,-
                             0623  1176                  P6     = R8,-
                             0623  1177                  P7     = #FLOAT_PTR
                000C'CF    DF  064E  1178          PUSHAL  BUFFER_PTR
                      01    DD  0652  1179          PUSHL   #1
                00741133 8F   DD  0654  1180          PUSHL   #UETP$_TEXT!STS$K_INFO
          00000000'GF    03   FB  065A  1181          CALLS   #3,G^LIB$SIGNAL         ; Type LPA11-K configuration
                             0661  1182 70$:
                             0661  1183
                2322'CF    3F  0661  1184          PUSHAW  LA_W_NUM
                2324'CF    DF  0665  1185          PUSHAL  LA_L_IND
                232C'CF    3F  0669  1186          PUSHAW  LA_W_IPRSET
                2330'CF    DF  066D  1187          PUSHAL  LA_L_IRATE
          00000000'GF    04   FB  0671  1188          CALLS   #4,G^LPA$CLOCKA         ; Preset the real time clock
                      06F9   30  0678  1189          BSBW    CLOCKA_CHECK
                             067B  1190 ;
                             067B  1191 ; Set up the buffer structure necessary to transmit and keep track of data.
                             067B  1192 ; Issue test requests based on the devices detected on the LPA11-K I/O bus.
                             067B  1193 ;
                        00   DD  067B  1194          PUSHL   #SETIBF                 ; Indicate the function we want done
                   0A73'CF    01   FB  067D  1195          CALLS   #1,ONCE_FOR_EACH        ; Set up buffers for each channel
                             0682  1196
                2310'CF    95  0682  1197          TSTB    LA_B_ADS                ; If there are no AD11-K's to test...
                        29   13  0686  1198          BEQL    80$                     ; ...BR, else...
2318'CF    2310'CF    40 8F   85  0688  1199          MULB3   #64,LA_B_ADS,LA_W_NCHN   ; ...calc AD11-K samples/seq
```

```
                2324'CF    DF   0691  1200          PUSHAL  LA_L_IND
                025B'CF    DF   0695  1201          PUSHAL  LITERAL_1
                2318'CF    3F   0699  1202          PUSHAW  LA_W_NCHN
                0257'CF    DF   069D  1203          PUSHAL  LITERAL_0
                      00   DD   06A1  1204          PUSHL   #0
                E000'CF    DF   06A3  1205          PUSHAL  LA_AK_IBUF1
    00000000'GF     06     FB   06A7  1206          CALLS   #6,G^LPA$SETADC         ; ...and set up A-to-D channel info
                  05E8     30   06AE  1207          BSBW    SETADC_CHECK
                                06B1  1208 80$:
                                06B1  1209 ;
                                06B1  1210 ; Release to the queue for initial LPA11-K use the buffers used by the test
                                06B1  1211 ; requests.
                                06B1  1212 ;
                      02    DD  06B1  1213          PUSHL   #RLSBUF
                0A73'CF     01    FB  06B3  1214          CALLS   #1,ONCE_FOR_EACH        ; Release buffers for each channel
                                06B8  1215 ;
                                06B8  1216 ; Start the sweeps.  First, initialize some parameters common to all sweep
                                06B8  1217 ; routines.  Then set up a counter, SWEEP_COUNT, which will be incremented
                                06B8  1218 ; for each sweep we successfully start.  Its use is explained below.
                                06B8  1219 ;
                2334'CF     01    B0  06B8  1220          MOVW    #1,LA_W_DWELL           ; # of clock overflows between samples
                2318'CF     01    B0  06BD  1221          MOVW    #1,LA_W_NCHN            ; # of I/O device channels to sample
                2334'CF     02    D0  06C2  1222          MOVL    #2,LA_L_NBUF            ; Count of total buffers to be filled
                      22B8'CF   94  06C7  1223          CLRB    SWEEP_COUNT             ; Clear counter for successful sweeps
                      01    DD  06CB  1224          PUSHL   #SWEEP
                0A73'CF     01    FB  06CD  1225          CALLS   #1,ONCE_FOR_EACH        ; Start sweeps on each channel
                                06D2  1226 ;
                                06D2  1227 ; Wait for sweeps to complete.  Since notification of sweep completion happens
                                06D2  1228 ; at AST level but we want to continue with the rest of the test only when all
                                06D2  1229 ; sweeps have completed, we'll wait until SWEEP_COUNT gets back down to zero,
                                06D2  1230 ; polling it occasionally.  We don't want to wait too long, however, so provide
                                06D2  1231 ; an escape.  Other error checking is done by the AST completion routines.
                                06D2  1232 ;
                        56  D4  06D2  1233          CLRL    R6                      ; Counter for times we've polled
                                06D4  1234 90$:
                      22B8'CF   95  06D4  1235          TSTB    SWEEP_COUNT             ; Finished all sweeps yet?
                            2D  13  06D8  1236          BEQL    100$                    ; BR if we have
                                06DA  1237          $SCHDWK_S DAYTIM = FIVE_SECONDS ; No, give them more time
                                06EB  1238          $HIBER_S
                   DE 56    18    F2  06F2  1239          AOBLSS  #MU_TIME_OUT,R6,90$     ; Loop to see if finished now
                      072D'CF   DF  06F6  1240          PUSHAL  COUNTED_OUT_MSG         ; LPA11-K seems to be hung
                            01  DD  06FA  1241          PUSHL   #1
                00741132    8F  DD  06FC  1242          PUSHL   #UETP$_TEXT!STS$K_ERROR
                            03  DD  0702  1243          PUSHL   #3
                          0978  31  0704  1244          BRW     ERROR_EXIT
                                0707  1245 100$:
                                0707  1246
             13 0002'CF     05    E1  0707  1247          BBC     #DUMP_MODEV,FLAG,110$   ; BR if not typing debug info
                      0800'CF   DF  070D  1248          PUSHAL  MU_END_MSG
                            01  DD  0711  1249          PUSHL   #1
                00741133    8F  DD  0713  1250          PUSHL   #UETP$_TEXT!STS$K_INFO
    00000000'GF     03    FB  0719  1251          CALLS   #3,G^LIB$SIGNAL         ; Give some reassurance
                                0720  1252 110$:
```

```
                     0720  1254 ;+
                     0720  1255 ; This section will be skipped if there are no AD11-K's on the LPA11-K I/O
                     0720  1256 ; bus.  If there are, exercise each by initiating a request.  There will be two
                     0720  1257 ; buffers for each AD11-K, and 200 buffers of information will be passed to the
                     0720  1258 ; test program before the request terminates.  The sweep will be started and
                     0720  1259 ; run at the same clock rate as before.  Wait for the sweep to finish and check
                     0720  1260 ; for completion without errors.  Again, no data comparison is done.
                     0720  1261 ;-
                     0720  1262
                     0720  1263 AD_TEST:
          2310'CF    95  0720  1264          TSTB    LA_B_ADS                ; Have we any AD11-K's?
              03     12  0724  1265          BNEQ    10$                     ; BR if we do
             013C    31  0726  1266          BRW     100$                    ; Skip this section if not
                     0729  1267 10$:
 13 0002'CF  05     E1  0729  1268          BBC     #DUMP_MODEV,FLAG,20$    ; BR if not typing debug info
          081E'CF    DF  072F  1269          PUSHAL  AD_BEG_MSG
              01     DD  0733  1270          PUSHL   #1
       00741133 8F   DD  0735  1271          PUSHL   #UETP$_TEXT!STS$K_INFO
 00000000'GF  03     FB  073B  1272          CALLS   #3,G^LIB$SIGNAL
                     0742  1273 20$:
                     0742  1274
 22BD'CF   0279'CF   DE  0742  1275          MOVAL   AD_MCODE,MCODE_ADDR     ; We'll be dealing with A-to-D mcode
      22B9'CF   02   D0  0749  1276          MOVL    #LA$K_ADMCODE,MCODE_TYPE
                     074E  1277
          2328'CF    DF  074E  1278          PUSHAL  LA_L_IERROR             ; Load dedicated A-to-D microcode
          2324'CF    DF  0752  1279          PUSHAL  LA_L_IND
          2322'CF    3F  0756  1280          PUSHAW  LA_W_NUM
          22B9'CF    DF  075A  1281          PUSHAL  MCODE_TYPE
 00000000'GF  04     FB  075E  1282          CALLS   #4,G^LPA$LOADMC
             055F    30  0765  1283          BSBW    LOADMC_CHECK
                     0768  1284
          0257'CF    9F  0768  1285          PUSHAB  LITERAL_0
          232C'CF    3F  076C  1286          PUSHAW  LA_W_IPRSET
          2330'CF    DF  0770  1287          PUSHAL  LA_L_IRATE
          22FA'CF    7F  0774  1288          PUSHAQ  AINTRVL
 00000000'GF  04     FB  0778  1289          CALLS   #4,G^LPA$XRATE          ; Calculate the clock rate and...
             05DC    30  077F  1290          BSBW    XRATE_CHECK
                     0782  1291
          2322'CF    3F  0782  1292          PUSHAW  LA_W_NUM
          2324'CF    DF  0786  1293          PUSHAL  LA_L_IND
          232C'CF    3F  078A  1294          PUSHAW  LA_W_IPRSET
          2330'CF    DF  078E  1295          PUSHAL  LA_L_IRATE
 00000000'GF  04     FB  0792  1296          CALLS   #4,G^LPA$CLOCKA         ; ...preset the real time clock
             05D8    30  0799  1297          BSBW    CLOCKA_CHECK
                     079C  1298
       52    0A44'CF   DE  079C  1299          MOVAL   SETIBF_ARGL_1,R2        ; (Pacify error checking routine)
 00000000'GF  62     FA  07A1  1300          CALLG   (R2),G^LPA$SETIBF       ; Set up buffer structure
             0609    30  07A8  1301          BSBW    SETIBF_CHECK
                     07AB  1302
2318'CF   2310'CF  40 8F  85  07AB  1303          MULB3   #64,LA_B_ADS,LA_W_NCHN  ; Calculate AD11-K samples/sequence
          2324'CF    DF  07B4  1304          PUSHAL  LA_L_IND
          025B'CF    DF  07B8  1305          PUSHAL  LITERAL_1
          2318'CF    3F  07BC  1306          PUSHAW  LA_W_NCHN
          0257'CF    DF  07C0  1307          PUSHAL  LITERAL_0
              00     DD  07C4  1308          PUSHL   #0
          E000'CF    DF  07C6  1309          PUSHAL  LA_AK_IBUF1
 00000000'GF  06     FB  07CA  1310          CALLS   #6,G^LPA$SETADC         ; Set A-to-D channel information
```

```
                     04C5   30  07D1  1311          BSBW    SETADC_CHECK
                            07D4  1312
          52   0B60'CF   DE  07D4  1313          MOVAL   RLSBUF_ARGL_1,R2        ; (Pacify error checking routine)
00000000'GF     62   FA  07D9  1314          CALLG   (R2),G^LPA$RLSBUF       ; Release buffers for initial use
                     06D6   30  07E0  1315          BSBW    RLSBUF_CHECK
                            07E3  1316
                            07E3  1317 ;
                            07E3  1318 ; Start the sweep of the dedicated A-to-D test.  LA_W_MODE is set to a
                            07E3  1319 ; default value of 0 if there is but one AD11-K, a value of 8192 if there
                            07E3  1320 ; are two AD11-K's.
                            07E3  1321 ;
                2338'CF   B4  07E3  1322          CLRW    LA_W_MODE               ; Default is one AD11-K
          06 2310'CF   01   E1  07E7  1323          BBC     #1,LA_B_ADS,30$         ; BR if that is the case
          00 2338'CF   0D   E2  07ED  1324          BBSS    #13,LA_W_MODE,30$       ; Otherwise indicate two AD11-K's
                            07F3  1325 30$:
     2334'CF   000000C8 8F   D0  07F3  1326          MOVL    #200,LA_L_NBUF          ; Count of buffers to be filled
             233A'CF   01   B0  07FC  1327          MOVW    #1,LA_W_DWELL           ; Time between sample sequences
             233C'CF   01   B0  0801  1328          MOVW    #1,LA_W_LDELAY          ; Delay until first sample is taken
                233E'CF   94  0806  1329          CLRB    LA_B_ICHN               ; First I/O channel to be sampled
             2318'CF   01   B0  080A  1330          MOVW    #1,LA_W_NCHN            ; I/O channels to sample per sequence
          52   09EC'CF   DE  080F  1331          MOVAL   SYSWP_ARGL_1,R2         ; (Pacify error checking routine)
00000000'GF     62   FA  0814  1332          CALLG   (R2),G^LPA$ADSWP        ; Start the sweep
                     05C3   30  081B  1333          BSBW    SWEEP_CHECK
                            081E  1334
                            081F  1335 ;
                            081E  1336 ; Loop while the sweep is continuing.  Release a buffer whenever one is
                            081E  1337 ; returned.
                            081E  1338 ;
                            081E  1339 40$:
          52   0AEC'CF   DE  081E  1340          MOVAL   IWTBUF_ARGL_1,R2        ; (Pacify error checking routine)
00000000'GF     62   FA  0823  1341          CALLG   (R2),G^LPA$IWTBUF       ; Wait for a buffer to be ready
          50   E578'CF   D0  082A  1342          MOVL    LA_AK_BFNUM1,R0         ; Put return status where convenient
                  18   19  082F  1343          BLSS    50$                     ; BR when sweep finishes
             E578'CF   DF  0831  1344          PUSHAL  LA_AK_BFNUM1            ; More to go, release the buffer again
             2324'CF   DF  0835  1345          PUSHAL  LA_L_IND
             E000'CF   DF  0839  1346          PUSHAL  LA_AR_IBUF1
00000000'GF     03   FB  083D  1347          CALLS   #3,G^LPA$RLSBUF
                0672   30  0844  1348          BSBW    RLSBUF_CHECK
                  D5   11  0847  1349          BRB     40$                     ; Loop
                            0849  1350 50$:
                0SD5   30  0849  1351          BSBW    IWTBUF_CHECK
                            084C  1352
 13 0002'CF     05   E1  084C  1353          BBC     #DUMP_MODEV,FLAG,100$   ; BR if not typing debug info
             0846'CF   DF  0852  1354          PUSHAL  AD_END_MSG
                  01   DD  0856  1355          PUSHL   #1
       00741133 8F   DD  0858  1356          PUSHL   #UETP$_TEXT!STS$K_INFO
00000000'GF     03   FB  085E  1357          CALLS   #3,G^LIB$SIGNAL
                            0865  1358 100$:
```

```
                                0865  1360 ;+
                                0865  1361 ; If there is an AA11-K on the LPA11-K I/O bus, start the dedicated D-to-A
                                0865  1362 ; test.  This test will be slightly different from the previous ones in that
                                0865  1363 ; it will allow the user to view a test pattern on an oscilloscope if s/he
                                0865  1364 ; wishes.  The test will fill two adjacent buffers with "ramp data".  This
                                0865  1365 ; data will be output to channel 0 for several seconds and the user may view
                                0865  1366 ; the resultant waveform.  The test will run with the real time clock set at
                                0865  1367 ; the same speed as in the previous tests.
                                0865  1368 ;-
                                0865  1369
                                0865  1370 DA_TEST:
                    2314'CF  95 0865  1371         TSTB    LA_B_DAS                ; Have we any AA11-K's off the LPA11-K?
                         03  12 0869  1372         BNEQ    10$                     ; Continue with D-to-A test if so
                       0134  31 086B  1373         BRW     100$                    ; Skip this section if not
                                086E  1374 10$:
          13 0002'CF    05  E1 086E  1375         BBC     #DUMP_MODEV,FLAG,20$    ; BR if not typing debug info
                    086B'CF  DF 0874  1376         PUSHAL  DA_BEG_MSG
                         01  DD 0878  1377         PUSHL   #1
                   00741133  8F DD 087A  1378         PUSHL   #UETP$_TEXT!STS$K_INFO
              00000000'GF    03 FB 0880  1379         CALLS   #3,G^LIB$SIGNAL
                                0887  1380 20$:
                                0887  1381
                                0887  1382 ;
                                0887  1383 ; Fill the "ramp" data buffer with data which will appear as a sawtooth shape
                                0887  1384 ; when the AA11-K is watched with an oscilloscope.
                                0887  1385 ;
                         57  D4 0887  1386         CLRL    R7                      ; Used as both index and data
                                0889  1387 30$:
            02B8'CF47    57  B0 0889  1388         MOVW    R7,RAMP_DATA[R7]        ; RAMP1(R7) := R7
            12B8'CF47    57  B0 088F  1389         MOVW    R7,RAMP_DATA1[R7]       ; RAMP2(R7) := R7
        EC 57   00000800  8F F2 0895  1390         AOBLSS  #RAMP_HEIGHT,R7,30$     ; Loop until top of ramp is reached
                                089D  1391
          22BD'CF   028D'CF  DE 089D  1392         MOVAL   DA_MCODE,MCODE_ADDR     ; We'll be dealing with D-to-A mcode
               22B9'CF    03  D0 08A4  1393         MOVL    #LA$K_DAMCODE,MCODE_TYPE
                                08A9  1394
                    2328'CF  DF 08A9  1395         PUSHAL  LA_L_IERROR             ; Load dedicated D-to-A microcode
                    2324'CF  DF 08AD  1396         PUSHAL  LA_L_IND
                    2322'CF  3F 08B1  1397         PUSHAW  LA_W_NUM
                    22B9'CF  DF 08B5  1398         PUSHAL  MCODE_TYPE
              00000000'GF    04 FB 08B9  1399         CALLS   #4,G^LPA$LOADMC
                       0404  30 08C0  1400         BSBW    LOADMC_CHECK
                                08C3  1401
                    0257'CF  9F 08C3  1402         PUSHAB  LITERAL_0
                    232C'CF  3F 08C7  1403         PUSHAW  LA_W_IPRSET
                    2330'CF  DF 08CB  1404         PUSHAL  LA_L_IRATE
                    22FA'CF  7F 08CF  1405         PUSHAQ  AINTRVL
              00000000'GF    04 FB 08D3  1406         CALLS   #4,G^LPA$XRATE          ; Calculate the clock rate and...
                       0481  30 08DA  1407         BSBW    XRATE_CHECK
                                08DD  1408
                    2322'CF  3F 08DD  1409         PUSHAW  LA_W_NUM
                    2324'CF  DF 08E1  1410         PUSHAL  LA_L_IND
                    232C'CF  3F 08E5  1411         PUSHAW  LA_W_IPRSET
                    2330'CF  DF 08E9  1412         PUSHAL  LA_L_IRATE
              00000000'GF    04 FB 08ED  1413         CALLS   #4,G^LPA$CLOCKA         ; ...preset the real time clock
                       047D  30 08F4  1414         BSBW    CLOCKA_CHECK
                                08F7  1415
                    12B8'CF  DF 08F7  1416         PUSHAL  RAMP_DATA1              ; Set up...
```

```
                02B8'CF    DF  08FB  1417          PUSHAL  RAMP_DATA                ; ...a special...
                231A'CF    DF  08FF  1418          PUSHAL  LA_K_LAMSKB              ; ...argument list...
                2324'CF    DF  0903  1419          PUSHAL  LA_L_IND                 ; ...for this...
                E0C8'CF    DF  0907  1420          PUSHAL  LA_AR_IBUF2              ; ...one call
                     05    DD  090B  1421          PUSHL   #5
                52   5E    D0  090D  1422          MOVL    SP,R2                    ; Pacify error checking & fake CALLG
      00000000'GF   62    FA  0910  1423          CALLG   (R2),G^LPA$SETIBF        ; Set up buffer structure
                 049A   30  0917  1424          BSBW    SETIBF_CHECK
                5E   14    C0  091A  1425          ADDL2   #20,SP                   ; Clean up the stack
                           091D  1426
            52   0B74'CF    DE  091D  1427          MOVAL   RLSBUF_ARGL_2,R2         ; (Pacify error checking routine)
      00000000'GF   62    FA  0922  1428          CALLG   (R2),G^LPA$RLSBUF        ; Release buffers for initial use
                  058D   30  0929  1429          BSBW    RLSBUF_CHECK
                           092C  1430
    2334'CF    0000004B 8F    D0  092C  1431          MOVL    #75,LA_L_NBUF            ; Count of buffers to be filled
                2338'CF    B4  0935  1432          CLRW    LA_W_MODE                ; Sampling options
            233A'CF    01    B0  0939  1433          MOVW    #1,LA_W_DWELL            ; Time between sample sequences
            233C'CF    01    B0  093E  1434          MOVW    #1,LA_W_LDELAY           ; Delay until first sample is taken
                233E'CF    94  0943  1435          CLRB    LA_B_ICHN                ; First I/O channel to be sampled
            2318'CF    01    B0  0947  1436          MOVW    #1,LA_W_NCHN             ; I/O channels to sample per sequence
            52   0A18'CF    DE  094C  1437          MOVAL   SYSWP_ARGL_2,R2          ; (Pacify error checking routine)
      00000000'GF   62    FA  0951  1438          CALLG   (R2),G^LPA$DASWP         ; Start the sweep
                  0486   30  0958  1439          BSBW    SWEEP_CHECK
                           095B  1440
                           095B  1441 ;
                           095B  1442 ; Loop while the sweep is continuing.  Release a buffer whenever one is
                           095B  1443 ; returned.
                           095B  1444 ;
                           095B  1445 40$:
            52   0AFC'CF    DE  095B  1446          MOVAL   IWTBUF_ARGL_2,R2         ; (Pacify error checking routine)
      00000000'GF   62    FA  0960  1447          CALLG   (R2),G^LPA$IWTBUF        ; Wait for a buffer to be ready
               50   E57C'CF    D0  0967  1448          MOVL    LA_AK_BFNUM2,R0          ; Put return status where convenient
                        18    19  096C  1449          BLSS    50$                      ; BR when sweep finishes
                E57C'CF    DF  096E  1450          PUSHAL  LA_AK_BFNUM2             ; More to go, release the buffer again
                2324'CF    DF  0972  1451          PUSHAL  LA_L_IND
                E0C8'CF    DF  0976  1452          PUSHAL  LA_AR_IBUF2
      00000000'GF   03    FB  097A  1453          CALLS   #3,G^LPA$RLSBUF
                  0535   30  0981  1454          BSBW    RLSBUF_CHECK
                    D5   11  0984  1455          BRB     40$                      ; Loop
                           0986  1456 50$:
                  0498   30  0986  1457          BSBW    IWTBUF_CHECK
                           0989  1457'
     13 0002'CF    05    E1  0989  1459          BBC     #DUMP_MODEV,FLAG,100$    ; BR if not typing debug info
                0893'CF    DF  098F  1460          PUSHAL  DA_END_MSG
                     01    DD  0993  1461          PUSHL   #1
          00741133 8F    DD  0995  1462          PUSHL   #UETP$_TEXT!STS$K_INFO
      00000000'GF   03    FB  099C  1463          CALLS   #3,G^LIB$SIGNAL
                           09A2  1464 100$:
                           09A2  1465
                           09A2  1466
                0214'CF    D6  09A2  1467          INCL    ITERATION                ; Increment iteration count
     03 0002'CF    01    E0  09A6  1468          BBS     #TEST_OVERV,FLAG,110$    ; BR if the test is over
                  FB8D   31  09AC  1469          BRW     RESTART                  ; Loop until the test is over
                           09AF  1470 110$:
                  0047   31  09AF  1471          BRW     SUC_EXIT
```

```
                        09B2  1473          .SBTTL  One-Shot Testing
                        09B2  1474 ;+
                        09B2  1475 ; The forced overrun test will run only in one shot mode.  It will do a couple
                        09B2  1476 ; of regular data transfers using the multiuser microcode, then force a data
                        09B2  1477 ; overrun to see if the LPA11-K can detect the situation.  Instead of using
                        09B2  1478 ; the high level language interface routines, it will use $QIOs to access the
                        09B2  1479 ; LPA11-K most of the time.  The exception is in initializing the LPA11-K.
                        09B2  1480 ; This is an error prone task which is better performed by the system
                        09B2  1481 ; microcode loader process.  We rationalize by saying that unless a user had
                        09B2  1482 ; special microcode to load, he, too, would always use the standard procedure.
                        09B2  1483 ; Using $QIOs in other places permits us to take advantage of the $QIO's
                        09B2  1484 ; flexibility.  When we force an overrun, we don't want the user to think his
                        09B2  1485 ; LPA11-K is malfunctioning, so inhibit error logging with the IO$M_INHERLOG
                        09B2  1486 ; modifier.  The forced overrun test will be implemented in a future release.
                        09B2  1487 ;-
                        09B2  1488 ONE_SHOT_TEST:
                        09B2  1489 FO_TEST:
22BD'CF   0279'CF   DE  09B2  1490          MOVAL   AD_MCODE,MCODE_ADDR      ; Try loading all types of ucode...
     22B9'CF   02   D0  09B9  1491          MOVL    #LA$K_ADMCODE,MCODE_TYPE ; ...analog-to-digital...
               1F   10  09BE  1492          BSBB    100$
22BD'CF   028D'CF   DE  09C0  1493          MOVAL   DA_MCODE,MCODE_ADDR      ; ...
     22B9'CF   03   D0  09C7  1494          MOVL    #LA$K_DAMCODE,MCODE_TYPE ; ...digital-to-analogue...
               11   10  09CC  1495          BSBB    100$
22BD'CF   026F'CF   DE  09CE  1496          MOVAL   MU_MCODE,MCODE_ADDR      ; ...and finishing up with...
     22B9'CF   01   D0  09D5  1497          MOVL    #LA$K_MRMCODE,MCODE_TYPE ; ...multiuser mcode
               03   10  09DA  1498          BSBB    100$
             001A   31  09DC  1499          BRW     SUC_EXIT
                        09DF  1500
                        09DF  1501 100$:
          2328'CF   DF  09DF  1502          PUSHAL  LA_L_IERROR
          2324'CF   DF  09E3  1503          PUSHAL  LA_L_IND
          2322'CF   DF  09E7  1504          PUSHAL  LA_W_NUM
          22B9'CF   DF  09EB  1505          PUSHAL  MCODE_TYPE
00000000'GF   04   FB  09EF  1506          CALLS   #4,G^LPA$LOADMC          ; Load multiuser microcode
             02CE   31  09F6  1507          BRW     LOADMC_CHECK             ; Check for errors
                        09F9  1508          ; Note that LOADMC_CHECK returns via RSB if it returns at all.
                        09F9  1509          ; Branching to it is equivalent to BSBW followed by RSB.
```

```
                        09F9  1511          .SBTTL  Finish Testing
                        09F9  1512 SUC_EXIT:
                        09F9  1513          $TRNLOG_S LOGNAM = MODE,-
                        09F9  1514                    RSLLEN = BUFFER_PTR,-
                        09F9  1515                    RSLBUF = FAO_BUF      ; Get the run mode
     0014'CF    20   8A  0A12  1516          BICB2   #LC_BITM,BUFFER       ; Convert to upper case
  0014'CF    4C 8F   91  0A17  1517          CMPB    #^A7L/,BUFFER         ; Is this a loop for ever?
                40   12  0A1D  1518          BNEQ    10$                   ; BR if not
     0002'CF    02   AA  0A1F  1519          BICW2   #TEST_OVERM,FLAG      ; Reset the termination flag
          0218'CF   D6  0A24  1520          INCL    PASS                  ; Bump the pass count
                        0A28  1521          $FAO_S  CTRSTR = PASS_MSG,-
                        0A28  1522                  OUTLEN = BUFFER_PTR,-
                        0A28  1523                  OUTBUF = FAO_BUF,-
                        0A28  1524                  P1     = PASS,-
                        0A28  1525                  P2     = ITERATION,-
                        0A28  1526                  P3     = #0           ; Make the end of pass message
          000C'CF   DF  0A45  1527          PUSHAL  BUFFER_PTR            ; Push the string desc.
                01  DD  0A49  1528          PUSHL   #1                    ; Push arg count
        00741133 8F   DD  0A4B  1529          PUSHL   #UETP$_TEXT!STS$K_INFO ; Push the signal name
  00000000'GF    03   FB  0A51  1530          CALLS   #3,G^LIB$SIGNAL       ; Print the end of pass message
          0214'CF   D4  0A58  1531          CLRL    ITERATION             ; Reset the iteration count
             FACA   31  0A5C  1532          BRW     TIME_IT               ; Do the next pass
                        0A5F  1533 10$:
01EE'CF    10000001 8F   D0  0A5F  1534          MOVL    #SS$_NORMAL!STS$M_INHIB_MSG,STATUS ; Set successful exit status
                        0A68  1535          $EXIT_S STATUS                ; Exit with the status
```

```
                     0A73  1537          .SBTTL  Common Subroutine Caller for Multirequest Mode
                     0A73  1538 ;++
                     0A73  1539 ; FUNCTIONAL DESCRIPTION:
                     0A73  1540 ;       This routine will be called whenever there is a common function to
                     0A73  1541 ;       perform for each LPA11-K I/O bus channel in multirequest mode.  For
                     0A73  1542 ;       example, setting up channel buffer management info needs to be done
                     0A73  1543 ;       for all connected devices.  The routine provides a way to pool common
                     0A73  1544 ;       code and make the main listing somewhat more readable.
                     0A73  1545 ;
                     0A73  1546 ; CALLING SEQUENCE:
                     0A73  1547 ;       CALLx   #1,ONCE_FOR_EACH
                     0A73  1548 ;
                     0A73  1549 ; INPUT PARAMETERS:
                     0A73  1550 ;       4(AP) has the function code for what each channel needs done.
                     0A73  1551 ;
                     0A73  1552 ; IMPLICIT INPUTS:
                     0A73  1553 ;       The argument lists for all subroutine calls are defined by the
                     0A73  1554 ;       GEN_ARG_LIST macro.
                     0A73  1555 ;       Various buffers and other subroutine parameters are elsewhere defined.
                     0A73  1556 ;
                     0A73  1557 ; OUTPUT PARAMETERS:
                     0A73  1558 ;       NONE
                     0A73  1559 ;
                     0A73  1560 ; IMPLICIT OUTPUTS:
                     0A73  1561 ;       Buffers and parameters mentioned above may be modified by subroutine
                     0A73  1562 ;       calls.
                     0A73  1563 ;
                     0A73  1564 ; COMPLETION CODES:
                     0A73  1565 ;       NONE
                     0A73  1566 ;
                     0A73  1567 ; SIDE EFFECTS:
                     0A73  1568 ;       An error will result in an explanatory message and program abort.
                     0A73  1569 ;
                     0A73  1570 ;--
                     0A73  1571
                     0A73  1572 ONCE_FOR_EACH:
                0FFC 0A73  1573          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0A75  1574
02   00   04 AC   8F 0A75  1575          CASEB   4(AP),#0,#LAST_FUNCTION ; Go our separate ways based on function
                     0A7A  1576 10$:
                0032' 0A7A  1577                  .WORD   SETIBF_SECTION-10$
                00E4' 0A7C  1578                  .WORD   SWEEP_SECTION-10$
                008B' 0A7E  1579                  .WORD   RLSBUF_SECTION-10$
                     0A80  1580 ; Fall through only if there is a bad argument list.
                     0A80  1581
                     0A80  1582          $FAO_S  CTRSTR = BAD_CASE_MSG,- ; Bad parameter passed to ONCE_FOR_EACH
                     0A80  1583                  OUTLEN = BUFFER_PTR,-
                     0A80  1584                  OUTBUF = FAO_BUF,-
                     0A80  1585                  P1     = 4(AP)
        000C'CF   DF 0A96  1586          PUSHAL  BUFFER_PTR
              01  DD 0A9A  1587          PUSHL   #1
 00741134 8F   DD 0A9C  1588          PUSHL   #UETP$_TEXT!STS$K_SEVERE
              03  DD 0AA2  1589          PUSHL   #3
01EE'CF   2C   D0 0AA4  1590          MOVL    #SS$_ABORT,STATUS       ; This has to be a serious problem...
            05D3  31 0AA9  1591          BRW     ERROR_EXIT              ; ...and we can't possibly recover
```

```
                          0AAC  1593 SETIBF_SECTION.                          ; Do a call(s) to LPA$SETIBF
                2310'CF   95    0AAC  1594          TSTB    LA_B_ADS          ; Any AD11-Ks configured?
                      07  13    0AB0  1595          BEQL    10$               ; BR if not
           52   0A44'CF   DE    0AB2  1596          MOVAL   SETIBF_ARGL_1,R2  ; Set up AD11-K arg list
                      42  10    0AB7  1597          BSBB    SETIBF_COMMON     ; Set up buffers
                          0AB9  1598 10$:
                2314'CF   95    0AB9  1599          TSTB    LA_B_DAS          ; Any AA11-Ks configured?
                      07  13    0ABD  1600          BEQL    20$               ; BR if not
           52   0A5C'CF   DE    0ABF  1601          MOVAL   SETIBF_ARGL_2,R2  ; Set up AA11-K arg list
                      35  10    0AC4  1602          BSBB    SETIBF_COMMON     ; Set up buffers
                          0AC6  1603 20$:
     04   01   230C'CF   8F    0AC6  1604          CASEB   LA_B_DRS,#1,#5-1  ; Dispatch on the number of DR11-Ks
000B'0012'0019'0020'0027'   0ACC  1605 30$:          .WORD   80$-30$,70$-30$,60$-30$,50$-30$,40$-30$
                      04  0AD6  1606          RET                           ; Fall through if none configured
                          0AD7  1607 40$:
           52   0AD4'CF   DE    0AD7  1608          MOVAL   SETIBF_ARGL_7,R2  ; Set up DR11-K #5 arg list
                      1D  10    0ADC  1609          BSBB    SETIBF_COMMON     ; Set up buffers
                          0ADE  1610 50$:
           52   0ABC'CF   DE    0ADE  1611          MOVAL   SETIBF_ARGL_6,R2  ; Set up DR11-K #4 arg list
                      16  10    0AE3  1612          BSBB    SETIBF_COMMON     ; Set up buffers
                          0AE5  1613 60$:
           52   0AA4'CF   DE    0AE5  1614          MOVAL   SETIBF_ARGL_5,R2  ; Set up DR11-K #3 arg list
                      0F  10    0AEA  1615          BSBB    SETIBF_COMMON     ; Set up buffers
                          0AEC  1616 70$:
           52   0A8C'CF   DE    0AEC  1617          MOVAL   SETIBF_ARGL_4,R2  ; Set up DR11-K #2 arg list
                      08  10    0AF1  1618          BSBB    SETIBF_COMMON     ; Set up buffers
                          0AF3  1619 80$:
           52   0A74'CF   DE    0AF3  1620          MOVAL   SETIBF_ARGL_3,R2  ; Set up DR11-K #1 arg list
                      01  10    0AF8  1621          BSBB    SETIBF_COMMON     ; Set up buffers
                      04  0AFA  1622          RET
                          0AFB  1623
                          0AFB  1624 SETIBF_COMMON:                           ; Common code for each type of device
00000000'GF         62    FA    0AFB  1625          CALLG   (R2),G^LPA$SETIBF ; Set up buffer array for sweeps
                    02AF  31    0B02  1626          BRW     SETIBF_CHECK      ; Return fat and happy via error check
```

```
                        0B05  1628 RLSBUF_SECTION:                        ; Do a call(s) to LPA$RLSBUF
             2310'CF    95  0B05  1629          TSTB    LA_B_ADS            ; Any AD11-Ks configured?
                   07   13  0B09  1630          BEQL    10$                 ; BR if not
          52 0B60'CF    DE  0B0B  1631          MOVAL   RLSBUF_ARGL_1,R2    ; Set up AD11-K arg list
                   42   10  0B10  1632          BSBB    RLSBUF_COMMON       ; Release buffers
                        0B12  1633 10$:
             2314'CF    95  0B12  1634          TSTB    LA_B_DAS            ; Any AA11-Ks configured?
                   07   13  0B16  1635          BEQL    20$                 ; BR if not
          52 0B74'CF    DE  0B18  1636          MOVAL   RLSBUF_ARGL_2,R2    ; Set up AA11-K arg list
                   35   10  0B1D  1637          BSBB    RLSBUF_COMMON       ; Release buffers
                        0B1F  1638 20$:
       04   01 230C'CF    8F  0B1F  1639          CASEB   LA_B_DRS,#1,#5-1    ; Dispatch on the number of DR11-Ks
000B'0012'0019'0020'0027' 0B25  1640 30$:          .WORD   80$-30$,70$-30$,60$-30$,50$-30$,40$-30$
                   04   0B2F  1641          RET                         ; Fall through if none configured
                        0B30  1642 40$:
          52 0BD8'CF    DE  0B30  1643          MOVAL   RLSBUF_ARGL_7,R2    ; Set up DR11-K #5 arg list
                   1D   10  0B35  1644          BSBB    RLSBUF_COMMON       ; Release buffers
                        0B37  1645 50$:
          52 0BC4'CF    DE  0B37  1646          MOVAL   RLSBUF_ARGL_6,R2    ; Set up DR11-K #4 arg list
                   16   10  0B3C  1647          BSBB    RLSBUF_COMMON       ; Release buffers
                        0B3E  1648 60$:
          52 0BB0'CF    DE  0B3E  1649          MOVAL   RLSBUF_ARGL_5,R2    ; Set up DR11-K #3 arg list
                   0F   10  0B43  1650          BSBB    RLSBUF_COMMON       ; Release buffers
                        0B45  1651 70$:
          52 0B9C'CF    DE  0B45  1652          MOVAL   RLSBUF_ARGL_4,R2    ; Set up DR11-K #2 arg list
                   08   10  0B4A  1653          BSBB    RLSBUF_COMMON       ; Release buffers
                        0B4C  1654 80$:
          52 0B88'CF    DE  0B4C  1655          MOVAL   RLSBUF_ARGL_3,R2    ; Set up DR11-K #1 arg list
                   01   10  0B51  1656          BSBB    RLSBUF_COMMON       ; Release buffers
                   04   0B53  1657          RET
                        0B54  1658
                        0B54  1659 RLSBUF_COMMON:                          ; Common code for each type of device
00000000'GF          62  FA  0B54  1660          CALLG   (R2),G^LPA$RLSBUF   ; Release buffer array for sweeps
                 035B   31  0B5B  1661          BRW     RLSBUF_CHECK        ; Return fat and happy via error check
```

```
                     0B5E  1663 SWEEP_SECTION:                          ; Do a call(s) to LPA$SWEEP
          2310'CF    95    0B5E  1664          TSTB    LA_B_ADS          ; Any AD11-Ks configured?
                20    13    0B62  1665          BEQL    10$               ; BR if not
2338'CF    0040 8F    B0    0B64  1666          MOVW    #64,LA_W_MODE     ; Set up call specific...
     233C'CF    00    B0    0B6B  1667          MOVW    #00,LA_W_LDELAY   ; ...arguments...
     233E'CF    00    90    0B70  1668          MOVB    #00,LA_B_ICHN     ; ...
 53   00000000'GF    DE    0B75  1669          MOVAL   G^LPA$ADSWP,R3
     52   08B8'CF    DE    0B7C  1670          MOVAL   SWEEP_ARGL_1,R2   ; Set up AD11-K arg list
             0099    30    0B81  1671          BSBW    SWEEP_COMMON      ; Start up a sweep
                     0B84  1672 10$:
          2314'CF    95    0B84  1673          TSTB    LA_B_DAS          ; Any AA11-Ks configured?
                20    13    0B88  1674          BEQL    20$               ; BR if not
2338'CF    0040 8F    B0    0B8A  1675          MOVW    #64,LA_W_MODE     ; Set up call specific...
     233C'CF    19    B0    0B91  1676          MOVW    #25,LA_W_LDELAY   ; ...arguments...
     233E'CF    00    90    0B96  1677          MOVB    #00,LA_B_ICHN     ; ...
 53   00000000'GF    DE    0B9B  1678          MOVAL   G^LPA$DASWP,R3
     52   08E4'CF    DE    0BA2  1679          MOVAL   SWEEP_ARGL_2,R2   ; Set up AA11-K arg list
             0073    30    0BA7  1680          BSBW    SWEEP_COMMON      ; Start up a sweep
                     0BAA  1681 20$:
     2338'CF    00    B0    0BAA  1682          MOVW    #00,LA_W_MODE     ; Value for all DISWP calls
 53   00000000'GF    DE    0BAF  1683          MOVAL   G^LPA$DISWP,R3
04   01   230C'CF    8F    0BB6  1684          CASEB   LA_B_DRS,#1,#5-1  ; Dispatch on the number of DR11-Ks
000B'001C'002D'003E'004F'  0BBC  1685 30$:     .WORD   80$-30$,70$-30$,60$-30$,50$-30$,40$-30$
                     04    0BC6  1686          RET                       ; Fall through if none configured
                     0BC7  1687 40$:
     233C'CF    00    B0    0BC7  1688          MOVW    #00,LA_W_LDELAY   ; Set up call specific...
     233E'CF    04    90    0BCC  1689          MOVB    #04,LA_B_ICHN     ; ...arguments
     52   09C0'CF    DE    0BD1  1690          MOVAL   SWEEP_ARGL_7,R2   ; Set up DR11-K #5 arg list
                45    10    0BD6  1691          BSBB    SWEEP_COMMON      ; Start up a sweep
                     0BD8  1692 50$:
     233C'CF    19    B0    0BD8  1693          MOVW    #25,LA_W_LDELAY   ; Set up call specific...
233E'CF    03    90    0BDD  1694          MOVB    #03,LA_B_ICHN     ; ...arguments
52   0994'CF    DE    0BE2  1695          MOVAL   SWEEP_ARGL_6,R2   ; Set up DR11-K #4 arg list
                34    10    0BE7  1696          BSBB    SWEEP_COMMON      ; Start up a sweep
                     0BE9  1697 60$:
233C'CF    00    B0    0BE9  1698          MOVW    #00,LA_W_LDELAY   ; Set up call specific...
233E'CF    02    90    0BEE  1699          MOVB    #02,LA_B_ICHN     ; ...arguments
52   0968'CF    DE    0BF3  1700          MOVAL   SWEEP_ARGL_5,R2   ; Set up DR11-K #3 arg list
                23    10    0BF8  1701          BSBB    SWEEP_COMMON      ; Start up a sweep
                     0BFA  1702 70$:
233C'CF    19    B0    0BFA  1703          MOVW    #25,LA_W_LDELAY   ; Set up call specific...
233E'CF    01    90    0BFF  1704          MOVB    #01,LA_B_ICHN     ; ...arguments
52   093C'CF    DE    0C04  1705          MOVAL   SWEEP_ARGL_4,R2   ; Set up DR11-K #2 arg list
                12    10    0C09  1706          BSBB    SWEEP_COMMON      ; Start up a sweep
                     0C0B  1707 80$:
233C'CF    00    B0    0C0B  1708          MOVW    #00,LA_W_LDELAY   ; Set up call specific...
233E'CF    00    90    0C10  1709          MOVB    #00,LA_B_ICHN     ; ...arguments
52   0910'CF    DE    0C15  1710          MOVAL   SWEEP_ARGL_3,R2   ; Set up DR11-K #1 arg list
                01    10    0C1A  1711          BSBB    SWEEP_COMMON      ; Start up a sweep
                     04    0C1C  1712          RET
                     0C1D  1713
                     0C1D  1714 SWEEP_COMMON:                           ; Common code for each type of device
     63   62    FA    0C1D  1715          CALLG   (R2),(R3)             ; Start up a sweep
     22B8'CF    96    0C20  1716          INCB    SWEEP_COUNT           ; Assume sweep started successfully
        01BA    31    0C24  1717          BRW     SWEEP_CHECK           ; Return fat and happy via error check
```

```
                     0C27  1719          .SBTTL  AST Level Completion Routines
                     0C27  1720 ;++
                     0C27  1721 ; FUNCTIONAL DESCRIPTION:
                     0C27  1722 ;       These routines are called whenever the LPA11-K signals that it has
                     0C27  1723 ;       a buffer ready to process.  Although we do nothing with the buffer,
                     0C27  1724 ;       we have to appease the LPA11-K by picking up the buffer number.
                     0C27  1725 ;
                     0C27  1726 ; CALLING SEQUENCE:
                     0C27  1727 ;       Called at AST level by LPA11-K support routines via CALLx.  Indicated
                     0C27  1728 ;       for use by the LPA$ sweep routine IEFN argument.
                     0C27  1729 ;
                     0C27  1730 ; INPUT PARAMETERS:
                     0C27  1731 ;       NONE
                     0C27  1732 ;
                     0C27  1733 ; IMPLICIT INPUTS:
                     0C27  1734 ;       Because each LPA11-K channel gets the address of a different routine
                     0C27  1735 ;       when the sweeps are started, the entry point to each of these routines
                     0C27  1736 ;       effectively tells us which channel has a waiting buffer.
                     0C27  1737 ;
                     0C27  1738 ; OUTPUT PARAMETERS:
                     0C27  1739 ;       NONE
                     0C27  1740 ;
                     0C27  1741 ; IMPLICIT OUTPUTS:
                     0C27  1742 ;       Buffers and parameters mentioned above may be modified by subroutine
                     0C27  1743 ;       calls.
                     0C27  1744 ;
                     0C27  1745 ; COMPLETION CODES:
                     0C27  1746 ;       NONE
                     0C27  1747 ;
                     0C27  1748 ; SIDE EFFECTS:
                     0C27  1749 ;       An error will result in an explanatory message and program abort.
                     0C27  1750 ;
                     0C27  1751 ;--
                     0C27  1752
                     0C27  1753 LA_A_IGTBUF_7:
                0FFC 0C27  1754          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0C29  1755
52   0B54'CF    DE   0C29  1756          MOVAL   IGTBUF_ARGL_7,R2        ; Set up DR11-K #5 arg list
          34    11   0C2E  1757          BRB     IGTBUF_COMMON
                     0C30  1758
                     0C30  1759 LA_A_IGTBUF_6:
                0FFC 0C30  1760          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0C32  1761
52   0B48'CF    DE   0C32  1762          MOVAL   IGTBUF_ARGL_6,R2        ; Set up DR11-K #4 arg list
          2B    11   0C37  1763          BRB     IGTBUF_COMMON
                     0C39  1764
                     0C39  1765 LA_A_IGTBUF_5:
                0FFC 0C39  1766          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0C3B  1767
52   0B3C'CF    DE   0C3B  1768          MOVAL   IGTBUF_ARGL_5,R2        ; Set up DR11-K #3 arg list
          22    11   0C40  1769          BRB     IGTBUF_COMMON
                     0C42  1770
                     0C42  1771 LA_A_IGTBUF_4:
                0FFC 0C42  1772          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0C44  1773
52   0B30'CF    DE   0C44  1774          MOVAL   IGTBUF_ARGL_4,R2        ; Set up DR11-K #2 arg list
          19    11   0C49  1775          BRB     IGTBUF_COMMON
```

```
                                 0C4B  1776
                                 0C4B  1777 LA_A_IGTBUF_3:
                          0FFC   0C4B  1778          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                 0C4D  1779
          52    0B24'CF    DE   0C4D  1780          MOVAL   IGTBUF_ARGL_3,R2        ; Set up DR11-K #1 arg list
                      10    11   0C52  1781          BRB     IGTBUF_COMMON
                                 0C54  1782
                                 0C54  1783 LA_A_IGTBUF_2:
                          0FFC   0C54  1784          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                 0C56  1785
          52    0B18'CF    DE   0C56  1786          MOVAL   IGTBUF_ARGL_2,R2        ; Set up AA11-K arg list
                      07    11   0C5B  1787          BRB     IGTBUF_COMMON
                                 0C5D  1788
                                 0C5D  1789 LA_A_IGTBUF_1:
                          0FFC   0C5D  1790          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                 0C5F  1791
          52    0B0C'CF    DE   0C5F  1792          MOVAL   IGTBUF_ARGL_1,R2        ; Set up AD11-K arg list
                                 0C64  1793          ;BRB    IGTBUF_COMMON
                                 0C64  1794
                                 0C64  1795 IGTBUF_COMMON:
00000000'GF    62    FA   0C64  1796          CALLG   (R2),G^LPA$IGTBUF       ; Get next buffer to process
          50  08 B2    D0   0C6B  1797          MOVL    @08(R2),R0              ; Save the return status
                01AF   30   0C6F  1798          BSBW    IGTBUF_CHECK            ; See if sweep went correctly
                       04   0C72  1799          RET                             ; Dismiss the AST
```

```
                     0C73  1801          .SBTTL  Error Checking Subroutines
                     0C73  1802 ;++
                     0C73  1803 ; FUNCTIONAL DESCRIPTION:
                     0C73  1804 ;       This set of routines checks for errors returned by the various high
                     0C73  1805 ;       level LPA11-K support routines.  It was decided to isolate them from
                     0C73  1806 ;       the inline code because they greatly diminished the readability of the
                     0C73  1807 ;       text.
                     0C73  1808 ;
                     0C73  1809 ; CALLING SEQUENCE:
                     0C73  1810 ;       They are typically invoked immediately following one of the LPA$...
                     0C73  1811 ;       routines.  Called by BSBW.
                     0C73  1812 ;
                     0C73  1813 ; INPUTS, OUTPUTS, COMPLETION CODES AND SIDE EFFECTS:
                     0C73  1814 ;       The code executes in the context of the portion of the main code which
                     0C73  1815 ;       called it.  Forming subroutines of these is an attempt to avoid the
                     0C73  1816 ;       space overhead which would be incurred were they implemented as macros.
                     0C73  1817 ;       Therefore, there are no explict input/output parameters or returns.  If
                     0C73  1818 ;       an error is found, an appropriate message is formed and the program is
                     0C73  1819 ;       exited.  It is the responsibility of the programmer to see that each
                     0C73  1820 ;       routine is called uniformly throughout the program.
                     0C73  1821 ;--
                     0C73  1822
                     0C73  1823 CONFIG_CHECK:                            ; Here when we determine I/O bus configurati
     57  2314'CF  2310'CF  81  0C73  1824          ADDB3   LA_B_ADS,LA_B_DAS,R7    ; See...
         57  230C'CF  80  0C7B  1825          ADDB2   LA_B_DRS,R7             ; ...if we can see any devices at all
                 01  13  0C80  1826          BEQL    10$
                 05  0C82  1827          RSB
                     0C83  1828 10$:
         01EE'CF  14  DO  0C83  1829          MOVL    #SS$_BADPARAM&<^C^B111>!STS$K_SEVERE,STATUS
             05F1'CF  DF  0C88  1830          PUSHAL  NAKED_ERR_MSG           ; Nothing configured, must be busted
                 01  DD  0C8C  1831          PUSHL   #1
         00741134 8F  DD  0C8E  1832          PUSHL   #UETP$_TEXT!STS$K_SEVERE
                 03  DD  0C94  1833          PUSHL   #3
              03E6  31  0C96  1834          BRW     ERROR_EXIT
                     0C99  1835
                     0C99  1836
                     0C99  1837
                     0C99  1838 SETADC_CHECK:                            ; Here when A-to-D channel info set up
      01 2324'CF  E9  0C99  1839          BLBC    LA_L_IND,10$            ; No error message if call succeeded
                 05  0C9E  1840          RSB
                     0C9F  1841 10$:
                     0C9F  1842          $FAO_S  CTRSTR = AD11K_ERR_MSG,- ; Yell if it failed (impossible, but)
                     0C9F  1843                  OUTLEN = BUFFER_PTR,-
                     0C9F  1844                  OUTBUF = FAO_BUF,-
                     0C9F  1845                  P1     = MU_MCODE
             000C'CF  DF  0CB6  1846          PUSHAL  BUFFER_PTR
                 01  DD  0CBA  1847          PUSHL   #1
         00741132 8F  DD  0CBC  1848          PUSHL   #UETP$_TEXT!STS$K_ERROR
                 03  DD  0CC2  1849          PUSHL   #3
              03B8  31  0CC4  1850          BRW     ERROR_EXIT
```

```
                                 0CC7  1852 LOADMC_CHECK:                                 ; Here when mcode just loaded
                01 2324'CF    E9 0CC7  1853          BLBC    LA_L_IND,10$                ; Microcode loaded correctly?
                               05 0CCC  1854          RSB                                 ; Return if all is well
                                 0CCD  1855 10$:
 56  0238'CF  00000238'8F    C1 0CCD  1856          ADDL3   #UNIT_LIST,UNIT_LIST,R6     ; Point to unit data block
                            02 8A 0CD7  1857          BICB2   #UETUNT$M_TESTABLE,-        ; This unit is no longer testable
                         0B A6        0CD9  1858                  UETUNT$B_FLAGS(R6)      ; (LOADMC_CHECK called in ONE-SHOT mode)
          2324'CF    0334 8F    B1 0CDB  1859          CMPW    #SS$_DEVREQERR,LA_L_IND     ; Is this...
                         21    13 0CE2  1860          BEQL    20$
          2324'CF    0054 8F    B1 0CE4  1861          CMPW    #SS$_CTRLERR,LA_L_IND       ; ...an error code which...
                         18    13 0CEB  1862          BEQL    20$
          2324'CF    032C 8F    B1 0CED  1863          CMPW    #SS$_DEVCMDERR,LA_L_IND     ; ...yields additional info?
                         0F    13 0CF4  1864          BEQL    20$                         ; BR if it is
               54    032D'CF    DE 0CF6  1865          MOVAL   LOAD_MISC_ERR_MSG,R4        ; It's not - give a dumb message
                            50 D4 0CFB  1866          CLRL    R0                          ; Supply dummies for $FAO below
                            51 D4 0CFD  1867          CLRL    R1
                            52 D4 0CFF  1868          CLRL    R2
                            53 D4 0D01  1869          CLRL    R3
                            19 11 0D03  1870          BRB     30$
                                 0D05  1871 20$:
               54    02B0'CF    DE 0D05  1872          MOVAL   LOAD_BAD_ERR_MSG,R4         ; Give msg with additional info
               50    2328'CF    9A 0D0A  1873          MOVZBL  LA_L_IERROR+0,R0            ; Get LPA11-K Control Out register...
               51    2329'CF    9A 0D0F  1874          MOVZBL  LA_L_IERROR+1,R1            ; ...Status Out register...
               52    232A'CF    9A 0D14  1875          MOVZBL  LA_L_IERROR+2,R2            ; ...low byte Maintenance Status...
               53    232B'CF    9A 0D19  1876          MOVZBL  LA_L_IERROR+3,R3            ; ...and high byte Maintenance Status
                                 0D1E  1877 30$:
                                 0D1E  1878          $FAO_S  CTRSTR = (R4),-             ; Form whichever error message
                                 0D1E  1879                  OUTLEN = BUFFER_PTR,-
                                 0D1E  1880                  OUTBUF = FAO_BUF,-
                                 0D1E  1881                  P1     = MCODE_ADDR,-
                                 0D1E  1882                  P2     = R1,-
                                 0D1E  1883                  P3     = R0,-
                                 0D1E  1884                  P4     = R3,-
                                 0D1E  1885                  P5     = R2
      01EE'CF    2324'CF    D0 0D3B  1886          MOVL    LA_L_IND,STATUS
                 01EE'CF    DD 0D42  1887          PUSHL   STATUS
                 000C'CF    DF 0D46  1888          PUSHAL  BUFFER_PTR
                         01 DD 0D4A  1889          PUSHL   #1
                00741130 8F DD 0D4C  1890          PUSHL   #UETP$_TEXT
6E    03   00    2324'CF    F0 0D52  1891          INSV    LA_L_IND,#STS$V_SEVERITY,#STS$S_SEVERITY,(SP)
                         04 DD 0D59  1892          PUSHL   #4
                      0321  31 0D5B  1893          BRW     ERROR_EXIT
                                 0D5E  1894
                                 0D5E  1895
                                 0D5E  1896
                                 0D5E  1897 XRATE_CHECK:                                 ; Here when clock A rate calculated
                         50    D5 0D5E  1898          TSTL    R0                          ; Error only when function value is 0
                         01    13 0D60  1899          BEQL    10$
                               05 0D62  1900          RSB
                                 0D63  1901 10$:
                 0351'CF    DF 0D63  1902          PUSHAL  BADRATE_ERR_MSG
                         01 DD 0D67  1903          PUSHL   #1
                00741132 8F DD 0D69  1904          PUSHL   #UETP$_TEXT!STS$K_ERROR
                         03 DD 0D6F  1905          PUSHL   #3
                      030B  31 0D71  1906          BRW     ERROR_EXIT
```

```
                                 0D74  1908 CLOCKA_CHECK:                                  ; Here when real time clock gets preset
                01 2324'CF       E9  0D74  1909          BLBC    LA_L_IND,10$             ; Any error?
                                 05  0D79  1910          RSB                              ; Return if not
                                     0D7A  1911 10$:
                                     0D7A  1912          $FAO_S  CTRSTR = CLOCK_ERR_MSG,-
                                     0D7A  1913                  OUTLEN = BUFFER_PTR,-
                                     0D7A  1914                  OUTBUF = FAO_BUF,-
                                     0D7A  1915                  P1     = MCODE_ADDR
        01EE'CF  2324'CF         DO  0D91  1916          MOVL    LA_L_IND,STATUS
                 01EE'CF         DD  0D98  1917          PUSHL   STATUS
                 000C'CF         DF  0D9C  1918          PUSHAL  BUFFER_PTR
                      01         DD  0DA0  1919          PUSHL   #1
                00741130         8F  DD  0DA2  1920      PUSHL   #UETP$_TEXT
    6E  03  00   2324'CF         F0  0DA8  1921          INSV    LA_L_IND,#STS$V_SEVERITY,#STS$S_SEVERITY,(SP)
                      04         DD  0DAF  1922          PUSHL   #4
                    02CB         31  0DB1  1923          BRW     ERROR_EXIT
                                     0DB4  1924
                                     0DB4  1925
                                     0DB4  1926
                                     0DB4  1927 SETIBF_CHECK:                               ; Here when buffer list initialized
                01 08 B2         E9  0DB4  1928          BLBC    @08(R2),10$              ; BR if some error was found
                                 05  0DB8  1929          RSB
                                     0DB9  1930 10$:
                                     0DB9  1931          $FAO_S  CTRSTR = SETUP_ERR_MSG,- ; Complain if we found an error
                                     0DB9  1932                  OUTLEN = BUFFER_PTR,-
                                     0DB9  1933                  OUTBUF = FAO_BUF,-
                                     0DB9  1934                  P1     = MCODE_ADDR
                 000C'CF         DF  0DD0  1935          PUSHAL  BUFFER_PTR               ; ...
                      01         DD  0DD4  1936          PUSHL   #1                       ; ...
                00741132         8F  DD  0DD6  1937      PUSHL   #UETP$_TEXT!STS$K_ERROR  ; ...
                      03         DD  0DDC  1938          PUSHL   #3                       ; ...
                    029E         31  0DDE  1939          BRW     ERROR_EXIT               ; We can't continue
                                     0DE1  1940
                                     0DE1  1941
                                     0DE1  1942
                                     0DE1  1943 SWEEP_CHECK:                                ; Here when returning from sweeps
                01 2324'CF       E9  0DE1  1944          BLBC    LA_L_IND,10$             ; Any indication of an error?
                                 05  0DE6  1945          RSB                              ; No, return
                                     0DE7  1946 10$:
                                     0DE7  1947          ;DECB   SWEEP_COUNT              ; This sweep failed, don't wait for it
                                     0DE7  1948          $FAO_S  CTRSTR = SWEEP_ERR_MSG,- ; Complain about the error
                                     0DE7  1949                  OUTLEN = BUFFER_PTR,-
                                     0DE7  1950                  OUTBUF = FAO_BUF,-
                                     0DE7  1951                  P1     = MCODE_ADDR
        01EE'CF  2324'CF         DO  0DFE  1952          MOVL    LA_L_IND,STATUS          ; Supply an exit status
                 01EE'CF         DD  0E05  1953          PUSHL   STATUS
                 000C'CF         DF  0E09  1954          PUSHAL  BUFFER_PTR
                      01         DD  0E0D  1955          PUSHL   #1
                00741130         8F  DD  0E0F  1956      PUSHL   #UETP$_TEXT
    6E  03  00   2324'CF         F0  0E15  1957          INSV    LA_L_IND,#STS$V_SEVERITY,#STS$S_SEVERITY,(SP)
                      04         DD  0E1C  1958          PUSHL   #4
                    025E         31  0E1E  1959          BRW     ERROR_EXIT
```

```
                     0E21  1961 IGTBUF_CHECK:                          ; See if buffer got filled correctly
                     0E21  1962 IWTBUF_CHECK:                          ; Here after waiting for a buffer to be fill
           50   D5   0E21  1963          TSTL    R0                    ; Did the sweep terminate?
           01   19   0E23  1964          BLSS    10$                   ; BR if it did
                05   0E25  1965          RSB                           ; Return if it hasn't yet
                     0E26  1966 10$:
     51    04 A2   D0   0E26  1967          MOVL    04(R2),R1             ; Point to IOSB for this sweep
           61   B5   0E2A  1968          TSTW    (R1)                  ; Is the sweep still active?
           01   12   0E2C  1969          BNEQ    20$                   ; BR if not
                05   0E2E  1970          RSB                           ; We do nothing if it is
                     0E2F  1971 20$:
        22B8'CF   97   0E2F  1972          DECB    SWEEP_COUNT           ; This sweep has terminated, don't wait
                     0E33  1973                                        ; (This has no effect unless multiuser)
         01 61    E9   0E33  1974          BLBC    (R1),30$              ; Any indication of an error?
                05   0E36  1975          RSB                           ; No, return
                     0E37  1976 30$:
  01EE'CF    61   3C   0E37  1977          MOVZWL  (R1),STATUS           ; Supply an exit status
61    0334 8F   B1   0E3C  1978          CMPW    #SS$_DEVREQERR,(R1)   ; Is this...
           1D   13   0E41  1979          BEQL    40$
61    0054 8F   B1   0E43  1980          CMPW    #SS$_CTRLERR,(R1)     ; ...an error code which...
           16   13   0E48  1981          BEQL    40$
61    032C 8F   B1   0E4A  1982          CMPW    #SS$_DEVCMDERR,(R1)   ; ...yields additional info?
           0F   13   0E4F  1983          BEQL    40$                   ; BR if it is
54    03FD'CF   DE   0E51  1984          MOVAL   IWTBUF_MISC_ERR_MSG,R4 ; It's not - give a dumb message
           50   D4   0E56  1985          CLRL    R0                    ; Supply dummies for $FAO below
           51   D4   0E58  1986          CLRL    R1
           52   D4   0E5A  1987          CLRL    R2
           53   D4   0E5C  1988          CLRL    R3
           20   11   0E5E  1989          BRB     50$
                     0E60  1990 40$:
     54  042C'CF   DE   0E60  1991          MOVAL   IWTBUF_BAD_ERR_MSG,R4 ; Give msg with additional info
       50    04 A1   9A   0E65  1992          MOVZBL  04(R1),R0             ; Get LPA11-K Control Out register...
       52    06 A1   9A   0E69  1993          MOVZBL  06(R1),R2             ; ...low byte Maintenance Status...
       53    07 A1   9A   0E6D  1994          MOVZBL  07(R1),R3             ; ...high byte Maintenance Status...
       51    05 A1   9A   0E71  1995          MOVZBL  05(R1),R1             ; ...and Status Out register...
       51    A3 8F   91   0E75  1996          CMPB    #LA_K_OVERRUN,R1      ; Error was buffer overrun/underrun?
             05   12   0E79  1997          BNEQ    50$                   ; BR if it was something else
     54  04AC'CF   DE   0E7B  1998          MOVAL   OVERRUN_ERR_MSG,R4    ; Supply specific error msg if it was
                     0E80  1999 50$:
                     0E80  2000          $FAO_S  CTRSTR = (R4),-       ; Form whichever error message
                     0E80  2001                  OUTLEN = BUFFER_PTR,-
                     0E80  2002                  OUTBUF = FAO_BUF,-
                     0E80  2003                  P1     = MCODE_ADDR,-
                     0E80  2004                  P2     = R1,-
                     0E80  2005                  P3     = R0,-
                     0E80  2006                  P4     = R3,-
                     0E80  2007                  P5     = R2
        01EE'CF   DD   0E9D  2008          PUSHL   STATUS
        000C'CF   DF   0EA1  2009          PUSHAL  BUFFER_PTR
             01   DD   0EA5  2010          PUSHL   #1
   00741130 8F   DD   0EA7  2011          PUSHL   #UETP$_TEXT
6E 03 00 01EE'CF   F0   0EAD  2012          INSV    STATUS,#STS$V_SEVERITY,#STS$S_SEVERITY,(SP)
             04   DD   0EB4  2013          PUSHL   #4
            01C6  31   0EB6  2014          BRW     ERROR_EXIT
```

```
                    0EB9  2016 RLSBUF_CHECK:                                ; Here after buffers are made again availabl
          01 2324'CF  E9  0EB9  2017        BLBC    LA_L_IND,10$            ; Any indication of an error?
                      05  0EBE  2018        RSB                             ; No, return
                          0EBF  2019 10$:
                          0EBF  2020        $FAO_S  CTRSTR = RELEASE_ERR_MSG,- ; Yes, complain
                          0EBF  2021                OUTLEN = BUFFER_PTR,-
                          0EBF  2022                OUTBUF = FAO_BUF,-
                          0EBF  2023                P1     = MCODE_ADDR
    01EE'CF   2324'CF     D0  0ED6  2024        MOVL    LA_L_IND,STATUS         ; Supply an exit status
        000C'CF           DF  0EDD  2025        PUSHAL  BUFFER_PTR
              01          DD  0EE1  2026        PUSHL   #1
      00741132 8F         DD  0EE3  2027        PUSHL   #UETP$_TEXT!STS$K_ERROR
              03          DD  0EE9  2028        PUSHL   #3
            0191          31  0EEB  2029        BRW     ERROR_EXIT
```

```
                     0EEE  2031          .SBTTL  Timer Expiration Routine
                     0EEE  2032 ;++
                     0EEE  2033 ; FUNCTIONAL DESCRIPTION:
                     0EEE  2034 ;       This routine will be called when the timer to mark the end of the test
                     0EEE  2035 ;       goes off.
                     0EEE  2036 ;
                     0EEE  2037 ; CALLING SEQUENCE:
                     0EEE  2038 ;       Called via AST at $SETIMR expiration.
                     0EEE  2039 ;
                     0EEE  2040 ; INPUT PARAMETERS:
                     0EEE  2041 ;       NONE
                     0EEE  2042 ;
                     0EEE  2043 ; IMPLICIT INPUTS:
                     0EEE  2044 ;       NONE
                     0EEE  2045 ;
                     0EEE  2046 ; OUTPUT PARAMETERS:
                     0EEE  2047 ;       NONE
                     0EEE  2048 ;
                     0EEE  2049 ; IMPLICIT OUTPUTS:
                     0EEE  2050 ;       NONE
                     0EEE  2051 ;
                     0EEE  2052 ; COMPLETION CODES:
                     0EEE  2053 ;       NONE
                     0EEE  2054 ;
                     0EEE  2055 ; SIDE EFFECTS:
                     0EEE  2056 ;       Sets a flag to indicate timer expiration.
                     0EEE  2057 ;
                     0EEE  2058 ;--
                     0EEE  2059
                     0EEE  2060 TIME_OUT:
                OFFC 0EEE  2061          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0EF0  2062
0002'CF    02    A8 0EF0  2063          BISW2   #TEST_OVERM,FLAG        ; Indicate end of pass or of test
                   04 0EF5  2064          RET
```

```
                        .SBTTL  System Service Exception Handler
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;       This routine is executed if a software or hardware exception occurs or
                ;       if a LIB$SIGNAL system service is used to output a message.
                ;
                ; CALLING SEQUENCE:
                ;       Entered via an exception from the system
                ;
                ; INPUT PARAMETERS:
                ;       ERROR_COUNT     = previous cumulative error count
                ;
                ;          AP ---->     +---------------+
                ;                       |       2       |
                ;                       |---------------|
                ;                       | SIGNL ARY PNT |
                ;                       |---------------|
                ;                       | MECH  ARY PNT |  ---------
                ;                       |---------------|         ^
                ;                       |       4       |         |
                ;                       |---------------|         |
                ;                       | ESTABLISH FP  |         |
                ;                       |---------------|         |
                ;                       |    DEPTH      |  Mechanism Array
                ;                       |---------------|         |
                ;                       |      R0       |         |
                ;                       |---------------|         |
                ;                       |      R1       |         v
                ;                       |---------------|  ---------
                ;                       |       N       |         ^
                ;                       |---------------|         .
                ;                       | CONDITION NAME|         .
                ;                       |---------------|         .
                ;                       | N-3 ADDITIONAL|    Signal Array
                ;                       | LONG WORD ARGS|         .
                ;                       |---------------|         .
                ;                       |      PC       |         .
                ;                       |---------------|         .
                ;                       |      PSL      |         v
                ;                       +---------------+  ---------
                ; IMPLICIT INPUTS:
                ;       NONE
                ;
                ; OUTPUT PARAMETERS:
                ;       NONE
                ;
                ; IMPLICIT OUTPUTS:
                ;       NONE
                ;
                ; COMPLETION CODES:
                ;       SS$_NORMAL if it's a UETP condition or RMS error.
                ;       Error status from exception, otherwise.
                ;
                ; SIDE EFFECTS:
                ;       May branch to ERROR_EXIT.
                ;       May print a message.
                ;--
```

```
                        0EF6  2123 SSERROR:
                   0FFC 0EF6  2124         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0EF8  2125
                        0EF8  2126         $SETAST_S ENBFLG = #0                ; Disable AST delivery
                     01 DD    0F01  2127         PUSHL   #1                   ; Assume ASTs were enabled
                  50 09 D1    0F03  2128         CMPL    S^#SS$_WASSET,R0     ; Were ASTs enabled?
                     02 13    0F06  2129         BEQL    10$                  ; BR if they were
                     6E D4    0F08  2130         CLRL    (SP)                 ; Set ASTs to remain disabled
                        0F0A  2131 10$:
                        0F0A  2132         $SETSFM_S ENBFLG = #0                ; Disable SS failure mode
                     01 DD    0F13  2133         PUSHL   #1                   ; Assume SS failure mode was enabled
                  50 09 D1    0F15  2134         CMPL    S^#SS$_WASSET,R0     ; Was SS failure mode enabled?
                     02 13    0F18  2135         BEQL    20$                  ; BR if it was
                     6E D4    0F1A  2136         CLRL    (SP)                 ; Set SS failure mode to remain off
                        0F1C  2137 20$:
               56 04 AC D0    0F1C  2138         MOVL    CHF$L_SIGARGLST(AP),R6  ; Get the signal array pointer
               59 04 A6 7D    0F20  2139         MOVQ    CHF$L_SIG_NAME(R6),R9   ; Get NAME in R9 and ARG1 in R10
                     10 ED    0F24  2140         CMPZV   #STS$V_FAC_NO,-      ; Is this a message from LIB$SIGNAL?
                        0C          0F26  2141                 #STS$S_FAC_NO,-
         00000074 8F    59          0F27  2142                 R9,#UETP$_FACILITY
                     14 12    0F2D  2143         BNEQ    30$                  ; BR if this is not a UETP exception
                  66 02 C2    0F2F  2144         SUBL2   #2,CHF$L_SIG_ARGS(R6)   ; Drop the PC and PSL
                        0F32  2145         $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6) ; Print the message
                     21 11    0F41  2146         BRB     40$                  ; Restore ASTs and SS fail mode
                        0F43  2147 30$:
   59    0000045C 8F    D1    0F43  2148         CMPL    #SS$_SSFAIL,R9       ; RMS failures are SysSvc failures
                     32 12    0F4A  2149         BNEQ    50$                  ; BR if this can't be an RMS failure
                     10 ED    0F4C  2150         CMPZV   #STS$V_FAC_NO,-      ; Is it an RMS failure?
                        0C          0F4E  2151                 #STS$S_FAC_NO,-
                  01    5A          0F4F  2152                 R10,#RMS$_FACILITY
                     2B 12    0F51  2153         BNEQ    50$                  ; BR if not
   5A    F0000000 8F    CA    0F53  2154         BICL2   #^XF0000000,R10      ; Strip control bits from status code
            08 A6    04 39    0F5A  2155         MATCHC  #4,CHF$L_SIG_ARG1(R6),- ; Is it an RMS failure for which...
                     14          0F5E  2156                 #NRAT_LENGTH,-
                  004D'CF        0F5F  2157                 NO_RMS_AST_TABLE     ; ...no AST can be delivered?
                     1A 13    0F62  2158         BEQL    50$                  ; BR if so - must give error here
                        0F64  2159 40$:
                     01 BA    0F64  2160         POPR    #^M<R0>              ; Restore SS failure mode...
                        0F66  2161         $SETSFM_S ENBFLG = R0                ; ...
                     01 BA    0F6F  2162         POPR    #^M<R0>              ; Restore AST enable...
                        0F71  2163         $SETAST_S ENBFLG = R0                ; ...
                  50 01 D0    0F7A  2164         MOVL    S^#SS$_NORMAL,R0     ; Supply a standard status for exit
                     04 D7    0F7D  2165         RET                          ; Resume processing (or goto RMS_ERROR)
                        0F7E  2166 50$:
         01EE'CF    59 D0    0F7E  2167         MOVL    R9,STATUS            ; Save the status
                     58 D4    0F83  2168         CLRL    R8                   ; Assume for now it's not SS failure
   59    0000045C 8F    D1    0F85  2169         CMPL    #SS$_SSFAIL,R9       ; But is it a System Service failure?
                     38 12    0F8C  2170         BNEQ    70$                  ; BR if not - no special case message
                        0F8E  2171         $GETMSG_S MSGID = R10,-      ; Get SS failure code associated text
                        0F8E  2172                   MSGLEN = BUFFER_PTR,-
                        0F8E  2173                   BUFADR = FAO_BUF,-
                        0F8E  2174                   FLAGS  = #14,-
                        0F8E  2175                   OUTADR = MSG_BLOCK
                  021D'CF    95 0FA5  2176         TSTB    MSG_BLOCK+1          ; Get FAO arg count for SS failure code
                     16 13    0FA9  2177         BEQL    60$                  ; Don't use $GETMSG if no $FAO args...
                  000C'CF    DF 0FAB  2178         PUSHAL  BUFFER_PTR           ; ...else build up...
                     01 DD    0FAF  2179         PUSHL   #1                   ; ...a message describing...
```

```
                00741130 8F    DD  0FB1  2180          PUSHL   #UETP$_TEXT                     ; ...why the System Service failed
                    00   5A    F0  0FB7  2181          INSV    R10,#STS$V_SEVERITY,-           ; Give the message...
                    6E   03        0FBA  2182                  #STS$S_SEVERITY,(SP)            ; ...the correct severity code
                    58   03    D0  0FBC  2183          MOVL    #3,R8                           ; Count the number of args we pushed
                         05    11  0FBF  2184          BRB     70$
                                   0FC1  2185 60$:
                         5A    DD  0FC1  2186          PUSHL   R10                             ; Save SS failure code
                    58   01    D0  0FC3  2187          MOVL    #1,R8                           ; Count the number of args we pushed
                                   0FC6  2188 70$:
                57  66   04    C5  0FC6  2189          MULL3   #4,CHF$L_SIG_ARGS(R6),R7 ; Convert longwords to bytes
                    5E   57    C2  0FCA  2190          SUBL2   R7,SP                           ; Save the current signal array...
            6E  04  A6   57    28  0FCD  2191          MOVC3   R7,CHF$L_SIG_NAME(R6),(SP) ; ...on the stack
                7E  66   58    C1  0FD2  2192          ADDL3   R8,CHF$L_SIG_ARGS(R6),-(SP) ; Push the current arg count
                      00A6     31  0FD6  2193          BRW     ERROR_EXIT
```

```
                        0FD9  2195          .SBTTL  RMS Error Handler
                        0FD9  2196 ;++
                        0FD9  2197 ; FUNCTIONAL DESCRIPTION:
                        0FD9  2198 ;       This routine handles error returns from RMS calls.
                        0FD9  2199 ;
                        0FD9  2200 ; CALLING SEQUENCE:
                        0FD9  2201 ;       Called by RMS when a file processing error is found.
                        0FD9  2202 ;
                        0FD9  2203 ; INPUT PARAMETERS:
                        0FD9  2204 ;       The FAB or RAB associated with the RMS call.
                        0FD9  2205 ;
                        0FD9  2206 ; IMPLICIT INPUTS:
                        0FD9  2207 ;       NONE
                        0FD9  2208 ;
                        0FD9  2209 ; OUTPUT PARAMETERS:
                        0FD9  2210 ;       NONE
                        0FD9  2211 ;
                        0FD9  2212 ; IMPLICIT OUTPUTS:
                        0FD9  2213 ;       Error message
                        0FD9  2214 ;
                        0FD9  2215 ; COMPLETION CODES:
                        0FD9  2216 ;       NONE
                        0FD9  2217 ;
                        0FD9  2218 ; SIDE EFFECTS:
                        0FD9  2219 ;       Program may exit, depending on severity of the error.
                        0FD9  2220 ;
                        0FD9  2221 ;--
                        0FD9  2222
                        0FD9  2223 RMS_ERROR:
                  0FFC  0FD9  2224          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0FDB  2225
          56  04 AC    D0  0FDB  2226          MOVL    4(AP),R6                ; See whether we're dealing with...
              66  03    91  0FDF  2227          CMPB    #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
                  16    12  0FE2  2228          BNEQ    10$                     ; BR if it's a RAB
          57  01ED'CF    DE  0FE4  2229          MOVAL   FILE,R7                 ; FAB-specific code:  text string...
              58  56    D0  0FE9  2230          MOVL    R6,R8                   ; ...address of FAB...
                0C A6   DD  0FEC  2231          PUSHL   FAB$L_STV(R6)           ; ...STV field for error...
                08 A6   DD  0FEF  2232          PUSHL   FAB$L_STS(R6)           ; ...STS field for error...
01EE'CF         08 A6   D0  0FF2  2233          MOVL    FAB$L_STS(R6),STATUS    ; ...and save the error code
                  15    11  0FF8  2234          BRB     COMMON                  ; FAB and RAB share other code
                        0FFA  2235 10$:
          57  01F9'CF    DE  0FFA  2236          MOVAL   RECORD,R7               ; RAB-specific code:  text string...
              58  3C A6   D0  0FFF  2237          MOVL    RAB$L_FAB(R6),R8        ; ...address of associated FAB...
                0C A6   DD  1003  2238          PUSHL   RAB$L_STV(R6)           ; ...STV field for error...
                08 A6   DD  1006  2239          PUSHL   RAB$L_STS(R6)           ; ...STS field for error...
01EE'CF         08 A6   D0  1009  2240          MOVL    RAB$L_STS(R6),STATUS    ; ...and save the error code
                        100F  2241 COMMON:
              5A  34 A8   9A  100F  2242          MOVZBL  FAB$B_FNS(R8),R10       ; Get the file name size
                        1013  2243          $FAO_S  CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
                        1013  2244                  OUTLEN = BUFFER_PTR,-
                        1013  2245                  OUTBUF = FAO_BUF,-
                        1013  2246                  P1     = R7,-
                        1013  2247                  P2     = R10,-
                        1013  2248                  P3     = FAB$L_FNA(R8)
                000C'CF   DF  102D  2249          PUSHAL  BUFFER_PTR              ; ...and arguments for ERROR_EXIT...
                      01  DD  1031  2250          PUSHL   #1                      ; ...
              00741130  8F  DD  1033  2251          PUSHL   #UETP$_TEXT             ; ...
```

```
                00    EF  1039  2252          EXTZV   #STS$V_SEVERITY,-
                03        103B  2253                  #STS$S_SEVERITY,-
    59  01EE'CF          103C  2254                  STATUS,R9               ; ...get the severity code...
        6E    59    8F  1040  2255          BISB2   R9,(SP)                 ; ...and add it into the signal name
              05    DD  1043  2256          PUSHL   #5                      ; Current arg count
          0037      31  1045  2257          BRW     ERROR_EXIT
```

```
                     1048  2259          .SBTTL  CTRL/C Handler
                     1048  2260 ;++
                     1048  2261 ; FUNCTIONAL DESCRIPTION:
                     1048  2262 ;       This routine handles CTRL/C AST's
                     1048  2263 ;
                     1048  2264 ; CALLING SEQUENCE:
                     1048  2265 ;       Called via AST
                     1048  2266 ;
                     1048  2267 ; INPUT PARAMETERS:
                     1048  2268 ;       NONE
                     1048  2269 ;
                     1048  2270 ; IMPLICIT INPUTS:
                     1048  2271 ;       NONE
                     1048  2272 ;
                     1048  2273 ; OUTPUT PARAMETERS:
                     1048  2274 ;       NONE
                     1048  2275 ;
                     1048  2276 ; IMPLICIT OUTPUTS:
                     1048  2277 ;       NONE
                     1048  2278 ;
                     1048  2279 ; COMPLETION CODES:
                     1048  2280 ;       NONE
                     1048  2281 ;
                     1048  2282 ; SIDE EFFECTS:
                     1048  2283 ;       NONE
                     1048  2284 ;
                     1048  2285 ;--
                     1048  2286
                     1048  2287 CCASTHAND:
                OFFC 1048  2288          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     104A  2289
          00A3'CF    DF   104A  2290          PUSHAL  CNTRLCMSG                       ; Set message pointer
                01   DD   104E  2291          PUSHL   #1                              ; Set arg count
      00741130 8F    DD   1050  2292          PUSHL   #UETP$_TEXT!STS$K_WARNING       ; Set signal name
                00   DD   1056  2293          PUSHL   #0                              ; Indicate an abnormal termination
          0148'CF    DF   1058  2294          PUSHAL  PROCESS_NAME                    ; ...
                02   DD   105C  2295          PUSHL   #2                              ; ...
      007410E0 8F    DD   105E  2296          PUSHL   #UETP$_ABENDD!STS$K_WARNING     ; ...
00000000'GF    07   FB   1064  2297          CALLS   #7,G^LIB$SIGNAL                 ; Output the message
                D0   106B  2298          MOVL    #<STS$M_INHIB_MSG!-             ; Set the exit status
                     106C  2299                  SS$_CONTROLC-=
                     106C  2300                  STS$K_SUCCESS+STS$K_WARNING>,-
01EE'CF    10000650 8F      106C  2301                  STATUS
                     1074  2302          $EXIT_S STATUS                          ; Terminate program cleanly
```

```
                        107F  2304          .SBTTL  Error Exit
                        107F  2305 ;++
                        107F  2306 ; FUNCTIONAL DESCRIPTION:
                        107F  2307 ;       This routine prints an error message and exits.
                        107F  2308 ;
                        107F  2309 ; CALLING SEQUENCE:
                        107F  2310 ;       MOVx  error status value,STATUS
                        107F  2311 ;       PUSHx error specific information on the stack
                        107F  2312 ;       PUSHL current argument count
                        107F  2313 ;       BRW   ERROR_EXIT
                        107F  2314 ;
                        107F  2315 ; INPUT PARAMETERS:
                        107F  2316 ;       Arguments to LIB$SIGNAL, as above
                        107F  2317 ;
                        107F  2318 ; IMPLICIT INPUTS:
                        107F  2319 ;       NONE
                        107F  2320 ;
                        107F  2321 ; OUTPUT PARAMETERS:
                        107F  2322 ;       Message to SYS$OUTPUT and SYS$ERROR
                        107F  2323 ;
                        107F  2324 ; IMPLICIT OUTPUTS:
                        107F  2325 ;       Program exit
                        107F  2326 ;
                        107F  2327 ; COMPLETION CODES:
                        107F  2328 ;       NONE
                        107F  2329 ;
                        107F  2330 ; SIDE EFFECTS:
                        107F  2331 ;       NONE
                        107F  2332 ;
                        107F  2333 ;--
                        107F  2334
                        107F  2335 ERROR_EXIT:
                        107F  2336
                        107F  2337          $SETAST_S ENBFLG = #0              ; ASTs can play havoc with messages
     15 0002'CF    03   E0  1088  2338          BBS     #BEGIN_MSGV,FLAG,10$      ; BR if "begin" msg already printed
                   7E   D4  108E  2339          CLRL    -(SP)                     ; Set the time stamp flag
              000F'CF   DF  1090  2340          PUSHAL  TEST_NAME                 ; Set the test name
                   02   DD  1094  2341          PUSHL   #2                        ; Push the argument count
          00741039 8F   DD  1096  2342          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
    00000000'GF   04   FB  109C  2343          CALLS   #4,G^LIB$SIGNAL           ; Print the startup message
                        10A3  2344 10$:
    0230'CF    08   8E   C1  10A3  2345          ADDL3   (SP)+,#8,ARG_COUNT        ; Get total # args, pop partial count
              01EA'CF   D6  10A9  2346          INCL    ERROR_COUNT               ; Keep running error count
                   00   DD  10AD  2347          PUSHL   #0                        ; Push the time parameter
              0148'CF   DF  10AF  2348          PUSHAL  PROCESS_NAME              ; Push test name...
          000F0002 8F   DD  10B3  2349          PUSHL   #^XF0002                  ; ...arg count...
          007410E2 8F   DD  10B9  2350          PUSHL   #UETP$_ABENDD:STS$K_ERROR ; ...and signal name
              01EA'CF   DD  10BF  2351          PUSHL   ERROR_COUNT               ; Finish off arg list...
              0148'CF   DF  10C3  2352          PUSHAL  PROCESS_NAME              ; ...
          00010002 8F   DD  10C7  2353          PUSHL   #^X10002                  ; ...
          00748022 8F   DD  10CD  2354          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
00000000'GF   0230'CF   FB  10D3  2355          CALLS   ARG_COUNT,G^LIB$SIGNAL    ; Truly bitch
                        10DC  2356
              01EE'CF   D5  10DC  2357          TSTL    STATUS                    ; Did we exit with an error code?
                   09   12  10E0  2358          BNEQ    20$                       ; BR if we did
          007410E2 8F   D0  10E2  2359          MOVL    #UETP$_ABENDD!STS$K_ERROR,- ; Supply a generic one otherwise
              01EE'CF         10E8  2360                  STATUS
```

```
                     10EB  2361 20$:
01EE'CF  10000000 8F CB  10EB  2362          BISL    #STS$M_INHIB_MSG,STATUS ; Don't print messages twice!
                     10F4  2363          $EXIT_S STATUS                   ; Exit in error
```

```
                     10FF  2365          .SBTTL  Exit Handler
                     10FF  2366 ;++
                     10FF  2367 ; FUNCTIONAL DESCRIPTION:
                     10FF  2368 ;       This routine handles cleanup at exit.  If the MODE logical name is
                     10FF  2369 ;       equated to "ONE", the routine will update the test flag in the
                     10FF  2370 ;       UETINIDEV.DAT file depending on the UETUNT$M_TESTABLE flag state in the
                     10FF  2371 ;       UETUNT$B_FLAGS field of the unit block for each unit for the device
                     10FF  2372 ;       under test.
                     10FF  2373 ;
                     10FF  2374 ; CALLING SEQUENCE:
                     10FF  2375 ;       Invoked automatically by $EXIT System Service.
                     10FF  2376 ;
                     10FF  2377 ; INPUT PARAMETERS:
                     10FF  2378 ;       STATUS  contains the exit status.
                     10FF  2379 ;       FLAG    has synchronizing bits.
                     10FF  2380 ;       DDB_RFA contains the RFA of the DDB record for this device in UETINIDEV.
                     10FF  2381 ;
                     10FF  2382 ; IMPLICIT INPUTS:
                     10FF  2383 ;       UNIT_LIST points to the head of a doubly linked circular list of unit
                     10FF  2384 ;                 blocks for the device under test.
                     10FF  2385 ;
                     10FF  2386 ; OUTPUT PARAMETERS:
                     10FF  2387 ;       NONE
                     10FF  2388 ;
                     10FF  2389 ; IMPLICIT OUTPUTS:
                     10FF  2390 ;       Various files are de-accessed, the process name is reset, and any
                     10FF  2391 ;       necessary synchronization with UETPDEV01 is carried out.
                     10FF  2392 ;       If the MODE logical name is equated to "ONE", the routine will update
                     10FF  2393 ;       the test flag in the UETINIDEV.DAT file depending on the
                     10FF  2394 ;       UETUNT$M_TESTABLE flag state in the UETUNT$B_FLAGS field of the unit
                     10FF  2395 ;       block for each unit for the device under test.
                     10FF  2396 ;
                     10FF  2397 ; COMPLETION CODES:
                     10FF  2398 ;       NONE
                     10FF  2399 ;
                     10FF  2400 ; SIDE EFFECTS:
                     10FF  2401 ;       NONE
                     10FF  2402 ;
                     10FF  2403 ;--
                     10FF  2404
                     10FF  2405 EXIT_HANDLER:
                OFFC 10FF  2406          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     1101  2407
                     1101  2408          $SETSFM_S ENBFLG = #0                ; Turn off System Service failure mode
                     110A  2409          $SETAST_S ENBFLG = #0                ; We're finished - no more ASTs
                     1113  2410          $TRNLOG_S LOGNAM = MODE,-            ; Get the run mode
                     1113  2411                    RSLLEN = BUFFER_PTR,-
                     1113  2412                    RSLBUF = FAO_BUF
   0014'CF    20    8A  112C  2413          BICB2   #LC_BITM,BUFFER              ; Convert to upper case
0014'CF    4F 8F    91  1131  2414          CMPB    #^A/O/,BUFFER                ; Is this a one shot?
              03    13  1137  2415          BEQL    10$                          ; BR if yes...
            00B8    31  1139  2416          BRW     END_UPDATE                   ; ...else don't update UETINIDEV.DAT
                     113C  2417 10$:
03 0002'CF    02    E0  113C  2418          BBS     #SAFE_TO_UPDV,FLAG,20$       ; Only update if it's safe
            00AF    31  1142  2419          BRW     END_UPDATE                   ; Else forget it
                     1145  2420 20$:
   5A   2428'CF    DE  1145  2421          MOVAL   INI_RAB,R10                  ; Set the RAB address
```

```
              1E AA     02   90  114A          MOVB    #RAB$C_RFA,RAB$B_RAC(R10) ; Set RFA mode
   10 AA   246C'CF      06   28  114E          MOVC3   #6,DDB_RFA,RAB$W_RFA(R10) ; Set RFA to DDB line
                                 1155          $GET    RAB = (R10)               ; Go back to the DDB record
                 75 50       E9  115E          BLBC    R0,UPDATE_FAILED          ; If failure then forget it
              1E AA     00   90  1161          MOVB    #RAB$C_SEQ,RAB$B_RAC(R10) ; Set back to sequential mode
 5B   0238'CF  00000238'8F   C1  1165          ADDL3   #UNIT_LIST,UNIT_LIST,R11  ; Set the unit block list header
                        59   D4  116F          CLRL    R9                        ; Init a counter
                                 1171 UNIT_LOOP:
                        01   E1  1171          BBC     #UETUNT$V_TESTABLE,-      ; BR if this unit is not testable
              02 0B AB           1173                  UETUNT$B_FLAGS(R11),10$
                        59   D6  1176          INCL    R9                        ; Count testable units
                                 1178 10$:
                 5B     6B   C0  1178          ADDL2   (R11),R11                 ; Next unit block
       00000238'8F      5B   D1  117B          CMPL    R11,#UNIT_LIST            ; Are we full circle in the list?
                        ED   12  1182          BNEQ    UNIT_LOOP                 ; BR if not
                        59   D5  1184          TSTL    R9                        ; Any testable units?
                        12   12  1186          BNEQ    20$                       ; BR if yes...
        0018'CF      4E 8F   90  1188          MOVB    #^A/N/,BUFFER+4           ; ...else disable the DDB record...
                                 118E          $UPDATE RAB = (R10)               ; ...here
                 3C 50       E9  1197          BLBC    R0,UPDATE_FAILED          ; If error then forget it
                                 119A 20$:
                 5B     6B   C0  119A          ADDL2   (R11),R11                 ; Next unit block
       00000238'8F      5B   D1  119D          CMPL    R11,#UNIT_LIST            ; Are we full circle in the list?
                        4E   13  11A4          BEQL    END_UPDATE                ; BR if yes
                                 11A6          $GET    RAB = (R10)               ; Get a record
                 24 50       E9  11AF          BLBC    R0,UPDATE_FAILED          ; If error then forget it
           0014'CF      20   8A  11B2          BICB2   #LC_BITM,BUFFER           ; Convert to uppercase
        0014'CF      55 8F   91  11B7          CMPB    #^A/U/,BUFFER             ; Is it a UCB record?
                        35   12  11BD          BNEQ    END_UPDATE                ; BR if not
                        01   E0  11BF          BBS     #UETUNT$V_TESTABLE,-      ; BR if this unit is testable...
              D6 0B AB           11C1                  UETUNT$B_FLAGS(R11),20$
        0018'CF      4E 8F   90  11C4          MOVB    #^A/N/,BUFFER+4           ; ...else disable the UCB record...
                                 11CA          $UPDATE RAB = (R10)               ; ...here
                 C4 50       E8  11D3          BLBS    R0,20$                    ; Look at the next record if no error
                                 11D6 UPDATE_FAILED:
                 0C AA       DD  11D6          PUSHL   RAB$L_STV(R10)            ; Do a simple message...
                        50   DD  11D9          PUSHL   R0                        ; ...to tell of the failure
                   01B8'CF   DF  11DB          PUSHAL  INIDEV_UPDERR
                        01   DD  11DF          PUSHL   #1
                        00   EF  11E1          EXTZV   #STS$V_SEVERITY,-         ; Copy the severity from RMS status...
            7E    50    03       11E3                  #STS$S_SEVERITY,R0,-(SP)
     6E   00741130 8F        C8  11E6          BISL2   #UETP$_TEXT,(SP)          ; ...to our message
       00000000'GF      05   FB  11ED          CALLS   #5,G^LIB$SIGNAL
                                 11F4 END_UPDATE:
                   0258'CF   D5  11F4          TSTL    BASE_PRIORITY             ; Could we have changed our priority?
                        11   13  11F8          BEQL    10$                       ; BR if not - don't try to reset
                                 11FA          $SETPRI_S PRI = BASE_PRIORITY     ; Restore our base priority
                                 120B 10$:
                        00   E1  120B          BBC     #LA$V_MCVALID,-           ; BR if there was...
            18 233F'CF           120D                  LA_L_DEVDEPEND,20$        ; ...no valid microcode originally
                   2322'CF   DF  1211          PUSHAL  LA_W_NUM                  ; Otherwise, try to...
                   02   01   EF  1215          EXTZV   #LA$V_MCTYPE,#LA$S_MCTYPE,- ; ...
       22B9'CF   233F'CF         1218                  LA_L_DEVDEPEND,MCODE_TYPE
                   22B9'CF   DF  121E          PUSHAL  MCODE_TYPE                ; ...
       00000000'GF      02   FB  1222          CALLS   #2,G^LPA$LOADMC           ; ...reload the original microcode
                                 1229 20$:
                        00   DD  1229          PUSHL   #0                        ; Set the time flag
```

```
                000F'CF    DF   122B  2479          PUSHAL  TEST_NAME                ; Push the test name
                     02    DD   122F  2480          PUSHL   #2                      ; Push arg count
                     00    EF   1231  2481          EXTZV   #STS$V_SEVERITY,-       ; Push the proper exit severity...
                     03         1233  2482                  #STS$S_SEVERITY,-
           7E   01EE'CF         1234  2483                  STATUS,-(SP)
        6E   00741080 8F    C8   1238  2484          BISL2   #UETP$_ENDEDD,(SP)      ; ...and use it in our message code
                     04    DD   123F  2485          PUSHL   #4
                51   5E    D0   1241  2486          MOVL    SP,R1
                          1244  2487          $PUTMSG_S MSGVEC = (R1)          ; Output the message
                          1253  2488          $SETPRN_S PRCNAM = ACNT_NAME     ; Reset the process name
                     04   125E  2489          RET                              ; That's all folks!
                          125F  2490
                          125F  2491          .END    UETLPAK00
```

```
$$.TAB                  = 00002474 R      03
$$.TABEND               = 000024C4 R      03
$$.TMP                  = 00000002
$$.TMP1                 = 00000001
$$.TMP2                 = 0000006A
$$.TMPX                 = 00000016 R      05
$$.TMPX1                = 0000000D
$$T1                    = 00000000
$$T2                    = 00000006
ACNT_NAME                 00000000 R      02
AD11K_ERR_MSG             0000059D R      02
AD_BEG_MSG                0000081E R      02
AD_END_MSG                00000846 R      02
AD_MCODE                  00000279 R      02
AD_TEST                   00000720 R      06
AINTRVL                   000022FA R      03
ALL_SET                   00000432 R      06
ARG_COUNT                 00000230 R      03
BADRATE_ERR_MSG           0000C351 R      02
BAD_CASE_MSG              000006F6 R      02
BASE_PRIORITY             00000258 R      03
BEGIN_MSGM              = 00000008
BEGIN_MSGV              = 00000003
BUFFER                    00000014 R      03
BUFFER_PTR                0000000C R      03
BUFFER_SIZE             = 00000000
CCASTHAND                 00001048 R      06
CHF$L_SIGARGLST         = 000000C4
CHF$L_SIG_ARG1          = 00000008
CHF$L_SIG_ARGS          = 00000000
CHF$L_SIG_NAME          = 00000004
CLOCKA_CHECK              00000D74 R      06
CLOCK_ERR_MSG             00000570 R      02
CLOCK_SPEED             = B7173A51
CNTRLCMSG                 000000A3 R      02
COMMON                    0000100F R      06
COMP_RATE                 00002302 R      03
CONFIG_CHECK              00000C73 R      06
CONFIG_MSG                00000768 R      02
CONTROLLER                00000031 R      02
CONT_DESC                 000001E5 R      02
COUNTED_OUT_MSG           0000072D R      02
CS1                       00000082 R      02
CS3                       00000094 R      02
DA_BEG_MSG                0000086B R      02
DA_END_MSG                00000893 R      02
DA_MCODE                  0000028D R      02
DA_TEST                   00000865 R      06
DDB_RFA                   0000246C R      03
DEAD_CTRLNAME             000000E4 R      02
DEV$V_TRM               = 00000002
DEVDEP_SIZE             = 00000000
DEVDSC                    00000140 R      03
DEVNAM_LEN                0000020A R      03
DEV_NAME                  0000015F R      03
DIB                       0000016E R      03
DIB$B_DEVCLASS          = 00000004
DIB$B_DEVTYPE           = 00000005
DIB$K_LENGTH            = 00000074
DIB$L_DEVDEPEND         = 00000008
DIBBUF                    00000176 R      03
DUMP_MODEM              = 00000020
DUMP_MODEV              = 00000005
DVI$_DEVNAM             = 00000020
EFN2                    = 00000004
END_UPDATE                000011F4 R      06
ERROR_COUNT               000001EA R      03
ERROR_EXIT                0000107F R      06
EXIT_DESC                 00000220 R      03
EXIT_HANDLER              000010FF R      06
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_CR                = 00000001
FAB$V_FILE_MODE         = 00000004
FAB$V_GET               = 00000001
FAB$V_LNM_MODE          = 00000000
FAB$V_PUT               = 00000000
FAB$V_UFO               = 00000011
FAB$V_UPD               = 00000003
FAB$V_UPI               = 00000006
FAB$W_GBC               = 00000048
FAO_BUF                   00000004 R      03
FILE                      000001ED R      02
FIND_IT                   0000021C R      06
FIVE_SECONDS              0000025F R      02
FLAG                      00000002 R      03
FLOAT_PTR                 000022C1 R      03
FOR$CVT_D_TF              ******** X      06
FOUND_IT                  000002B4 R      06
FO_MCODE                  000002A1 R      02
FO_TEST                   000009B2 R      06
IGTBUF_ARGL_1             00000B0C R      02
IGTBUF_ARGL_2             00000B18 R      02
IGTBUF_ARGL_3             00000B24 R      02
IGTBUF_ARGL_4             00000B30 R      02
IGTBUF_ARGL_5             00000B3C R      02
IGTBUF_ARGL_6             00000B48 R      02
IGTBUF_ARGL_7             00000B54 R      02
IGTBUF_CHECK              00000E21 R      06
IGTBUF_COMMON             00000C64 R      06
ILLEGAL_REC               00000151 R      02
IMAGNAM_IOSB              000002AE R      03
IMAGNAM_ITMLST            0000025C R      03
```

```
INADDRESS                       000001FA R      03
INIDEV_UPDERR                   000001B8 R      02
INI_FAB                         000023D8 R      03
INI_RAB                         00002428 R      03
INPUT_ITMLST                    00000072 R      02
IO$M_CTRLCAST                   ********   X    06
IO$_SETMODE                     ********   X    06
ITERATION                       00000214 R      03
IWTBUF_ARGL_1                   00000AEC R      02
IWTBUF_ARGL_2                   00000AFC R      02
IWTBUF_BAD_ERR_MSG              0000042C R      02
IWTBUF_CHECK                    00000E21 R      06
IWTBUF_MISC_ERR_MSG             000003FD R      02
JOB_PRIORITY                =   00000000
JPI$_IMAGNAME               =   00000207
JPI$_PRIB                   =   00000309
JPI_EFN                     =   00000005
LA$K_ADMCODE                =   00000002
LA$K_DAMCODE                =   00000003
LA$K_MRMCODE                =   00000001
LA$S_MCTYPE                 =   00000002
LA$V_AD1                    =   00000005
LA$V_DA                     =   00000007
LA$V_DIO1                   =   00000008
LA$V_MCTYPE                 =   00000001
LA$V_MCVALID                =   00000000
LALOADER_IMAGE                  00000636 R      02
LALOADER_PROC                   00000626 R      02
LAST_FUNCTION               =   00000002
LA_AK_BFNUM1                    0000E578 R      04
LA_AK_BFNUM2                    0000E57C R      04
LA_AK_BFNUM3                    0000E580 R      04
LA_AK_BFNUM4                    0000E584 R      04
LA_AK_BFNUM5                    0000E588 R      04
LA_AK_BFNUM6                    0000E58C R      04
LA_AK_BFNUM7                    0000E590 R      04
LA_AK_BUF11                     00000000 R      04
LA_AK_BUF12                     00001000 R      04
LA_AK_BUF21                     00002000 R      04
LA_AK_BUF22                     00003000 R      04
LA_AK_BUF31                     00004000 R      04
LA_AK_BUF32                     00005000 R      04
LA_AK_BUF41                     00006000 R      04
LA_AK_BUF42                     00007000 R      04
LA_AK_BUF51                     00008000 R      04
LA_AK_BUF52                     00009000 R      04
LA_AK_BUF61                     0000A000 R      04
LA_AK_BUF62                     0000B000 R      04
LA_AK_BUF71                     0000C000 R      04
LA_AK_BUF72                     0000D000 R      04
LA_AK_IBUF1                     0000E000 R      04
LA_AK_IBUF2                     0000E0C8 R      04
LA_AK_IBUF3                     0000E190 R      04
LA_AK_IBUF4                     0000E258 R      04
LA_AK_IBUF5                     0000E320 R      04
LA_AK_IBUF6                     0000E3E8 R      04
LA_AK_IBUF7                     0000E4B0 R      04
LA_A_IGTBUF_1                   00000C5D R      06
LA_A_IGTBUF_2                   00000C54 R      06
LA_A_IGTBUF_3                   00000C4B R      06
LA_A_IGTBUF_4                   00000C42 R      06
LA_A_IGTBUF_5                   00000C39 R      06
LA_A_IGTBUF_6                   00000C30 R      06
LA_A_IGTBUF_7                   00000C27 R      06
LA_B_ADS                        00002310 R      03
LA_B_DAS                        00002314 R      03
LA_B_DRS                        0000230C R      03
LA_B_ICHN                       0000233E R      03
LA_K_LAMSKB                     0000231A R      03
LA_K_OVERRUN                =   00000CA3
LA_L_DEVDEPEND                  0000233F R      03
LA_L_IERROR                     00002328 R      03
LA_L_IND                        00002324 R      03
LA_L_IRATE                      00002330 R      03
LA_L_NBUF                       00002334 R      03
LA_W_DWELL                      0000233A R      03
LA_W_IPRSET                     0000232C R      03
LA_W_LBUF                       0000230A R      03
LA_W_LDELAY                     0000233C R      03
LA_W_MODE                       00002338 R      03
LA_W_NCHN                       00002318 R      03
LA_W_NUM                        00002322 R      03
LC_BITM                     =   00000020
LIB$SIGNAL                      ********   X    06
LITERAL_0                       00000257 R      02
LITERAL_1                       0000025B R      02
LOADMC_CHECK                    00000CC7 R      06
LOAD_BAD_ERR_MSG                000002B0 R      02
LOAD_MISC_ERR_MSG               0000032D R      02
LPA$ADSWP                       ********   X    06
LPA$CLOCKA                      ********   X    06
LPA$DASWP                       ********   X    06
LPA$DISWP                       ********   X    06
LPA$IGTBUF                      ********   X    06
LPA$IWTBUF                      ********   X    06
LPA$LAMSKS                      ********   X    06
LPA$LOADMC                      ********   X    06
LPA$RLSBUF                      ********   X    06
LPA$SETADC                      ********   X    06
LPA$SETIBF                      ********   X    06
LPA$XRATE                       ********   X    06
LPA11K_PRIORITY                 00000241 R      02
M                           =   00000004
MAX_DEV_DESIG               =   0000000A
MAX_PROC_NAME               =   0000000F
MAX_UNIT_DESIG              =   00000005
MCODE_ADDR                      000022BD R      03
MCODE_IMAGLEN                   000002AC R      03
MCODE_IMAGNAM                   0000026C R      03
MCODE_TYPE                      000022B9 R      03
MGT_BUF_SIZE                =   00000032
MODE                            00000041 R      02
MSG_BLOCK                       0000021C R      03
MU_BEG_MSG                      000007DF R      02
```

```
MU_END_MSG                      00000800 R      02
MU_MCODE                        0000026F R      02
MU_TEST                         0000053C R      06
MU_TIME_OUT                   = 00000018
N                             = 00000008
NAKED_ERR_MSG                   000005F1 R      02
NAME_LEN                      = 0000000F
NEEDED_PRIVS                    00000267 R      02
NEW_NODE                        00000240 R      03
NOUNIT_SELECTED                 0000012B R      02
NO_CTRLNAME                     000000C4 R      02
NO_LALOADER                     00000651 R      02
NO_OVRN_ERR_MSG                 000005CB R      02
NO_RMS_AST_TABLE                0000004D R      02
NRAT_LENGTH                   = 00000014
NULL_MSG                        000007DE R      02
NUM_CHANNELS                  = 00000007
ONCE_FOR_EACH                   00000A73 R      06
ONE_SHOTM                     = 00000010
ONE_SHOTV                     = 00000004
ONE_SHOT_TEST                   000009B2 R      06
OTS$CVT_TI_L                    ********   X    06
OUTADDRESS                      00000202 R      03
OVERRUN_ERR_MSG                 000004AC R      02
PAGES                         = 00000001
PASS                            00000218 R      03
PASS_MSG                        00000185 R      02
PMTSIZ                        = 00000019
PRIB_ITMLST                     00000248 R      03
PROCESS_NAME                    00000148 R      03
PROCESS_NAME_FREE             = 0000000B
PROC_CONT_NAME                  0000008B R      06
PROMPT                          00000228 R      02
PRV$V_SETPRI                  = 0000000D
QUAD_STATUS                     000001F2 R      03
RAB$B_PSZ                     = 00000034
RAB$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$L_PBF                     = 00000030
RAB$L_ROP                     = 00000004
RAB$L_STS                     = 00000008
RAB$L_STV                     = 0000000C
RAB$V_PMT                     = 0000001E
RAB$W_RFA                     = 00000010
RAB$W_RSZ                     = 00000022
RAMP_DATA                       000002B8 R      03
RAMP_DATA1                      000012B8 R      03
RAMP_HEIGHT                   = 00000800
RANDOM1                         0000020C R      03
RANDOM2                         00000210 R      03
RECORD                          000001F9 R      02
REC_SIZE                      = 00000028
RELEASE_ERR_MSG                 000003AF R      02
RESTART                         0000053C R      06
RLSBUF                        = 00000002
RLSBUF_ARGL_1                   00000B60 R      02
RLSBUF_ARGL_2                   00000B74 R      02
RLSBUF_ARGL_3                   00000B88 R      02
RLSBUF_ARGL_4                   00000B9C R      02
RLSBUF_ARGL_5                   00000BB0 R      02
RLSBUF_ARGL_6                   00000BC4 R      02
RLSBUF_ARGL_7                   00000BD8 R      02
RLSBUF_CHECK                    00000EB9 R      06
RLSBUF_COMMON                   00000B54 R      06
RLSBUF_SECTION                  00000B05 R      06
RMS$_BCN                        ********   X    02
RMS$_BUSY                       ********   X    02
RMS$_CDA                        ********   X    02
RMS$_FAB                        ********   X    02
RMS$_FACILITY                 = 00000001
RMS$_RAB                        ********   X    02
RMS_ERROR                       00000FD9 R      06
RMS_ERR_STRING                  00000207 R      02
SAFE_TO_UPDM                  = 00000004
SAFE_TO_UPDV                  = 00000002
SEC$M_EXPREG                    ********   X    06
SEC$M_GBL                       ********   X    06
SETADC_CHECK                    00000C99 R      06
SETIBF                        = 00000000
SETIBF_ARGL_1                   00000A44 R      02
SETIBF_ARGL_2                   00000A5C R      02
SETIBF_ARGL_3                   00000A74 R      02
SETIBF_ARGL_4                   00000A8C R      02
SETIBF_ARGL_5                   00000AA4 R      02
SETIBF_ARGL_6                   00000ABC R      02
SETIBF_ARGL_7                   00000AD4 R      02
SETIBF_CHECK                    00000DB4 R      06
SETIBF_COMMON                   00000AFB R      06
SETIBF_SECTION                  00000AAC R      06
SETUP_ERR_MSG                   00000385 R      02
SHR$_ABENDD                   = 000010E0
SHR$_BEGIND                   = 00001038
SHR$_ENDEDD                   = 00001080
SHR$_OPENIN                   = 00001098
SHR$_TEXT                     = 00001130
SS$_ABORT                     = 0000002C
SS$_BADPARAM                  = 00000014
SS$_CONTROLC                  = 00000651
SS$_CTRLERR                   = 00000054
SS$_DEVCMDERR                 = 0000032C
SS$_DEVREQERR                 = 00000334
SS$_NONEXPR                   = 000008E8
SS$_NORMAL                    = 00000001
SS$_NOSUCHSEC                 = 00000978
SS$_NOTRAN                    = 00000629
SS$_SSFAIL                    = 0000045C
SS$_WASSET                    = 00000009
SSERROR                         00000EF6 R      06
SS_SYNCH_EFN                  = 00000003
```

```
STATUS                  000001EE R      03
STR$UPCASE              ******** X      06
STS$K_ERROR           = 00000002
STS$K_INFO            = 00000003
STS$K_SEVERE          = 00000004
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$V_SEVERITY        = 00000000
SUC_EXIT                000009F9 R      06
SUPDEV_GBLSEC           00000020 R      02
SUP_FAB                 00002474 R      03
SWEEP                 = 00000001
SWEEP_ARGL_1            000008B8 R      02
SWEEP_ARGL_2            000008E4 R      02
SWEEP_ARGL_3            00000910 R      02
SWEEP_ARGL_4            0000093C R      02
SWEEP_ARGL_5            00000968 R      02
SWEEP_ARGL_6            00000994 R      02
SWEEP_ARGL_7            000009C0 R      02
SWEEP_CHECK             00000DE1 R      06
SWEEP_COMMON            0000CC1D R      06
SWEEP_COUNT             000022B8 R      03
SWEEP_ERR_MSG           000003D8 R      02
SWEEP_SECTION           00000B5E R      06
SYS$ASSIGN              ********  GX    06
SYS$CONNECT             ********  GX    06
SYS$CRMPSC              ********  GX    06
SYS$DCLEXH              ********  GX    06
SYS$EXIT                ********  GX    06
SYS$EXPREG              ********  GX    06
SYS$FAO                 ********   X    06
SYS$GET                 ********  GX    06
SYS$GETDEV              ********  GX    06
SYS$GETDVI              ********  GX    06
SYS$GETJPI              ********  GX    06
SYS$GETJPIW             ********  GX    06
SYS$GETMSG              ********  GX    06
SYS$HIBER               ********  GX    06
SYS$INPUT               00000061 R      02
SYS$MGBLSC              ********  GX    06
SYS$OPEN                ********  GX    06
SYS$PUTMSG              ********  GX    06
SYS$QIOW                ********  GX    06
SYS$SCHDWK              ********  GX    06
SYS$SETAST              ********  GX    06
SYS$SETIMR              ********  GX    06
SYS$SETPRI              ********  GX    06
SYS$SETPRN              ********  GX    06
SYS$SETPRV              ********  GX    06
SYS$SETSFM              ********  GX    06
SYS$TRNLOG              ********  GX    06
SYS$UPDATE              ********  GX    06
SYSIN_FAB               00002344 R      03
SYSIN_RAB                 00002394 R      03
SYSWP_ARGL_1              000009EC R      02
SYSWP_ARGL_2              00000A18 R      02
S_MSG                     000007DB R      02
TEST_NAME                 0000000F R      02
TEST_OVERM              = 00000002
TEST_OVERV              = 00000001
TEXT_BUFFER             = 0000012C
THREEMIN                  000001DD R      02
TIME_IT                   00000529 R      06
TIME_OUT                  00000EEE R      06
TRANSFER_TABLE            000022D0 R      03
TTCHAN                    00000000 R      03
UETLPAK00                 00000000 RG     06
UETP                    = 00740000
UETP$_ABENDD            = 007410F0
UETP$_ABORTC            = 0074832B
UETP$_BEGIND            = 00741038
UETP$_DENOSU            = 00748333
UETP$_ENDEDD            = 00741080
UETP$_ERBOXPROC         = 00748020
UETP$_FACILITY          = 00000074
UETP$_OPENIN            = 00741098
UETP$_TEXT              = 00741130
UETUNT$B_FLAGS          = 0000000B
UETUNT$B_TYPE           = 00000008
UETUNT$C_INDSIZ         = 000001A4
UETUNT$M_TESTABLE       = 00000002
UETUNT$T_FILSPC         = 00000014
UETUNT$V_TESTABLE       = 00000001
UETUNT$W_SIZE           = 00000009
UNIT_LIST                 00000238 R      03
UNIT_LOOP                 00001171 R      06
UPDATE_FAILED             000011D6 R      06
USER_STATUS               000022F8 R      03
WRITE_SIZE              = 00000800
XRATE_CHECK               00000D5E R      06
```

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	00000BEC (3052.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
RWDATA	00002404 (9412.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
BUFFERS	0000E594 (58772.)	04 (4.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
$RMSNAM	00000023 (35.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
LPA11K	0000125F (4703.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.07	00:00:00.58
Command processing	130	00:00:00.63	00:00:06.04
Pass 1	656	00:00:29.50	00:01:02.00
Symbol table sort	0	00:00:02.70	00:00:05.17
Pass 2	538	00:00:08.42	00:00:20.10
Symbol table output	2	00:00:00.34	00:00:00.78
Psect synopsis output	1	00:00:00.04	00:00:00.08
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1366	00:00:41.71	00:01:34.76

The working set limit was 1950 pages.
169310 bytes (331 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1716 non-local and 102 local symbols.
2491 source lines were read in Pass 1, producing 52 object records in Pass 2.
67 pages of virtual memory were used to define 59 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[UETP.OBJ]UETP.MLB;1	2
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	51
TOTALS (all libraries)	53

1932 GETS were required to define 53 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:UETLPAK00/OBJ=OBJ$:UETLPAK00 MSRC$:UETLPAK00/UPDATE=(ENH$:UETLPAK00)+EXECML$/LIB+LIB$:UETP/LIB
```

UETFORT03
LIS

UETFORT01
LIS

UETLPAK00
LIS

UETNETS00
LIS

UETDRIW00
LIS

UETFORT02
LIS