```
UUU          UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSS  YYY          YYY
UUU          UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSS  YYY          YYY
UUU          UUU  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTT  PPPPPPPPPPPP        SSSSSSSSSSS  YYY          YYY
UUU          UUU  EEE                    TTT        PPP          PPP  SSS              YYY          YYY
UUU          UUU  EEE                    TTT        PPP          PPP  SSS              YYY          YYY
UUU          UUU  EEE                    TTT        PPP          PPP  SSS              YYY          YYY
UUU          UUU  EEE                    TTT        PPP          PPP  SSS                 YYY    YYY
UUU          UUU  EEE                    TTT        PPP          PPP  SSS                 YYY    YYY
UUU          UUU  EEE                    TTT        PPP          PPP  SSS                 YYY    YYY
UUU          UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP       SSSSSSSSS              YYY
UUU          UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP       SSSSSSSSS              YYY
UUU          UUU  EEEEEEEEEEEE           TTT        PPPPPPPPPPPP       SSSSSSSSS              YYY
UUU          UUU  EEE                    TTT        PPP                        SSS              YYY
UUU          UUU  EEE                    TTT        PPP                        SSS              YYY
UUU          UUU  EEE                    TTT        PPP                        SSS              YYY
UUU          UUU  EEE                    TTT        PPP                        SSS              YYY
UUU          UUU  EEE                    TTT        PPP                        SSS              YYY
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP                SSSSSSSSSSS              YYY
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP                SSSSSSSSSSS              YYY
UUUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP                SSSSSSSSSSS              YYY
```

```
UU      UU  EEEEEEEEEE  TTTTTTTTTT    IIIIII    NN      NN    IIIIII    TTTTTTTTTT    000000       11
UU      UU  EEEEEEEEEE  TTTTTTTTTT    IIIIII    NN      NN    IIIIII    TTTTTTTTTT    000000       11
UU      UU  EE              TT          II      NN      NN      II          TT        00      00   1111
UU      UU  EE              TT          II      NN      NN      II          TT        00      00   1111
UU      UU  EE              TT          II      NNNN    NN      II          TT        00    0000     11
UU      UU  EE              TT          II      NNNN    NN      II          TT        00    0000     11
UU      UU  EEEEEEEE        TT          II      NN  NN  NN      II          TT        00  00  00     11
UU      UU  EEEEEEEE        TT          II      NN  NN  NN      II          TT        00  00  00     11
UU      UU  EE              TT          II      NN    NNNN      II          TT        0000    00     11
UU      UU  EE              TT          II      NN    NNNN      II          TT        0000    00     11
UU      UU  EE              TT          II      NN      NN      II          TT        00      00     11          ....
UU      UU  EE              TT          II      NN      NN      II          TT        00      00     11          ....
UUUUUUUUUU  EEEEEEEEEE      TT        IIIIII    NN      NN    IIIIII        TT          000000     111111        ....
UUUUUUUUUU  EEEEEEEEEE      TT        IIIIII    NN      NN    IIIIII        TT          000000     111111        ....

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
```

```
0000     1          .TITLE UETINIT01 VAX/VMS UETP SYSTEM CONFIGURATION SIZER
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 creates a file called UETINIDEV.DAT containing all device
0000    36 ;       names and unit numbers of devices that are supported by the UETP.
0000    37 ;       Various lists internal to VMS are searched to collect this information.
0000    38 ;       As this file is being created, the device, if supported, has a quick
0000    39 ;       test done on it to make sure that it is basically functional.  If
0000    40 ;       the device is not functional it is deselected from further UETP
0000    41 ;       testing by modifying UETINIDEV.DAT.
0000    42 ;
0000    43 ; ENVIRONMENT:
0000    44 ;       This program will run primarily in user access mode, with ASTs enabled
0000    45 ;       except during error processing.  The UETP$CLSIODB subroutine runs in
0000    46 ;       kernel mode.  We require the following privileges and quotas:
0000    47 ;                       CMKRNL
0000    48 ;                       GRPNAM
0000    49 ;                       TMPMBX
0000    50 ;
0000    51 ;--
0000    52 ;
0000    53 ; AUTHOR: Larry D. Jones,       CREATION DATE: November, 1980
0000    54 ;
0000    55 ; MODIFIED BY:
0000    56 ;
0000    57 ;       V03-008 RNH0009         Richard N. Holstein,    06-Jul-1984
```

```
0000     58 ;               Add NOA as known, but not testable.
0000     59 ;
0000     60 ;       V03-007 RNH0008         Richard N. Holstein,    15-Feb-1984
0000     61 ;               Take advantage of the new UETP message codes.  Fix SSERROR
0000     62 ;               interaction with RMS_ERROR.  Fix algorithm which checks for
0000     63 ;               known, but not testable devices.
0000     64 ;
0000     65 ;       V03-006 RNH0007         Richard N. Holstein,    10-Jul-1983
0000     66 ;               Add NDA as known, but not testable.
0000     67 ;
0000     68 ;       V03-005 RNH0006         Richard N. Holstein,    06-Mar-1983
0000     69 ;               Be even more cautious if a process hangs - time out $DELPRC.
0000     70 ;
0000     71 ;       V03-004 RNH0005         Richard N. Holstein,    14-Jan-1983
0000     72 ;               Define SYS$ERROR as NL: for the device test subprocesses.
0000     73 ;
0000     74 ;       V03-003 RNH0004         Richard N. Holstein,    07-Jan-1983
0000     75 ;               Convert to use UETP$CLSIODB routine to search VMS's I/O
0000     76 ;               database.  Redesign logic which deals with known, but
0000     77 ;               untestable devices.  Fix SSERROR to be reentrent.  Have
0000     78 ;               AST routines specifically set up SSERROR as their error
0000     79 ;               handler as necessary.  Clean out the remains of pre-V2.
0000     80 ;               Miscellaneous fixes listed in the V3B UETP Workplan.
0000     81 ;               Indent lines copied from subprocess device test log files.
0000     82 ;
0000     83 ;       V03-002 RNH0003         Richard N. Holstein,    22-Jun-1982
0000     84 ;               Fixed problem of ACCVIO in kernal mode if a controller has no
0000     85 ;               units connected to it.  Related problems of bogus line in
0000     86 ;               UETINIDEV.DAT and wrong unit number in error messages also
0000     87 ;               fixed.
0000     88 ;
0000     89 ;       V03-001 LDJ0003         Larry D. Jones,         15-Apr-1982
0000     90 ;               Fixed MA780 extra UCB record in the inidev file bug.
0000     91 ;
0000     92 ;       V02-007 RNH0002         Richard N. Holstein,    18-Jan-9182
0000     93 ;               Increase TEXT_BUFFER size to hold larger messages.  Fix
0000     94 ;               bug where BBS tested mask instead of bit position.
0000     95 ;
0000     96 ;       V02-006 RNP0003         Robert N. Perron,       31-Dec-1981
0000     97 ;               Removed upper casing of begin sentinels
0000     98 ;
0000     99 ;       V02-005 RNP0002         Robert N. Perron,       23-Dec-1981
0000    100 ;               Fixed so that non-supported devices do not appear in
0000    101 ;               UETINIDEV.DAT. Added device summary printout to SUC_EXIT and
0000    102 ;               removed printing of DDB and UCB lines. Fixed handling of
0000    103 ;               remote terminals - skip them.
0000    104 ;
0000    105 ;       V02-004 RNP0001         Robert N. Perron,       05-Nov-1981
0000    106 ;               Modified to allow batch execution.
0000    107 ;
0000    108 ;       V02-003 LDJ0002         Larry D. Jones,         14-Oct-1981
0000    109 ;               Reset DNTM before creating each new subprocess.
0000    110 ;
0000    111 ;       V02-002 RNH0001         Richard N. Holstein,    05-Oct-1981
0000    112 ;               Change SS$_CONTROLC to be a warning.  Use secondary device
0000    113 ;               characteristics for DDB devices as appropriate.  Remove
0000    114 ;               ALWAYS flag - it prevents useful info from being typed and
```

```
0000   115 ;                is redundantly implemented by judicious use of SHRT_RPRT.
0000   116 ;
0000   117 ;       V02-001 LDJ0001         Larry D. Jones,         08-Sep-1981
0000   118 ;               Fixed reporting of NET as a non-supported device.
0000   119 ;
0000   120 ;**
```

```
                     0000   122          .SBTTL  Declarations
                     0000   123 ;
                     0000   124 ; INCLUDE FILES:
                     0000   125 ;
                     0000   126
                     0000   127 ;
                     0000   128 ; MACROS:
                     0000   129 ;
                     0000   130          $ACCDEF                         ; Accounting definitions
                     0000   131          $CHFDEF                         ; Condition handler frame definitions
                     0000   132          $DCDEF                          ; Device classes and types
                     0000   133          $DIBDEF                         ; Device Information Block
                     0000   134          $DVIDEF                         ; $GETDVI ITMLST item codes
                     0000   135          $IODEF                          ; $QIO function codes
                     0000   136          $JPIDEF                         ; JPI definitions
                     0000   137          $PQLDEF                         ; Process quota list definitions
                     0000   138          $SECDEF                         ; Section definitions
                     0000   139          $SHRDEF                         ; Shared messages
                     0000   140          $SSDEF                          ; System service status codes
                     0000   141          $STSDEF                         ; Status return
                     0000   142          $UETIDBDEF                      ; UETP I/O database definitions
                     0000   143          $UETPDEF                        ; UETP
                     0000   144 ;
                     0000   145 ; EQUATED SYMBOLS:
                     0000   146 ;
                     0000   147 ;    Facility number definitions:
00000001             0000   148          RMS$_FACILITY = 1
                     0000   149
                     0000   150 ;    SHR message definitions:
00740000             0000   151          UETP = UETP$_FACILITY@STS$V_FAC_NO ; Make a mask of UETP facility code
007410E0             0000   152          UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
00741038             0000   153          UETP$_BEGIND = UETP!SHR$_BEGIND
00741080             0000   154          UETP$_ENDEDD = UETP!SHR$_ENDEDD
00741098             0000   155          UETP$_OPENIN = UETP!SHR$_OPENIN
00741130             0000   156          UETP$_TEXT   = UETP!SHR$_TEXT
                     0000   157
                     0000   158 ;    Internal flag bits...:
00000000             0000   159          SHRT_RPRTV   = 0                 ; Set if short report format desired
00000001             0000   160          BEGIN_MSGV   = 1                 ; Set when "begin" msg has been output
                     0000   161 ;    ...and corresponding masks:
00000001             0000   162          SHRT_RPRTM   = 1@SHRT_RPRTV
00000002             0000   163          BEGIN_MSGM   = 1@BEGIN_MSGV
                     0000   164
                     0000   165 ;    Miscellany:
00000020             0000   166          LC_BITM      = ^X20             ; Mask to convert lower case to upper
00000028             0000   167          REC_SIZE     = 40               ; UETINIDEV.DAT maximum record size
000000FA             0000   168          TEXT_BUFFER  = 250              ; Internal text buffer size
00000003             0000   169          SS_SYNCH_EFN = 3                ; Synch miscellaneous system services
0000000A             0000   170          MAX_DEV_DESIG = 10              ; Longest possible controller name
00000005             0000   171          MAX_UNIT_DESIG= 5               ; Longest possible unit number
00000100             0000   172          MBX_SIZE     = 256              ; Mailbox buffer size
00000050             0000   173          MAX_SUMM_LINE = 80              ; Longest summary line we'll create
00000004             0000   174          INDENT       = 4                ; Indentation when copying log files
```

```
                                0000   176          .SBTTL  Read-Only Data
                            00000000   177          .PSECT  RODATA,NOEXE,NOWRT,PAGE
                                0000   178
                                0000   179 ACNT_NAME:                              ; Process name on exit
53 45 54 53 59 53 00000008'010E0000'   180          .ASCID  /SYSTEST/
                                  54   000E
                                000F   181
                                000F   182 TEST_NAME:                              ; This test name
49 4E 49 54 45 55 00000017'010E0000'   183          .ASCID  /UETINIT01/
                              31 30 54   001D
                                0020   184
                                0020   185 SUPDEV_GBLSEC:                         ; How we access UETSUPDEV.DAT
50 55 53 54 45 55 00000028'010E0000'   186          .ASCID  /UETSUPDEV/
                              56 45 44   002E
                                0031   187
                                0031   188 MODE:                                  ; Run mode logical name
      45 44 4F 4D 00000039'010E0000'   189          .ASCID  /MODE/
                                003D   190
                                003D   191 REPORT_NAME:                           ; Long or short report indicator
54 52 4F 50 45 52 00000045'010E0000'   192          .ASCID  /REPORT/             ; See note where $TRNLOG is done
                                004B   193
                                004B   194 EQUA1:
         45 4E 4F 00000053'010E0000'   195          .ASCID  /ONE/
                                0056   196
                                0056   197 SUBPROC_ERROR:                         ; SYS$ERROR for the device tests
         3A 4C 4E 0000005E'010E0000'   198          .ASCID  /NL:/
                                0061   199
                                0061   200 SYS$COMMAND:                          ; Name of device from which we...
4F 43 24 53 59 53 00000069'010E0000'   201          .ASCID  /SYS$COMMAND/        ; ...get any input
                        44 4E 41 4D 4D   006F
                                0074   202
                                0074   203 NO_RMS_AST_TABLE:                     ; List of errors for which...
                    00000000'   0074   204          .LONG   RMS$_BLN             ; ...RMS cannot deliver an AST...
                    00000000'   0078   205          .LONG   RMS$_BUSY            ; ...even if one has an ERR= arg
                    00000000'   007C   206          .LONG   RMS$_CDA             ; Note that we can search table...
                    00000000'   0080   207          .LONG   RMS$_FAB             ; ...via MATCHC since <31:16>...
                    00000000'   0084   208          .LONG   RMS$_RAB             ; ...pattern can't be in <15:0>
                    00000014    0088   209 NRAT_LENGTH = .-NO_RMS_AST_TABLE
                                0088   210
                                0088   211 COMMAND_ITMLST:                       ; $GETDVI arg list for SYS$COMMAND
                  0004 0004     0088   212          .WORD   4,DVI$_DEVCLASS      ; We need the device class...
            00000000 000002DF'  008C   213          .LONG   DEVBUF,0
                  0020 0040     0094   214          .WORD   64,DVI$_DEVNAM       ; ...and the equivalence name
          0000003B'00000043'    0098   215          .LONG   BUFFER,BUFFER_PTR
                    00000000    00A0   216          .LONG   0                    ; Terminate the list
                                00A4   217
                                00A4   218 CLSIODB_ARGLST:                       ; What we send to UETP$CLSIODB
                    00000004    00A4   219          .LONG   4
00000499'00000491'000004A1'     00A8   220          .ADDRESS CLSPTR,LCLPTR,MPMPTR
                    0000001D    00B4   221          .LONG   UIDFLAG$M_DDB!UIDFLAG$M_UCB!UIDFLAG$M_MPM!UIDFLAG$M_SID
                                00B8   222
                                00B8   223 ; There is some trickiness in using the following.  We have a table of short
                                00B8   224 ; ASCIC strings to compare against a short ASCIC device name.  If we just used
                                00B8   225 ; a MATCHC of ASCII strings on ASCII strings, one string could "slur" into the
                                00B8   226 ; next, resulting in a bogus match.  With ASCIC strings though, the length
                                00B8   227 ; byte (assuming "short" strings of all printable characters!) serves as a
                                00B8   228 ; marker to prevent string overlap and allows the MATCHC to work as we wish.
```

```
                        00B8   229 KNOWN_BUT_NOT_TESTABLE:                ; Start of ASCIC string table
41 50 4F 00'            00B8   230          .ASCIC  /OPA/                 ; Operator console name
                  03    00B8
41 53 43 00'            00BC   231          .ASCIC  /CSA/                 ; Diagnostic load device name
                  03    00BC
41 4C 4E 00'            00C0   232          .ASCIC  /NLA/                 ; Null device
                  03    00C0
41 52 43 00'            00C4   233          .ASCIC  /CRA/                 ; First card reader
                  03    00C4
42 52 43 00'            00C8   234          .ASCIC  /CRB/                 ; Second card reader
                  03    00C8
41 54 52 00'            00CC   235          .ASCIC  /RTA/                 ; Remote terminals
                  03    00CC
54 45 4E 00'            00D0   236          .ASCIC  /NET/                 ; Networks
                  03    00D0
41 42 4D 00'            00D4   237          .ASCIC  /MBA/                 ; Mailboxes (local)
                  03    00D4
42 42 4D 00'            00D8   238          .ASCIC  /MBB/                 ; Mailboxes (shared memory)
                  03    00D8
41 44 4E 00'            00DC   239          .ASCIC  /NDA/                 ; Network dummy
                  03    00DC
41 4F 4E 00'            00E0   240          .ASCIC  /NOA/                 ; Asynchronous DECnet
                  03    00E0
20 54 50 00'            00E4   241          .ASCIC  /PT /                 ; Generic UDA for magtapes
                  03    00E4
20 55 50 00'            00E8   242          .ASCIC  /PU /                 ; Generic UDA for disks
                  03    00E8
20 41 50 00'            00EC   243          .ASCIC  /PA /                 ; Generic CI as a cluster connect
                  03    00EC
20 4E 43 00'            00F0   244          .ASCIC  /CN /                 ; DECnet over a CI
                  03    00F0
20 4A 43 00'            00F4   245          .ASCIC  /CJ /                 ; Generic common journalling
                  03    00F4
                00000040  00F8   246 KBNT_LENGTH = .-KNOWN_BUT_NOT_TESTABLE
                00000100  00F8   247          .BLKB   8                     ; Space for patching other devices
                          0100   248
                          0100   249 CS1:                                  ; Device class and type control string
21 20 42 58 32 21 00000108'010E0000'  0100   250          .ASCID  /!2XB !2XB /
                  20 42 58 32   010E
                          0112   251
                          0112   252 CS3:                                  ; Device class-only control string
2A 20 42 58 32 21 0000011A'010E0000'  0112   253          .ASCID  /!2XB ** /
                        20 2A   0120
                          0122   254
                          0122   255 MPM_CS:                               ; Faked $FAO result...
20 2A 2A 20 20 20 0000012A'010E0000'  0122   256          .ASCID  /   ** /                 ; ...from MPM "characteristics"
                          0130   257
                          0130   258 CONTOO_OVERHEAD:                     ; Table of overhead lines...
                0000013C'         0130   259          .ADDRESS PHASE_NAME               ; ...that begin UETCONTOO.DAT
                0000014D'         0134   260          .ADDRESS LOG_NAME
                00000162'         0138   261          .ADDRESS HEADER_REC
                00000003  013C   262 OVERHEAD_LENGTH = . - CONTOO_OVERHEAD/4
                          013C   263 PHASE_NAME:                            ; Phase name for the device test phase
50 54 45 55 20 3D 20 45 4D 41 4E 00'  013C   264          .ASCIC  /NAME = UETPDEVOO/
                  30 30 56 45 44  0148
                              10  013C
                          014D   265 LOG_NAME:                             ; UETCONTOO file log name
```

```
50 54 45 55 20 3D 20 20 47 4F 4C 00' 014D    266          .ASCIC  /LOG  = UETPDEV00.LOG/
          47 4F 4C 2E 30 30 56 45 44  0159
                                   14  014D
                                       0162    267 HEADER_REC:                                  ; UETCONT00.DAT init records
69 76 65 44 20 50 54 45 55 20 21 00' 0162    268          .ASCIC  /! UETP Device Test Phase/
73 61 68 50 20 74 73 65 54 20 65 63  016E
                                   65  017A
                                   18  0162
                                       017B    269
                                       017B    270 THREEMIN:                                    ; Time out value when creating...
                  FFFFFFFF 94B62E00    017B    271          .LONG   -10*1000*1000*180,-1        ; ...device test subprocesses
                                       0183    272
                                       0183    273 ONEMIN:                                      ; Time out when $DELPRCing...
                  FFFFFFFF DC3CBA00    0183    274          .LONG   -10*1000*1000*60,-1         ; ...device test subprocesses
                                       018B    275
                                       018B    276 BLANK_LINE_PTR:                              ; $PUTMSG MSGVEC for writing blank line
                         0001 0003     018B    277          .WORD   3,1
                          00741131     018F    278          .LONG   UETP$_TEXT!STS$K_SUCCESS
                          00000001     0193    279          .LONG   1
                          000001AB'    0197    280          .ADDRESS BLANK_LINE
                                       019B    281
                                       019B    282 LOG_BEGIN:                                   ; $PUTMSG MSGVEC for copying log file
                         000F 0003     019B    283          .WORD   3,^XF
                          007480B1     019F    284          .LONG   UETP$_COPY_LOG
                          00000001     01A3    285          .LONG   1
                          000004A9'    01A7    286          .ADDRESS TESTING_MSG
                                       01AB    287
                                       01AB    288 BLANK_LINE:
                  000001B3'010E0000'   01AB    289          .ASCID  //                          ; This line intentionally left blank
                                       01B3    290
                                       01B3    291 LOGEXT:                                      ; Log file type
                               47 4F 4C 2E     01B3    292          .ASCII  /.LOG/
                                       01B7    293
                                       01B7    294 CONT_STR:                                    ; Control string for UETCONT00 records
41 21 20 59 20 59 000001BF'010E0000'   01B7    295          .ASCID  /Y Y !AS ''!AS''/
                     22 53 41 21 22 20 53  01C5
                                       01CC    296
                                       01CC    297 TEST_COUNT:                                  ; Testable count logical name...
43 5F 54 53 45 54 000001D4'010E0000'   01CC    298          .ASCID  /TEST_COUNT/                ; ...for UETCONT00 file
                               54 4E 55 4F     01DA
                                       01DE    299
                                       01DE    300 TST_CNT_STR:                                 ; Control string for UETCONT00 records
          4C 55 21 000001E6'010E0000'  01DE    301          .ASCID  /!UL/
                                       01E9    302
                                       01E9    303 DDB_CTRSTR:                                  ; UETINIDEV.DAT...
20 54 20 42 44 44 000001F1'010E0000'   01E9    304          .ASCID  /DDB T !AC/                 ; ...DDB record control string
                               43 41 21  01F7
                                       01FA    305
                                       01FA    306 UCB_CTRSTR:                                  ; UETINIDEV.DAT...
20 54 20 42 43 55 00000202'010E0000'   01FA    307          .ASCID  /UCB T !5ZW/                ; ...UCB record control string
                               57 5A 35 21     0208
                                       020C    308
                                       020C    309 END_MSG:                                     ; UETINIDEV.DAT...
4E 49 54 45 55 20 46 4F 20 44 4E 45    020C    310          .ASCII  /END OF UETINIDEV.DAT/      ; ...ending message
          54 41 44 2E 56 45 44 49      0218
                                   00000014    0220    311          END_MSGL = .-END_MSG
                                       0220    312
```

```
                     0220   313 JPI_LIST:
          0304 0004  0220   314         .WORD   4,JPI$_UIC              ; List for the $GETJPI service
00000000'00000000'   0224   315         .ADDRESS UIC,0
          0409 0004  022C   316         .WORD   4,JPI$_ASTLM
00000000'00000005'   0230   317         .ADDRESS ASTLM,0
          0310 0004  0238   318         .WORD   4,JPI$_BIOLM
00000000'0000000A'   023C   319         .ADDRESS BIOLM,0
          0313 0004  0244   320         .WORD   4,JPI$_DIOLM
00000000'0000000F'   0248   321         .ADDRESS DIOLM,0
          0410 0004  0250   322         .WORD   4,JPI$_TQLM
00000000'00000014'   0254   323         .ADDRESS TQLM,0
            00000000 025C   324         .LONG   0
                     0260   325
                     0260   326 CONT_DESC:                              ; Desc used to convert controller...
            0003 0000  0260   327         .WORD   0,3                     ; ...from lowercase to uppercase
             00000049' 0264   328         .ADDRESS BUFFER+6
                     0268   329
                     0268   330 MPM_LITERAL:                           ; Shared memory must be called 'MPM'
          4D 50 4D 00' 0268   331         .ASCIC  /MPM/
                  03 0268
                     026C   332
                     026C   333 TESTABLE:                              ; Message summarizing testable units
                     026C   334         .REPT   MAX_DEV_DESIG+1         ; Leave space for controller (which...
                     026C   335         .ASCII  / /                     ; ...is overwritten) and colon
                  20 026C   336         .ENDR
20 20 20 65 6C 62 61 74 73 65 74 20  0277   337         .ASCII  / testable     /        ; Corresponds to UNTESTABLE
                  20 20 0283
            00000019 0285   338 TESTABLE_LEN = .-TESTABLE
                     0285   339
                     0285   340 UNTESTABLE:                            ; Message summarizing untestable units
                     0285   341         .REPT   MAX_DEV_DESIG+1         ; Corresponds to TESTABLE
                     0285   342         .ASCII  / /
                  20 0285   343         .ENDR
20 65 6C 62 61 74 73 65 74 6E 75 20  0290   344         .ASCII  / untestable   /        ; Also corresponds to TESTABLE
                  20 20 029C
                     029E   345 .IIF NE .-UNTESTABLE-TESTABLE_LEN, .ERROR ; TESTABLE & UNTESTABLE must be same lengt
                     029E   346
                     029E   347 NONE:                                  ; Message saying no testable units
                  65 6E 6F 6E 029E   348         .ASCII  /none/
                  00000004 02A2   349 NONE_LEN = .-NONE
                     02A2   350
                     02A2   351
                     02A2   352 SUMM_HEADER:
53 20 2A 2A 2A 20 000002AA'010E0000'  02A2   353         .ASCID  / *** Summary of testable and untestable devices. ***/
65 74 20 66 6F 20 79 72 61 6D 6D 75  02B0
75 20 64 6E 61 20 55 6C 62 61 74 73  02BC
65 64 20 65 6C 62 61 74 73 65 74 6E  02C8
      2A 2A 2A 20 2E 73 65 63 69 76  02D4
                     02DE   354
                     02DE   355 FORMAT_ERR_MSG:
20 72 6F 72 72 45 000002E6'010E0000'  02DE   356         .ASCID  /Error in UETINIDEV.DAT format./
56 45 44 49 4E 49 54 45 55 20 6E 69  02EC
2E 74 61 6D 72 6F 66 20 54 41 44 2E  02F8
                     0304   357
                     0304   358 START_DESC:                            ; Sentinel for...
   4E 49 47 45 42 0000030C'010E0000'  0304   359         .ASCID /BEGIN/                  ; ...start of useful log file info
                     0311   360
```

```
                                                0311   361 STOP_DESC:                              ; Sentinel for...
    44 45 44 4E 45 00000319'010E0000'           0311   362         .ASCID /ENDED/                  ; ...end of useful log file info
                                                031E   363
                                                031E   364 FNF_MSG:                               ; Message for missing subprocess log
69 66 20 67 6F 4C 00000326'010E0000'            031E   365         .ASCID  /Log file !AS was not found from testing controller !AS./
6E 20 73 61 77 20 53 41 21 20 65 6C             032C
6F 72 66 20 64 6E 75 6F 66 20 74 6F             0338
6F 63 20 67 6E 69 74 73 65 74 20 6D             0344
53 41 21 20 72 65 6C 6C 6F 72 74 6E             0350
                                  2E            035C
                                                035D   366
                                                035D   367 FLK_MSG:                               ; Proc hasn't terminated, can't get log
69 66 20 67 6F 4C 00000365'010E0000'            035D   368         .ASCID  /Log file !AS is locked from testing controller !AS./
6F 6C 20 73 69 20 53 41 21 20 65 6C             036B
65 74 20 6D 6F 72 66 20 64 65 6B 63             0377
6F 72 74 6E 6F 63 20 67 6E 69 74 73             0383
          2E 53 41 21 20 72 65 6C 6C            038F
                                                0398   369
                                                0398   370 CNTRLCMSG:
65 74 72 6F 62 41 000003A0'010E0000'            0398   371         .ASCID  \Aborted via a user CTRL/C\
72 65 73 75 20 61 20 61 69 76 20 64             03A6
                  43 2F 4C 52 54 43 20          03B2
                                                03B9   372
                                                03B9   373 FILE:                                ; Fills in RMS_ERR_STRING
       65 6C 69 66 000003C1'010E0000'           03B9   374         .ASCID  /file/
                                                03C5   375 RECORD:                              ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 000003CD'010E0000'            03C5   376         .ASCID  /record/
                                                03D3   377 RMS_ERR_STRING:                      ; Announces an RMS error
41 21 20 53 4D 52 000003DB'010E0000'            03D3   378         .ASCID  /RMS !AS error in file !AD/
66 20 6E 69 20 72 6F 72 72 65 20 53             03E1
                  44 41 21 20 65 6C 69          03ED
                                                03F4   379
                                                03F4   380 GETDVI_FAIL:
56 44 54 45 47 24 000003FC'010E0000'            03F4   381         .ASCID  /$GETDVI failed - status was:/
73 20 2D 20 64 65 6C 69 61 66 20 49             0402
       3A 73 61 77 20 73 75 74 61 74            040E
                                                0418   382
                                                0418   383 CLSIODB_FAIL:
65 6C 62 61 6E 55 00000420'010E0000'            0418   384         .ASCID  /Unable to read list of devices available to this system./
73 69 6C 20 64 61 65 72 20 6F 74 20             0426
73 65 63 69 76 65 64 20 66 6F 20 74             0432
74 20 65 6C 62 61 6C 69 61 76 61 20             043E
65 74 73 79 73 20 73 69 68 74 20 6F             044A
                                  2E 6D         0456
                                                0458   385
                                                0458   386 BAD_PO_LIST:
61 67 65 6C 6C 49 00000460'010E0000'            0458   387         .ASCID  /Illegal format in list of devices available to this system./
20 6E 69 20 74 61 6D 72 6F 66 20 6C             0466
69 76 65 64 20 66 6F 20 74 73 69 6C             0472
6C 62 61 6C 69 61 76 61 20 73 65 63             047E
79 73 20 73 69 68 74 20 6F 74 20 65             048A
                        2E 6D 65 74 73          0496
                                                049B   388
                                                049B   389 MBXW_QIO_FAIL:
66 20 4F 49 51 24 000004A3'010E0000'            049B   390         .ASCID  /$QIO failed when writing mailbox - status was:/
77 20 6E 65 68 77 20 64 65 6C 69 61             04A9
62 6C 69 61 6D 20 67 6E 69 74 69 72             04B5
```

```
20 73 75 74 61 74 73 20 2D 20 78 6F  04C1
                  3A 73 61 77  04CD
                               04D1    391
                               04D1    392 CREPRC_QIO_FAIL:
66 20 4F 49 51 24 000004D9'010E0000'  04D1    393         .ASCID  /$QIO failed when creating subprocess - status was:/
63 20 6E 65 68 77 20 64 65 6C 69 61  04DF
70 62 75 73 20 67 6E 69 74 61 65 72  04EB
61 74 73 20 2D 20 73 73 65 63 6F 72  04F7
            3A 73 61 77 20 73 75 74  0503
                               050B    394
                               050B    395 SUBPROCESS_FAIL:
20 72 6F 72 72 45 00000513'010E0000'  050B    396         .ASCID  /Error running !AS for controller !AS - final status was:/
20 53 41 21 20 67 6E 69 6E 6E 75 72  0519
6C 6C 6F 72 74 6E 6F 63 20 72 6F 66  0525
6E 69 66 20 2D 20 53 41 21 20 72 65  0531
61 77 20 73 75 74 61 74 73 20 6C 61  053D
                        3A 73        0549
                               054B    397
                               054B    398 PROCESS_STOP_MSG:                       ; Message if we must stop subprocess...
65 74 20 53 41 21 00000553'010E0000'  054B    399         .ASCID  \!AS testing controller !AS was stopped ($DELPRC) at !%T\-
6F 72 74 6E 6F 63 20 67 6E 69 74 73  0559
73 61 77 20 53 41 21 20 72 65 6C 6C  0565
44 24 28 20 64 65 70 70 6F 74 73 20  0571
25 21 20 74 61 20 29 43 52 50 4C 45  057D
                           54        0589
20 65 73 75 61 63 65 62 5F 21 2F 21  058A    400                 \!/!_because it seemed hung or because UETINIT01 was aborted.\
75 68 20 64 65 6D 65 65 73 20 74 69  0596
73 75 61 63 65 62 20 72 6F 20 67 6E  05A2
20 31 30 54 49 4E 49 54 45 55 20 65  05AE
2E 64 65 74 72 6F 62 61 20 73 61 77  05BA
                               05C6    401                                     ; ...before getting termination mailbox
                               05C6    402
                               05C6    403 PASS_OVER_MSG:                      ; Message if $DELPRC couldn't stop proc
65 74 20 53 41 21 000005CE'010E0000'  05C6    404         .ASCID  \!AS testing controller !AS did not stop despite $DELPRC.\-
6F 72 74 6E 6F 63 20 67 6E 69 74 73  05D4
64 69 64 20 53 41 21 20 72 65 6C 6C  05E0
65 64 20 70 6F 74 73 20 74 6F 6E 20  05EC
52 50 4C 45 44 24 20 65 74 69 70 73  05F8
                           2E 43        0604
65 67 6E 6F 6C 20 6F 4E 5F 21 2F 21  0606    405                 \!/!_No longer waiting (!%T) for the termination mailbox.\
21 28 20 67 6E 69 74 69 61 77 20 72  0612
20 65 68 74 20 72 6F 66 20 29 54 25  061E
20 6E 6F 69 74 61 6E 69 6D 72 65 74  062A
            2E 78 6F 62 6C 69 61 6D  0636
```

```
                        063E   407          .SBTTL  Read/Write Data
                    00000000   408          .PSECT  RWDATA,WRT,NOEXE,PAGE
                        0000   409
                        0000   410 UIC:                                    ; Our UIC
                    00000000   411          .LONG   0
                        0004   412 QUOTA_LIST:                             ; Quota list for created subprocess
                          01   413          .BYTE   PQL$_ASTLM
                        0005   414 ASTLM:
                    00000000   415          .LONG   0
                          02   416          .BYTE   PQL$_BIOLM
                        000A   417 BIOLM:
                    00000000   418          .LONG   0
                          05   419          .BYTE   PQL$_DIOLM
                        000F   420 DIOLM:
                    00000000   421          .LONG   0
                          09   422          .BYTE   PQL$_TQELM
                        0014   423 TQLM:
                    00000000   424          .LONG   0
                          00   425          .BYTE   PQL$_LISTEND
                        0019   426
                        0019   427 TTCHAN:                                  ; Channe  associated with ctrl. term.
                        0000   428          .WORD   0
                        001B   429
                        001B   430 MBXCHN:                                  ; Mailbox channels - termination mbx
                        0000   431          .WORD   0
                        001D   432 MBX_UNIT:                                ; Unit number for termination mailbox
                        0000   433          .WORD   0
                        001F   434
                        001F   435 MBX1_CHAN:                               ; SYS$INPUT of created subprocesses
                        0000   436          .WORD   0
                        0021   437
                        0021   438 FLAG:                                   ; Miscellaneous flag bits
                        0000   439          .WORD   0                      ; (See Equated Symbo's for definitions)
                        0023   440
                        0023   441 LOG_RCD:                                ; Message vector for $PUTMSG
                        0003   442          .WORD   3                      ; Arg count
                        0001   443          .WORD   ^B0001                 ; Message only flag
                    00741130   444          .LONG   UETP$_TEXT             ; Message ID
                        0001   445          .WORD   1                      ; FAO arg count
                        0000   446          .WORD   0                      ; No new msg flags
                        002F   447 MSG_PTR:
                0000003B'   002F   448          .LONG   BUFFER_PTR             ; Message buffer for $PUTMSG
                        0033   449
                        0033   450 FAO_BUF:                                ; FAO output string descriptor
                    0000 00FA  451          .WORD   TEXT_BUFFER,0
                00000043'   0037   452          .ADDRESS BUFFER
                        003B   453 BUFFER_PTR:                             ; Fake .ASCID buffer for misc. strings
                    0000 00FA  454          .WORD   TEXT_BUFFER,0          ; A word for length, a word for desc.
                00000043'   003F   455          .ADDRESS BUFFER
                        0043   456 BUFFER:                                 ; FAO output and other misc. buffer
                    0000013D   457          .BLKB   TEXT_BUFFER
                        013D   458
                        013D   459 FAO_ALT:                                ; FAO output string descriptor #2
                    00000050   460          .LONG   MAX_SUMM_LINE
                0000014D'   0141   461          .ADDRESS ALTBUF
                        0145   462 ALTBUF_PTR:                             ; Output string descriptor #2
                    00000000   463          .LONG   0
```

```
0000014D' 0149   464          .ADDRESS ALTBUF
          014D   465 ALTBUF:                                ; FAO output buffer #2
0000019D  014D   466          .BLKB    MAX_SUMM_LINE
          019D   467
          019D   468 DEVDSC:                                ; Device name descriptor
000001A5  019D   469          .BLKL    2                    ; This gets filled at runtime
          01A5   470
          01A5   471 MBX_BUF:
000002A5  01A5   472          .BLKB    MBX_SIZE             ; Mailbox read buffer
          02A5   473
          02A5   474 TEST_DSC:                              ; Device test file name descriptor...
0000000D  02A5   475          .LONG    13                   ; ...for the subprocess we create
000002AD' 02A9   476          .ADDRESS TEST_IMAGE
          02AD   477 TEST_IMAGE:                            ; Device test file name
000002BA  02AD   478          .BLKB    13
          02BA   479
          02BA   480 PIDADR:                                ; Receives PID of the subprocesses...
00000000  02BA   481          .LONG    0                    ; ...we create for each controller
          02BE   482
          02BE   483 LOGNAM_DSC:                            ; Descriptor for log file name
00000000  02BE   484          .LONG    0
000002C6' 02C2   485          .ADDRESS LOGNAM
          02C6   486 LOGNAM:
000002D7  02C6   487          .BLKB    17
          02D7   488
          02D7   489 DEV:                                   ; Device Information Block
00000074  02D7   490          .LONG    DIB$K_LENGTH
000002DF' 02DB   491          .ADDRESS DEVBUF
          02DF   492 DEVBUF:
00000353  02DF   493          .BLKB    DIB$K_LENGTH
          0353   494
          0353   495 ERROR_COUNT:                           ; Cumulative error count at runtime
          00000000  0353   496          .LONG    0
          0357   497
          0357   498 STATUS:                                ; Status value on program exit
          00000000  0357   499          .LONG    0
          035B   500
          035B   501 QUAD_STATUS:                           ; Status block for return from...
00000000 00000000 035B   502          .QUAD    0           ; ...miscellaneous asynch services
          0363   503
          0363   504 INADDRESS:                            ; $CRMPSC address storage
00000000 00000000 0363   505          .LONG    0,0
          036B   506 OUTADDRESS:
00000000 00000000 036B   507          .LONG    0,0
          0373   508
          0373   509 TST_CNT:                              ; Count of UETCONT00 data records
          00000000  0373   510          .LONG    0
          0377   511
          0377   512 MSG_BLOCK:                           ; Auxiliary $GETMSG info
          0000037B  0377   513          .BLKB    4
          037B   514
          037B   515 EXIT_DESC:                           ; Exit handler descriptor
          00000000  037B   516          .LONG    0
          00000F16' 037F   517          .ADDRESS EXIT_HANDLER
          00000001  0383   518          .LONG    1
          00000357' 0387   519          .ADDRESS STATUS
                    038B   520
```

```
                     038B   521 ARG_COUNT:                              ; Argument counter used by ERROR_EXIT
           00000000  038B   522         .LONG   0
                     038F   523
                     038F   524 TEMP_BUFF_DESC:
       0000 00FA     038F   525         .WORD TEXT_BUFFER,0             ; Temp buffer to hold capitalized...
        00000397'    0393   526         .ADDRESS TEMP_BUFF_STR          ; ...record to check for sentinel
                     0397   527 TEMP_BUFF_STR:
           00000491  0397   528         .BLKB TEXT_BUFFER
                     0491   529
                     0491   530 LCLPTR:                                 ; Receives results of UETP$CLSIODB...
   00000000 00000000 0491   531         .LONG   0,0                     ; ...call for local peripherals
                     0499   532
                     0499   533 MPMPTR:                                 ; Receives results of UETP$CLSIODB...
   00000000 00000000 0499   534         .LONG   0,0                     ; ...call for shared memories
                     04A1   535
                     04A1   536 CLSPTR:                                 ; Receives results of UETP$CLSIODB...
   00000000 00000000 04A1   537         .LONG   0,0                     ; ...call for cluster peripherals
                     04A9   538
                     04A9   539 TESTING_MSG:                            ; This becomes part of the message...
           0000 0000 04A9   540         .WORD   0,0                     ; ...surrounding a copy of a...
        000004B1'    04AD   541         .ADDRESS .+4                    ; ...log file from a subprocess
20 67 6E 69 74 73 65 74  04B1   542         .ASCII  /testing /
           00000008  04B9   543 TESTING_MSG_LENGTH = .-TESTING_MSG-8
                     04B9   544 TESTING_MSG_TEXT:
                     04B9   545         .REPEAT MAX_DEV_DESIG
                     04B9   546         .ASCII  / /                     ; Gets overwritten with controller name
                 20  04B9   547         .ENDR
```

```
                     549          .SBTTL  RMS-32 Data Structures
                     550          .ALIGN  LONG
                     551
                     552 INI_FAB:                                    ; Allocate FAB for UETINIDEV
                     553          $FAB-
                     554          FAC = <GET,PUT,UPD,TRN>,-
                     555          RAT = CR,-
                     556          SHR = <GET,PUT,UPI>,-
                     557          FNM = <UETINIDEV.DAT>
                     558
                     559 INI_RAB:                                    ; Allocate RAB for UETINIDEV
                     560          $RAB-
                     561          FAB = INI_FAB,-
                     562          UBF = BUFFER,-
                     563          RBF = BUFFER,-
                     564          USZ = REC_SIZE
                     565
                     566 SUP_FAB:                                    ; Allocate FAB for UETSUPDEV
                     567          $FAB-
                     568          FAC = GET,-
                     569          SHR = <UPI,GET>,-
                     570          FOP = UFO,-
                     571          FNM = <UETSUPDEV.DAT>
                     572
                     573 CON_FAB:                                    ; Allocate FAB for UETCONT00
                     574          $FAB-
                     575          FAC = <PUT>,-
                     576          RAT = CR,-
                     577          FNM = <UETCONT00.DAT>
                     578
                     579 CON_RAB:                                    ; Allocate RAB for UETCONT00
                     580          $RAB-
                     581          FAB = CON_FAB
                     582
                     583 LOG_FAB:                                    ; Allocate FAB for UETINIT01 log
                     584          $FAB    FNM = <UETINIT01.LOG>,-
                     585                  RAT = CR,-
                     586                  FAC = PUT
                     587 LOG_RAB:                                    ; Allocate RAB for UETINIT01 log
                     588          $RAB    FAB = LOG_FAB,-
                     589                  RBF = BUFFER,-
                     590                  RSZ = TEXT_BUFFER
                     591
                     592 TMPLOG_FAB:                                ; Allocate FAB for subprocess log
                     593          $FAB    RAT = CR,-
                     594                  FAC = GET
                     595
                     596 TMPLOG_RAB:                                ; Allocate RAB for subprocess log
                     597          $RAB    FAB = TMPLOG_FAB,-
                     598                  UBF = BUFFER,-
                     599                  USZ = TEXT_BUFFER
                     600 DDB_RFA:                                    ; Storage for current DDB RFA
0000076A             601          .BLKB   6
                     602 END_RFA:                                    ; Storage for END line RFA
00000770             603          .BLKB   6
```

```
                     0770   605          .SBTTL  Main Program
                00000000    606          .PSECT  UETINIT01,EXE,NOWRT,PAGE
                     0000   607
                     0000   608          .DEFAULT DISPLACEMENT,WORD
                     0000   609
                0000 0000   610 .ENTRY UETINIT01,^M<>                    ; Entry mask
                     0002   611 ;+
                     0002   612 ;       Start to set things up.  We need our own overhead tasks (exception,
                     0002   613 ;       CTRL/C and exit handlers, name and log file), the items for
                     0002   614 ;       what we produce (UETINIDEV.DAT, UETCONTOO.DAT) and the means of
                     0002   615 ;       producing them (UETSUPDEV.DAT, various logical names, communication
                     0002   616 ;       for subprocesses).
                     0002   617 ;-
                     0002   618
                     0002   619 ;
                     0002   620 ; Overhead tasks.
                     0002   621 ;
          6D 0C91'CF   DE   0002   622          MOVAL   SSERROR,(FP)            ; Declare exception handler
                     0007   623          $SETSFM_S ENBFLG = #1           ; Enable system service failure mode
                     0010   624          $DCLEXH_S DESBLK = EXIT_DESC    ; Declare an exit handler
                     001B   625
                     001B   626          $CREATE FAB = LOG_FAB,-         ; Create our own log file
                     001B   627                  ERR = RMS_ERROR
                     002A   628          $CONNECT RAB = LOG_RAB,-
                     002A   629                   ERR = RMS_ERROR
                   7E   D4   0039   630          CLRL    -(SP)                   ; Set the time stamp flag
              000F'CF   DF   003B   631          PUSHAL  TEST_NAME               ; Set the test name
                   02   DD   003F   632          PUSHL   #2                      ; Push the argument count
           00741039 8F   DD   0041   633          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
        00000000'GF   04   FB   0047   634          CALLS   #4,G^LIB$SIGNAL         ; Print the startup message
           0021'CF   02   A8   004E   635          BISW2   #BEGIN_MSGM,FLAG        ; Set flag so we don't type it twice
                     0053   636          $SETPRN_S PRCNAM = TEST_NAME    ; Set the process name
                     005E   637
                     005E   638          $GETDVI_S DEVNAM = SYS$COMMAND,-    ; Get the name of...
                     005E   639                    EFN = #SS_SYNCH_EFN,- ; ...device which may abort test
                     005E   640                 ITMLST = COMMAND_ITMLST,-
                     005E   641                   IOSB = QUAD_STATUS
             28 035B'CF   E8   007A   642          BLBS    QUAD_STATUS,20$         ; BR if all went OK
          52 035B'CF   3C   007F   643          MOVZWL  QUAD_STATUS,R2          ; Set up arglist so we can appease...
                0BE2   30   0084   644          BSBW    FAO_CHECK               ; ...LIB$SIGNAL's use of $PUTMSG
                  52   DD   0087   645          PUSHL   R2                      ; Save flags and FAO count for $PUTMSG
                  02   13   0089   646          BEQL    10$                     ; BR if there are no FAO args
                  01   DD   008B   647          PUSHL   #1                      ; There are args - save dummy for count
                     008D   648 10$:
          52   8E   04   C1   008D   649          ADDL3   #4,(SP)+,R2             ; Pop bogus arg and get arg count
          7E 035B'CF   3C   0091   650          MOVZWL  QUAD_STATUS,-(SP)       ; Save the error status
               03F4'CF   DF   0096   651          PUSHAL  GETDVI_FAIL             ; Explain what went wrong
                  01   DD   009A   652          PUSHL   #1
           00741132 8F   DD   009C   653          PUSHL   #UETP$_TEXT!STS$K_ERROR
                  52   DD   00A2   654          PUSHL   R2                      ; Arg count for ERROR_EXIT
                0DEF   31   00A4   655          BRW     ERROR_EXIT
                     00A7   656 20$:
    02DF'CF   42 8F   91   00A7   657          CMPB    #DC$_TERM,DEVBUF        ; Were we invoked from a terminal?
                  45   12   00AD   658          BNEQ    30$                     ; BR if not
                     00AF   659          $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C ASTs if we were
                     00AF   660                    CHAN = TTCHAN
                     00C0   661          $QIOW_S CHAN = TTCHAN,-         ; Enable CTRL/C ASTs...
```

```
                                00C0   662                 FUNC = #IO$_SETMODE!IO$M_CTRLCAST,-
                                00C0   663                   P1 = CCASTHAND
                 000F'CF    DF  00E1   664         PUSHAL  TEST_NAME               ; ...and tell the user...
                       01   DD  00E5   665         PUSHL   #1
             0074832B  8F   DD  00E7   666         PUSHL   #UETP$_ABORTC!STS$K_SUCCESS
         00000000'GF   03   FB  00ED   667         CALLS   #3,G^LIB$SIGNAL         ; ...how to abort gracefully
                                00F4   668 ;
                                00F4   669 ; What we produce.
                                00F4   670 ;
                                00F4   671 30$:
                                00F4   672         $CREATE FAB = INI_FAB,-         ; Create UETINIDEV.DAT
                                00F4   673                 ERR = RMS_ERROR
                                0103   674         $CONNECT RAB = INI_RAB,-
                                0103   675                  ERR = RMS_ERROR
            5B   05F8'CF    DE  0112   676         MOVAL   CON_RAB,R11             ; Set the RAB pointer for UETCONT00.DAT
                                0117   677         $CREATE FAB = CON_FAB,-         ; Create UETCONT00.DAT
                                0117   678                 ERR = RMS_ERROR
                                0126   679         $CONNECT RAB = (R11),-
                                0126   680                  ERR = RMS_ERROR
                          5A    D4  0133   681         CLRL    R10                     ; Initialize counter for overhead lines
                                0135   682 40$:                                     ; Write UETCONT00.DAT overhead lines
             59   0130'CF4A    D0  0135   683         MOVL    CONT00_OVERHEAD[R10],R9 ; Point to the next ASCIC overhead line
                22 AB    69   9B  0138   684         MOVZBW  (R9),RAB$W_RSZ(R11)     ; Set the record size
           28 AB    59    01   C1  013F   685         ADDL3   #1,R9,RAB$L_RBF(R11)    ; Set the record address
                                0144   686         $PUT    RAB = (R11),-           ; Write a line
                                0144   687                 ERR = RMS_ERROR
                EO 5A    03   F2  0151   688         AOBLSS  #OVERHEAD_LENGTH,R10,40$ ; Repeat until overhead lines written
                                0155   689 ;
                                0155   690 ; What we need to use in order to produce.
                                0155   691 ;
                                0155   692         $OPEN   FAB = SUP_FAB,-         ; Open UETSUPDEV.DAT
                                0155   693                 ERR = RMS_ERROR
                                0164   694         $CRMPSC_S INADR = INADDRESS,-   ; Create its global section
                                0164   695                   CHAN = SUP_FAB+FAB$L_STV,-
                                0164   696                 RETADR = OUTADDRESS,-
                                0164   697                 GSDNAM = SUPDEV_GBLSEC,-
                                0164   698                  FLAGS = #SEC$M_EXPREG!SEC$M_GBL
                                018C   699         $TRNLOG_S RSLBUF = FAO_BUF,-    ; Get long or short report format
                                018C   700                   LOGNAM = REPORT_NAME
                                01A3   701 ; We need not check SS$_NOTRAN; BUFFER will contain "REPORT" on errors.
          0043'CF    20   8A  01A3   702         BICB2   #LC_BITM,BUFFER         ; Convert to upper case
        0043'CF    53 8F   91  01A8   703         CMPB    #^A/S/,BUFFER           ; Do we want a short report?
                      05   12  01AE   704         BNEQ    50$                     ; BR if not
           0021'CF    01   A8  01B0   705         BISW2   #SHRT_RPRTM,FLAG        ; Else set the short report flag bit
                                01B5   706 50$:
                                01B5   707         $CRELOG_S TBLFLG = #1,-         ; Set group logical name MODE...
                                01B5   708                   LOGNAM = MODE,-       ; ...to ONE
                                01B5   709                   EQLNAM = EQUATE
                                01C8   710         $CREMBX_S CHAN = MBXCHN         ; Create subprocess termination mailbox
                                01DB   711         $GETCHN_S CHAN = MBXCHN,-       ; Get the mailbox unit number
                                01DB   712                  PRIBUF = DEV
     001D'CF    02EB'CF    B0  01F1   713         MOVW    DEVBUF+DIB$W_UNIT,MBX_UNIT ; Save the mailbox unit number
                                01F8   714         $GETJPI_S ITMLST = JPI_LIST     ; Get our current quotas, etc.
                                020D   715         $CREMBX_S CHAN   = MBXT_CHAN,-  ; Create SYS$INPUT for subprocesses
                                020D   716                   MAXMSG = #MAX_DEV_DESIG,-
                                020D   717                   LOGNAM = TEST_NAME
                                0224   718 ; Fall into loop to build UETINIDEV.DAT.
```

```
                                0224   720 ;+
                                0224   721 ;       Form a representation in user mode P0 space of the devices which are
                                0224   722 ;       visible to this system.  Cull from the list those devices which we
                                0224   723 ;       know of, but know that we have no specific test for.  Cull from the
                                0224   724 ;       remainder those devices we don't know at all (i.e., devices which
                                0224   725 ;       don't appear in UETSUPDEV.DAT).  Use what remains to write
                                0224   726 ;       UETINIDEV.DAT.  See CLSIODB.MAR for a description of the database
                                0224   727 ;       we read.  Use the list to see which devices are really testable.
                                0224   728 ;
                                0224   729 ;       Note that R10 and R11 remain assigned throughout the range of
                                0224   730 ;       BUILD_INIDEV.
                                0224   731 ;-
                                0224   732
                                0224   733           $CMKRNL_S ROUTIN = UETP$CLSIODB,- ; Form a list of devices we can see
                                0224   734                     ARGLST = CLSIODB_ARGLST
                      26 50   E8 0233   735           BLBS    R0,110$                   ; BR if the list was formed
                   53    50   D0 0236   736           MOVL    R0,R3                     ; Save status over routine call
                   52    53   D0 0239   737           MOVL    R3,R2                     ; Set up arglist so we can appease...
                      0A2A   30 023C   738           BSBW    FAO_CHECK                 ; ...LIB$SIGNAL's use of $PUTMSG
                         52   DD 023F   739           PUSHL   R2                        ; Save flags and FAO count for $PUTMSG
                         02   13 0241   740           BEQL    100$                      ; BR if there are no FAO args
                         01   DD 0243   741           PUSHL   #1                        ; There are args - save dummy for count
                                0245   742 100$:
                52    8E    04   C1 0245   743           ADDL3   #4,(SP)+,R2               ; Pop bogus arg and get arg count
                         53   DD 0249   744           PUSHL   R3                        ; Save the error status
                   0418'CF    DF 024B   745           PUSHAL  CLSIODB_FAIL              ; Explain what went wrong
                         01   DD 024F   746           PUSHL   #1
                00741132 8F   DD 0251   747           PUSHL   #UETP$_TEXT!STS$K_ERROR
                         52   DD 0257   748           PUSHL   R2                        ; Arg count for ERROR_EXIT
                      0C3A   31 0259   749           BRW     ERROR_EXIT
                                025C   750 110$:
                                025C   751
5A    036F'CF   036B'CF   C3 025C   752           SUBL3   OUTADDRESS,OUTADDRESS+4,R10 ; Figure UETSUPDEV section length
         5B   0491'CF   D0 0264   753           MOVL    LCLPTR,R11                ; Point to first local peripheral rec
                   11   10 0269   754           BSBB    BUILD_INIDEV              ; Build UETINIDEV.DAT for local p's
         5B   0499'CF   D0 026B   755           MOVL    MPMPTR,R11                ; Point to first shared memory record
                   0A   10 0270   756           BSBB    BUILD_INIDEV              ; Build UETINIDEV.DAT for MPMs
         5B   04A1'CF   D0 0272   757           MOVL    CLSPTR,R11                ; Point to first cluster periph. record
                   03   10 0277   758           BSBB    BUILD_INIDEV              ; Build UETINIDEV.DAT for clus. periph.
                      025D   31 0279   759           BRW     DO_SUMMARY                ; Finish off with summary, etc.
                                027C   760
                                027C   761 BUILD_INIDEV:
                   06 AB    8F 027C   762           CASEB   UIDGNRC$B_TYPE(R11),-     ; Dispatch based on record type...
                      00         027F   763                   #UID$K_NULL_RTYPE,-       ; ...in the list we just got
                      06         0280   764                   #UID$K_END_RTYPE
                         000E' 0281   765 10$:              .WORD   NULL_RECORD-10$
                         0020' 0283   766                   .WORD   SID_RECORD-10$
                         000E' 0285   767                   .WORD   PATH_RECORD-10$
                         0169' 0287   768                   .WORD   DDB_RECORD-10$
                         000E' 0289   769                   .WORD   UCB_RECORD-10$
                         0144' 028B   770                   .WORD   MPM_RECORD-10$
                         001F' 028D   771                   .WORD   END_RECORD-10$
                                028F   772
                                028F   773 ; Still in BUILD_INIDEV loop.
                                028F   774 ; Fall into the routine for illegal or out of sequence record types.
```

```
                     028F    776 ;
                     028F    777 ; Null records are supposedly used as bookeeping and error recovery devices
                     028F    778 ; by UETP$CLSIODB.  However, there really is no case when they should appear.
                     028F    779 ;
                     028F    780 NULL_RECORD:
                     028F    781 ;
                     028F    782 ; This can be reached two ways.  It is the default branch from the CASEB
                     028F    783 ; above, indicating an illegal record type was seen.  We also get here if
                     028F    784 ; a record is out of sequence, e.g., a UCB record precedes a DDB record.
                     028F    785 ;
                     028F    786 PATH_RECORD:
                     028F    787 UCB_RECORD:
          0458'CF    DF  028F    788         PUSHAL  BAD_PO_LIST             ; Explain what went wrong
                01    DD  0293    789         PUSHL   #1
00741132 8F         DD  0295    790         PUSHL   #UETP$_TEXT!STS$K_ERROR
                03    DD  029B    791         PUSHL   #3                      ; Arg count for ERROR_EXIT
              0BF6    31  029D    792         BRW     ERROR_EXIT
                          02A0    793
                          02A0    794
                          02A0    795
                          02A0    796 ;
                          02A0    797 ; End records mark the end of a group of records returned by UETP$CLSIODB.
                          02A0    798 ;
                          02A0    799 END_RECORD:
                     05   02A0    800         RSB
                          02A1    801
                          02A1    802 ; Still in BUILD_INIDEV loop.
```

```
                        02A1    804 ;
                        02A1    805 ; A system id record introduces a potential tree of DDB and UCB records for
                        02A1    806 ; devices on that SCS cluster node.  Items for UETINIDEV.DAT will look similar
                        02A1    807 ; to those for local devices, except that the DDB line will have the full SCS
                        02A1    808 ; node name.  Once set up, we can use the same routines as local DDBs.
                        02A1    809 ;
                        02A1    810 $ID_RECORD:
            32 AB   9F  02A1    811         PUSHAB  UIDSID$T_NODENAME+1(R11)     ; Form descriptor...
         7E 31 AB   9A  02A4    812         MOVZBL  UIDSID$T_NODENAME(R11),-(SP) ; ...to cluster node name
         50 04 AB   3C  02A8    813         MOVZWL  UIDGNRC$W_SIZE(R11),R0  ; Get the length of this record...
      5B    50      C0  02AC    814         ADDL2   R0,R11                  ; ...so we can point to the next
                        02AF    815 10$:
               03   91  02AF    816         CMPB    #UID$K_DDB_RTYPE,-      ; Is this a DDB record?
            06 AB       02B1    817                 UIDGNRC$B_TYPE(R11)
               04   13  02B3    818         BEQL    20$                     ; BR if so - we can process it
               03   BA  02B5    819         POPR    #^M<R0,R1>              ; Finished the tree - fix up stack...
               C3   11  02B7    820         BRB     BUILD_INIDEV            ; ...and look to the next tree
                        02B9    821 20$:
                        02B9    822
         57    0C AB   9E  02B9    823         MOVAB   UIDDDB$T_NAME+1(R11),R7 ; From the ASCIC, form a descriptor...
         56    FF A7   9A  02BD    824         MOVZBL  -1(R7),R6               ; ...to the controller name
                        02C1    825         ; See the note where KNOWN_BUT_NOT_TESTABLE table is defined.
   0040 8F    67   56   39  02C1    826         MATCHC  R6,(R7),#KBNT_LENGTH,-  ; See if specific device is known...
            00B8'CF         02C7    827                 KNOWN_BUT_NOT_TESTABLE  ; ...but not testable
               0F   13  02CA    828         BEQL    30$                     ; BR if we found it in table
      50    57   01   C3  02CC    829         SUBL3   #1,R7,R0                ; See if generic device is known but...
   0040 8F    60   56   39  02D0    830         MATCHC  R6,(R0),#KBNT_LENGTH,-  ; ...not testable. (We strip "C"...
            00B8'CF         02D6    831                 KNOWN_BUT_NOT_TESTABLE  ; ...of "devc" but include ASCIC count)
               05   12  02D9    832         BNEQ    40$                     ; BR if device may be testable
                        02DB    833 30$:
            00D9    30  02DB    834         BSBW    SKIP_CLUS_RECORDS       ; Device is not testable, skip it...
               CF   11  02DE    835         BRB     10$                     ; ...and look to the next
                        02E0    836 40$:
                        02E0    837
         54    6E   7D  02E0    838         MOVQ    (SP),R4                 ; Restore desc for cluster node name
   019D'CF    56   54   C1  02E3    839         ADDL3   R4,R6,DEVDSC            ; Figure the length of...
            019D'CF   D6  02E9    840         INCL    DEVDSC                  ; ...the complete cluster device name
      5E    019D'CF   C2  02ED    841         SUBL2   DEVDSC,SP               ; Save the space to form a string
   01A1'CF    5E   D0  02F2    842         MOVL    SP,DEVDSC+4             ; Finish ASCID pointer to device name
   6E    65   54   28  02F7    843         MOVC3   R4,(R5),(SP)            ; Concatenate cluster node name...
         83   24   90  02FB    844         MOVB    #^A/$/,(R3)+            ; ...with syntax character...
   63    67   56   28  02FE    845         MOVC3   R6,(R7),(R3)            ; ...and with device name
   7E    019D'CF   90  0302    846         MOVB    DEVDSC,-(SP)            ; Set up ASCIC device name on stack
                        0307    847
      50    04 AB   3C  0307    848         MOVZWL  UIDGNRC$W_SIZE(R11),R0  ; Temporarily point...
   58 5B    50   C1  030B    849         ADDL3   R0,R11,R8               ; ...to the next record in the list
               04   91  030F    850         CMPB    #UID$K_UCB_RTYPE,-      ; Is this a UCB record?
            06 A8       0311    851                 UIDGNRC$B_TYPE(R8)
               06   13  0313    852         BEQL    50$                     ; BR if so
             009F   30  0315    853         BSBW    SKIP_CLUS_RECORDS       ; We can't figure out device, skip it
             0076   31  0318    854         BRW     70$
                        031B    855 50$:
      54    09 A8   9A  031B    856         MOVZBL  UIDUCB$B_DEVCLASS(R8),R4 ; We now need to check to see...
      55    0A A8   9A  031F    857         MOVZBL  UIDUCB$B_DEVTYPE(R8),R5  ; ...if this device is known to UETP
                        0323    858         $FAO_S  CTRSTR = CS1,-          ; Try first for an exact match...
                        0323    859                 OUTBUF = FAO_ALT,-
                        0323    860                 OUTLEN = ALTBUF_PTR,-
```

```
                                0323   861                 P1 = R4,-
                                0323   862                 P2 = R5
      5A  014D'CF  0145'CF  39  033A   863       MATCHC    ALTBUF_PTR,ALTBUF,R10,- ; ...of device class and type
                   036B'DF      0342
                                0345   864                 @OUTADDRESS
                          26    13  0345   865       BEQL      60$                   ; BR if we have a hit
                                0347   866       $FAO_S    CTRSTR = CS3,-          ; Next try a wildcard match...
                                0347   867                 OUTBUF = FAO_ALT,-
                                0347   868                 OUTLEN = ALTBUF_PTR,-   ; ...on device class only
                                0347   869                     P1 = R4
      5A  014D'CF  0145'CF  39  035C   870       MATCHC    ALTBUF_PTR,ALTBUF,R10,- ; Match device class only
                   036B'DF      0364
                                0367   871                 @OUTADDRESS
                          04    13  0367   872       BEQL      60$                   ; BR if we matched
                          30    10  0369   873       BSBB      SUPDEV_CLUS_DENOSU    ; We can't find device - complain...
                          24    11  036B   874       BRB       70$                   ; ...and skip to the next
                                036D   875 60$:
                                036D   876
                    55    5E    D0  036D   877       MOVL      SP,R5                 ; Save ASCIC pointer for $FAO
                                0370   878       $FAO_S    CTRSTR = DDB_CTRSTR,-   ; Set up a default UETINIDEV DDB line
                                0370   879                 OUTLEN = BUFFER_PTR,-
                                0370   880                 OUTBUF = FAO_BUF,-
                                0370   881                 P1     = R5
                      0327      30  0385   882       BSBW      POTENTIALLY_OK        ; Set up UETINIDEV.DAT
                      03CE      30  0388   883       BSBW      CREATE_SUBPROCESS     ; See if the device is really testable
                      059D      30  038B   884       BSBW      COPY_LOG_FILE         ; Copy the results of the test
                      0774      30  038E   885       BSBW      FINISH_CONTROLLER     ; Analyze the results of the test
                                0391   886 70$:
                        5E      D6  0391   887       INCL      SP                    ; Clean up ASCIC device name...
                5E  019D'CF     C0  0393   888       ADDL2     DEVDSC,SP             ; ...from the stack
                      FF14      31  0398   889       BRW       10$                   ; Loop to process next record
                                039B   890
                                039B   891 ;
                                039B   892 ; We found a device class and type with no match in UETSUPDEV.DAT.  Complain
                                039B   893 ; and skip over it.
                                039B   894 ;
                                039B   895 SUPDEV_CLUS_DENOSU:
                   019D'CF      DF  039B   896       PUSHAL    DEVDSC                ; No match at all,...
                   000F'CF      DF  039F   897       PUSHAL    TEST_NAME             ; ...so yell at the user
                          02    DD  03A3   898       PUSHL     #2
                00748333      8F  DD  03A5   899       PUSHL     #UETP$_DENOSU
                          00    F0  03AB   900       INSV      #STS$K_WARNING,-
                          00        03AD   901                 #STS$V_SEVERITY,-
                        6E    03  03AE   902                 #STS$S_SEVERITY,(SP)  ; Set the severity code
            00000000'GF       04  FB  03B0   903       CALLS     #4,G^LIB$SIGNAL       ; And print the message
                                03B7   904       ;BSBB     SKIP_CLUS_RECORDS     ; Fall into SKIP_CLUS_RECORDS
                                03B7   905 ;
                                03B7   906 ; We can't test this device for some reason, so skip over this DDB and any
                                03B7   907 ; associated UCBs.  Any complaints about the situation have been issued.
                                03B7   908 ; R11 still points to the DDB record from UETP$CLSIODB.  Note that we must
                                03B7   909 ; look at each record we want to skip rather than chain to the next record
                                03B7   910 ; of a specific type and that we skip the first record (DDB) without checking
                                03B7   911 ; its type.
                                03B7   912 ;
                                03B7   913 SKIP_CLUS_RECORDS:
                50    04  AB    3C  03B7   914       MOVZWL    UIDGNRC$W_SIZE(R11),R0  ; Pick up the record length...
                   5B   50    C0  03BB   915       ADDL2     R0,R11                ; ...and point to the next record
```

```
                                                N 7
                04  91  03BE   916          CMPB    #UID$K_UCB_RTYPE,-      ; Is it a UCB record?
06 AB                   03C0   917                  UIDGNR[$B_TYPE(R11)
                F3  13  03C2   918          BEQL    SKIP_CLUS_RECORDS       ; Loop if so
                    05  03C4   919          RSB
```

```
                                03C5    921 ;
                                03C5    922 ; A DDB record has a testable device if the DDB name doesn't appear on the
                                03C5    923 ; list of known but untestable devices and if the first UCB attached to the
                                03C5    924 ; DDB has a device class and type that match something in UETSUPDEV.DAT.
                                03C5    925 ;
                                03C5    926 ; Because the data returned for shared memory are similar to those for
                                03C5    927 ; ordinary peripherals, we can share most of the same code.  We will
                                03C5    928 ; occasionally need to pick up data from different places.
                                03C5    929 ;
                                03C5    930 ; R10 and R11 are already assigned as the length of the UETSUPDEV section
                                03C5    931 ; and the pointer to the current I/O database record, respectively.
                                03C5    932 ;
                                03C5    933 MPM_RECORD:
              57   0269'CF   9E 03C5    934         MOVAB   MPM_LITERAL+1,R7        ; Shared memory name is always "MPM"
                56   FF A7   9A 03CA    935         MOVZBL  -1(R7),R6               ; Parallel actions for DDB record
              019D'CF    56   7D 03CE    936         MOVQ    R6,DEVDSC               ; Put desc in a safe place
                    58   5B   D0 03D3    937         MOVL    R11,R8                  ; Fake a temp pointer to next record
014D'CF  012A'CF   0122'CF   28 03D6    938         MOVC3   MPM_CS,MPM_CS+8,ALTBUF  ; Fake $FAO device class and type
         0145'CF   0122'CF   B0 03E0    939         MOVW    MPM_CS,ALTBUF_PTR       ; Fake $FAO OUTLEN
                     0082   31 03E7    940         BRW     MPM_AND_DDB_BOTH
                            03EA    941
                            03EA    942
                            03EA    943 DDB_RECORD:
                57   0C AB   9E 03EA    944         MOVAB   UIDDDB$T_NAME+1(R11),R7 ; From the ASCIC, form a descriptor...
            019D'CF   FF A7   9A 03EE    945         MOVZBL  -1(R7),DEVDSC           ; ...
              01A1'CF   57   D0 03F4    946         MOVL    R7,DEVDSC+4             ; ...to the controller name
                            03F9    947         ; See the note where KNOWN_BUT_NOT_TESTABLE table is defined.
      0040 8F   67   019D'CF   39 03F9    948         MATCHC  DEVDSC,(R7),#KBNT_LENGTH,- ; See if specific device is known...
                     00B8'CF        0401    949                 KNOWN_BUT_NOT_TESTABLE     ; ...but not testable
                          11   13 0404    950         BEQL    10$                     ; BR if we found it in table
                50   57   01   C3 0406    951         SUBL3   #1,R7,R0                ; See if generic device is known but...
      0040 8F   60   019D'CF   39 040A    952         MATCHC  DEVDSC,(R0),#KBNT_LENGTH,- ; ...not testable. (We strip "c"...
                     00B8'CF        0412    953                 KNOWN_BUT_NOT_TESTABLE  ; ...of "devc" but include ASCIC count)
                          03   12 0415    954         BNEQ    20$                     ; BR if device may be testable
                               0417    955 10$:
                     00A3   31 0417    956         BRW     SKIP_RECORDS            ; Device is not testable, skip it
                               041A    957 20$:
                  50   04 AB   3C 041A    958         MOVZWL  UIDGNRC$W_SIZE(R11),R0  ; Temporarily point...
              58   5B   50   C1 041E    959         ADDL3   R0,R11,R8               ; ...to the next record in the list
                          04   91 0422    960         CMPB    #UID$K_UCB_RTYPE,-      ; Is this a UCB record?
                     06 A8        0424    961                 UIDGNRC$B_TYPE(R8)
                          03   13 0426    962         BEQL    30$
                     0092   31 0428    963         BRW     SKIP_RECORDS            ; BR if not, we can't figure out device
                54   09 A8   9A 042B    964 30$:    MOVZBL  UIDUCB$B_DEVCLASS(R8),R4 ; We now need to check to see...
                55   0A A8   9A 042F    965         MOVZBL  UIDUCB$B_DEVTYPE(R8),R5 ; ...if this device is known to UETP
                               0433    966         $FAO_S  CTRSTR = CS1,-          ; Try first for an exact match...
                               0433    967                 OUTBUF = FAO_ALT,-
                               0433    968                 OUTLEN = ALTBUF_PTR,-
                               0433    969                     P1 = R4,-
                               0433    970                     P2 = R5
5A  014D'CF    0145'CF   39 044A    971         MATCHC  ALTBUF_PTR,ALTBUF,R10,- ; ...of device class and type
                036B'DF        0452
                               0455    972                 @OUTADDRESS
                          22   13 0455    973         BEQL    SUPDEV_MATCH            ; BR if we have a hit
                               0457    974         $FAO_S  CTRSTR = CS3,-          ; Next try a wildcard match...
                               0457    975                 OUTBUF = FAO_ALT,-
                               0457    976                 OUTLEN = ALTBUF_PTR,-   ; ...on device class only
```

```
                                0457   977                      P1 = R4
                                046C   978
                                046C   979 MPM_AND_DDB_BOTH:
      5A  014D'CF  0145'CF  39  046C   980          MATCHC  ALTBUF_PTR,ALTBUF,R10,- ; Match device class only
                   036B'DF          0474
                                0477   981                  @OUTADDRESS
                          28    12  0477   982          BNEQ    SUPDEV_DENOSU           ; BR if we can't find this device
                                0479   983 SUPDEV_MATCH:
                       55 FF A7   DE  0479   984          MOVAL   -1(R7),R5               ; Save ASCIC pointer for $FAO
                                047D   985          $FAO_S  CTRSTR = DDB_CTRSTR,-   ; Set up a default UETINIDEV DDB line
                                047D   986                  OUTLEN = BUFFER_PTR,-
                                047D   987                  OUTBUF = FAO_BUF,-
                                047D   988                  P1     = R5
                             021A   30  0492   989          BSBW    POTENTIALLY_OK          ; Set up UETINIDEV.DAT
                             02C1   30  0495   990          BSBW    CREATE_SUBPROCESS       ; See if the device is really testable
                             0490   30  0498   991          BSBW    COPY_LOG_FILE           ; Copy the results of the test
                             0667   30  049B   992          BSBW    FINISH_CONTROLLER       ; Analyze the results of the test
                             FDDB   31  049E   993          BRW     BUILD_INIDEV            ; Loop for the next device
                                04A1   994
                                04A1   995 ;
                                04A1   996 ; We found a device class and type with no match in UETSUPDEV.DAT.  Complain
                                04A1   997 ; and skip over it.
                                04A1   998 ;
                                04A1   999 SUPDEV_DENOSU:
                       019D'CF  DF  04A1  1000          PUSHAL  DEVDSC                  ; No match at all,...
                       000F'CF  DF  04A5  1001          PUSHAL  TEST_NAME               ; ...so yell at the user
                             02 DD  04A9  1002          PUSHL   #2
                   00748333 8F  DD  04AB  1003          PUSHL   #UETP$_DENOSU
                             00 F0  04B1  1004          INSV    #STS$K_WARNING,-
                             00     04B3  1005                  #STS$V_SEVERITY,-
                          6E 03     04B4  1006                  #STS$S_SEVERITY,(SP)    ; Set the severity code
                00000000'GF 04 FB  04B6  1007          CALLS   #4,G^LIB$SIGNAL         ; And print the message
                                04BD  1008          ;BRB    SKIP_RECORDS            ; Fall into SKIP_RECORDS
                                04BD  1009 ;
                                04BD  1010 ; We can't test this device for some reason, so skip over records until we
                                04BD  1011 ; get to the next DDB or MPM in the UETP$CLSIODB group.  Any complaints about
                                04BD  1012 ; the situation have been issued.  R11 still points to the original DDB or
                                04BD  1013 ; MPM record from UETP$CLSIODB.  Note that we must look at each record as we
                                04BD  1014 ; skip over it rather than chain directly to the next record of a given type.
                                04BD  1015 ;
                                04BD  1016 SKIP_RECORDS:
                          56 04 D0  04BD  1017          MOVL    #UID$K_UCB_RTYPE,R6     ; Assume we will skip UCB records
                             03 91  04C0  1018          CMPB    #UID$K_DDB_RTYPE,-      ; See what we are actually skipping
                          06 AB     04C2  1019                  UIDGNRC$B_TYPE(R11)
                             03 13  04C4  1020          BEQL    10$                     ; BR if our assumption was correct
                          56 00 D0  04C6  1021          MOVL    #UID$K_NULL_RTYPE,R6    ; We will skip exactly one MPM record
                                04C9  1022 10$:
                50 04 AB  3C  04C9  1023          MOVZWL  UIDGNRC$W_SIZE(R11),R0  ; Pick up the record length...
                   5B 50  C0  04CD  1024          ADDL2   R0,R11                  ; ...and point to the next record
                06 AB 56  91  04D0  1025          CMPB    R6,UIDGNRC$B_TYPE(R11)  ; Is this a record we want to skip?
                             F3 13  04D4  1026          BEQL    10$                     ; BR if it is
                           FDA3 31  04D6  1027          BRW     BUILD_INIDEV            ; We've skipped intervening records
```

```
                                04D9  1029          .SBTTL  Summarize UETINIDEV.DAT
                                04D9  1030
                                04D9  1031 ;+
                                04D9  1032 ;       We've tried out all the supportable devices in the system and have
                                04D9  1033 ;       listed in UETINIDEV.DAT the testable/non-testable status of each.
                                04D9  1034 ;       Summarize those results in a more humanly readable form.  We will
                                04D9  1035 ;       keep certain register assignments throughout the summary:
                                04D9  1036 ;               R11 - Pointer to RAB for UETINIDEV.DAT
                                04D9  1037 ;               R10 - Count of characters on a line.  May use ALTBUF(R10)
                                04D9  1038 ;               R9  - Unit record type (T or N) targetted by APPEND_UNITS
                                04D9  1039 ;               R8  - Count of unwanted records found by APPEND_UNITS
                                04D9  1040 ;               R7  - Pointer to string descriptor for SUMM_OUTPUT
                                04D9  1041 ;-
                                04D9  1042
                                04D9  1043 DO_SUMMARY:
    5B    0514'CF    DE    04D9  1044          MOVAL   INI_RAB,R11             ; Keep pointer handy to UETINIDEV
28  AB    020C'CF    DE    04DE  1045          MOVAL   END_MSG,RAB$L_RBF(R11)  ; Set the end message address
  22  AB      14     B0    04E4  1046          MOVW    #END_MSGL,RAB$W_RSZ(R11) ; Set the end message size
                           04E8  1047          $PUT    RAB = (R11),-           ; Write the end message
                           04E8  1048                  ERR = RMS_ERROR
                           04F5  1049          $FLUSH  RAB = (R11),-           ; Make sure everything gets output
                           04F5  1050                  ERR = RMS_ERROR
                           0502  1051          $REWIND RAB = (R11),-           ; Go back to beginning of UETINIDEV
                           0502  1052                  ERR = RMS_ERROR
    57    0145'CF    DE    050F  1053          MOVAL   ALTBUF_PTR,R7           ; Skip a line before we start...
              67     B4    0514  1054          CLRW    (R7)                    ; ...
            0121     30    0516  1055          BSBW    SUMM_OUTPUT             ; ...
    57    02A2'CF    DE    0519  1056          MOVAL   SUMM_HEADER,R7          ; Set up to output summary header
            0119     30    051E  1057          BSBW    SUMM_OUTPUT             ; Go output line
    57    0145'CF    DE    0521  1058          MOVAL   ALTBUF_PTR,R7           ; We'll write from here, evermore
                           0526  1059          $GET    RAB = (R11),-           ; Prime pump with the first DDB record
                           0526  1060                  ERR = RMS_ERROR
                           0533  1061 DDB_LOOP:
              67     B4    0533  1062          CLRW    (R7)                    ; Skip a line between controllers
            0102     30    0535  1063          BSBW    SUMM_OUTPUT             ; Go output empty line
0043'CF     44  8F    91    0538  1064          CMPB    #^A/D/,BUFFER           ; Is the next line for a DDB?
              1C     13    053E  1065          BEQL    20$                     ; BR if it is
0043'CF     45  8F    91    0540  1066          CMPB    #^A/E/,BUFFER           ; Or is it the end of UETINIDEV.DAT?
              03     12    0546  1067          BNEQ    10$                     ; BR if not - we have an error
            0126     31    0548  1068          BRW     SUC_EXIT                ; Summary complete, finish UETINIT01
                           054B  1069 10$:
         02DE'CF     DF    054B  1070          PUSHAL  FORMAT_ERR_MSG          ; Something is wrong with...
              01     DD    054F  1071          PUSHL   #1                      ; ...UETINIDEV.DAT - output msg and...
  00741132  8F     DD    0551  1072          PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...quit
              03     DD    0557  1073          PUSHL   #3                      ; Current arg count
            093A     31    0559  1074          BRW     ERROR_EXIT
                           055C  1075 20$:
                           055C  1076 ;
                           055C  1077 ; We have a DDB line from UETINDEV.DAT.  We know that if the controller is
                           055C  1078 ; not testable, no devices will be testable (FINISH_CONTROLLER routine).
                           055C  1079 ;
         026C'CF     19    28    055C  1080          MOVC3   #TESTABLE_LEN,TESTABLE,- ; Set up initial msg
              014D'CF            0561  1081                  ALTBUF
              5A     19    D0    0564  1082          MOVL    #TESTABLE_LEN,R10       ; Set up count of chars on line
    50    22  AB     06    A3    0567  1083          SUBW3   #6,RAB$W_RSZ(R11),R0    ; Calc controller name length
014D'CF     0049'CF    50    28    056C  1084          MOVC3   R0,BUFFER+6,ALTBUF      ; Put controller name in output...
                           0574  1085                                              ; ...buffer, overwriting blanks
```

```
                63   3A   90  0574  1086          MOVB    #^A/:/,(R3)                     ; Insert ":" after controller name
                                0577  1087          $FIND   RAB = (R11),-                   ; Move RFA to the first UCB...
                                0577  1088                  ERR = RMS_ERROR                 ; ...but leave BUFFER unscathed
076A'CF    10 AB   06   28  0584  1089          MOVC3   #6,RAB$W_RFA(R11),END_RFA       ; Save 1st UCB RFA for poss reread
     0047'CF   54 8F   91  058B  1090          CMPB    #^A/T/,BUFFER+4                 ; Is DDB marked testable?
                   10   13  0591  1091          BEQL    UCB_T                           ; BR if it is
         029E'CF   04   28  0593  1092          MOVC3   #NONE_LEN,NONE,-                ; Indicate DDB has no testable units
                0166'CF         0598  1093                  ALTBUF+TESTABLE_LEN
                67   1D   B0  059B  1094          MOVW    #TESTABLE_LEN+NONE_LEN,(R7) ; Set line length indicating that
                  0099   30  059E  1095          BSBW    SUMM_OUTPUT                     ; Write the line
                    16   11  05A1  1096          BRB     UCB_N                           ; Now summarize the untestable units
                                05A3  1097 UCB_T:
              59   54 8F   9A  05A3  1098          MOVZBL  #^A/T/,R9                       ; We will search for testable units
                  0024   30  05A7  1099          BSBW    APPEND_UNITS                    ; Form line(s) of testable units
                    58   D5  05AA  1100          TSTL    R8                              ; Any untestable units found?
                    85   13  05AC  1101          BEQL    DDB_LOOP                        ; Get next controller if not
            1E AB   02   90  05AE  1102          MOVB    #RAB$C_RFA,RAB$B_RAC(R11)       ; Some untestables - we must reread
10 AB   076A'CF   06   28  05B2  1103          MOVC3   #6,END_RFA,RAB$W_RFA(R11)       ; Point us back to first UCB
                                05B9  1104 UCB_N:
                    19   28  05B9  1105          MOVC3   #TESTABLE_LEN,-                 ; Set up a line for untestable units
   014D'CF    0285'CF         05BB  1106                  UNTESTABLE,ALTBUF
                5A   19   D0  05C1  1107          MOVL    #TESTABLE_LEN,R10               ; Set up count of chars on line
              59   4E 8F   9A  05C4  1108          MOVZBL  #^A/N/,R9                       ; We'll search for untestable units
                  0003   30  05C8  1109          BSBW    APPEND_UNITS                    ; Form line(s) of untestable units
                  FF65   31  05CB  1110          BRW     DDB_LOOP                        ; This controller is summarized
```

```
                        05CE  1112 ;
                        05CE  1113 ; Append unit numbers to a line.  Wrap around to a new line if the line gets
                        05CE  1114 ; too long.  Write out the lines when done or when wrapping to a new line.
                        05CE  1115 ; Register assignments are listed near DO_SUMMARY.
                        05CE  1116 ;
                        05CE  1117 APPEND_UNITS:
                  58    D4  05CE  1118          CLRL    R8                              ; Init counter of unwanted records
                        05D0  1119 10$:
                        05D0  1120          $GET    RAB = (R11),-                   ; Get some record - our access mode...
                        05D0  1121                  ERR = RMS_ERROR                 ; ...is determined externally
              1E AB     00    90  05DD  1122          MOVB    #RAB$C_SEQ,RAB$B_RAC(R11) ; We must read UCBs sequentially
      0043'CF   55  8F    91  05E1  1123          CMPB    #^A/U/,BUFFER                   ; Have we a UCB?
                        06    13  05E7  1124          BEQL    20$                             ; Continue on if so
           67   5A      02    A3  05E9  1125          SUBW3   #2,R10,(R7)                     ; Write line, stripping separator
                        4B    11  05ED  1126          BRB     SUMM_OUTPUT                     ; Exit APPEND_UNITS via SUMM_OUTPUT
                        05EF  1127 20$:
           0047'CF      59    91  05EF  1128          CMPB    R9,BUFFER+4                     ; Is this the kind of unit we want?
                        04    13  05F4  1129          BEQL    30$                             ; BR if so
                        58    D6  05F6  1130          INCL    R8                              ; Count opposites if not
                        D6    11  05F8  1131          BRB     10$                             ; Continue with next record
                        05FA  1132 30$:
                 04     30    3B  05FA  1133          SKPC    #^A/0/,#MAX_UNIT_DESIG-1,- ; Skip over leading zeros...
                 0049'CF        05FD  1134                  BUFFER+6                        ;  ...in unit number
                        50    D6  0600  1135          INCL    R0                              ; Compensate for short length search
           52   50      02    A1  0602  1136          ADDW3   #2,R0,R2                        ; Figure out if we can...
                52      5A    A0  0606  1137          ADDW2   R10,R2                          ; ...fit unit number and separator...
           52   0050    8F    B1  0609  1138          CMPW    #MAX_SUMM_LINE,R2               ; ...on this line
                        15    18  060E  1139          BGEQ    40$                             ; BR if we will fit
                67      5A    B0  0610  1140          MOVW    R10,(R7)                        ; We need to wrap to a new line
                        03    BB  0613  1141          PUSHR   #^M<R0,R1>                      ; Save these over routine call
                        23    10  0615  1142          BSBB    SUMM_OUTPUT                     ; Write out the old line first
              20    00    8F    00  0617  1143          MOVC5   #0,#0,#^A/ /,-                  ; Start out the next line by lining...
                 014D'CF        19  061C  1144                  #TESTABLE_LEN,ALTBUF            ; ...up next unit beneath the first
                        5A    19  D0  0620  1145          MOVL    #TESTABLE_LEN,R10               ; Reinitialize character count on line
                        03    BA  0623  1146          POPR    #^M<R0,R1>                      ; Restore length and pointer to unit
                        0625  1147 40$:
        014D'CA   61    50    28  0625  1148          MOVC3   R0,(R1),ALTBUF(R10)             ; Append the unit to the line
              83    202C    8F    B0  062B  1149          MOVW    #^A/, /,(R3)+                   ; Append separator to that
   5A   53    0000014D'     8F    C3  0630  1150          SUBL3   #ALTBUF,R3,R10                  ; Calculate new line length
                        96    11  0638  1151          BRB     10$                             ; Go look for the next unit
                        063A  1152
                        063A  1153 ;
                        063A  1154 ; Write a line to the UETINIT01 log file.  If long report is specified,
                        063A  1155 ; write that same line to SYS$OUTPUT also.
                        063A  1156 ;
                        063A  1157 SUMM_OUTPUT:
              06AE'CF   67    B0  063A  1158          MOVW    (R7),LOG_RAB+RAB$W_RSZ  ; Get the message size
           06B4'CF    04   A7    D0  063F  1159          MOVL    4(R7),LOG_RAB+RAB$L_RBF ; Set the message address
                        0645  1160          $PUT    RAB = LOG_RAB,-                 ; Write the log file
                        0645  1161                  ERR = RMS_ERROR
           16 0021'CF   00    E0  0654  1162          BBS     #SHRT_RPRTV,FLAG,10$    ; Skip if short report
              002F'CF   57    D0  065A  1163          MOVL    R7,MSG_PTR              ; Set up $PUTMSG message descriptor
                        065F  1164          $PUTMSG_S MSGVEC = LOG_RCD      ; Write everything to SYS$OUTPUT
                        0670  1165 10$:
                        05    0670  1166          RSB
```

```
                     0671  1168          .SBTTL  Close Up Shop
                     0671  1169
                     0671  1170 SUC_EXIT:
                     0671  1171          $FAO_S  CTRSTR = TST_CNT_STR,-  ; Create a string giving...
                     0671  1172                  OUTBUF = FAO_BUF,-      ; ...the number of testable controllers
                     0671  1173                  OUTLEN = BUFFER_PTR,-
                     0671  1174                  P1     = TST_CNT
                     0688  1175          $CRELOG_S TBLFLG = #T,-         ; Create a logical name for...
                     0688  1176                    LOGNAM = TEST_COUNT,- ; ...the count of testable controllers
                     0688  1177                    EQLNAM = BUFFER_PTR
 10000001 8F    DO   069B  1178          MOVL    #SS$_NORMAL!STS$M_INHIB_MSG,- ; Set successful exit status
      0357'CF        06A1  1179                  STATUS
                     06A4  1180          $EXIT_S STATUS                  ; Exit with the status
```

```
                     06AF  1182          .SBTTL  BUILD_INIDEV Routines - POTENTIALLY_OK
                     06AF  1183 ;
                     06AF  1184 ; We have a device (controller and unit number) that we've identified as
                     06AF  1185 ; potentially testable.  To see if it really is, have its associated image
                     06AF  1186 ; (from UETSUPDEV) check out each unit.  The DDB and UCB or MPM info has to be
                     06AF  1187 ; written to UETINIDEV for that.  The DDB line is described by BUFFER_PTR and
                     06AF  1188 ; R8 points to the first UCB or MPM record.  R3 is left over from a MATCHC.
                     06AF  1189 ; We are still in the BUILD_INIDEV loop, so R10 and R11 are in use as
                     06AF  1190 ; described therein.
                     06AF  1191 ;
                     06AF  1192 POTENTIALLY_OK:
        59    53     D0  06AF  1193          MOVL    R3,R9                   ; Save pointer to start of image name
        5B    58     D0  06B2  1194          MOVL    R8,R11                  ; It's safe to update real database ptr
  003B'CF    B0  06B5  1195          MOVW    BUFFER_PTR,-            ; Set line size
  0536'CF          06B9  1196                  INI_RAB+RAB$W_RSZ
                   06BC  1197          $PUT    RAB = INI_RAB,-         ; Write a UETINIDEV.DAT line...
                   06BC  1198                  ERR = RMS_ERROR         ; ...for the DDB or MPM
0524'CF    06    28  06CB  1199          MOVC3   #6,INI_RAB+RAB$W_RFA,-  ; Save RFA so we can reread record
  0764'CF          06D0  1200                  DDB_RFA
                   06D3  1201 10$:
     56    07 AB    3C  06D3  1202          MOVZWL  UIDUCB$W_NUMBER(R11),R6 ; Assume we're looking at a UCB
           04    91  06D7  1203          CMPB    #UID$K_UCB_RTYPE,-      ; But are we really?
        06 AB          06D9  1204                  UIDGNRC$B_TYPE(R11)
           04    13  06DB  1205          BEQL    20$                     ; BR if we are
     56    07 AB    3C  06DD  1206          MOVZWL  UIDMPM$W_NUMBER(R11),R6 ; Get the equivalent MPM value if not
                   06E1  1207 20$:
                   06E1  1208          $FAO_S  CTRSTR = UCB_CTRSTR,-   ; Form a UETINIDEV line for UCB or MPM
                   06E1  1209                  OUTLEN = BUFFER_PTR,-
                   06E1  1210                  OUTBUF = FAO_BUF,-
                   06E1  1211                      P1 = R6
  003B'CF    B0  06F6  1212          MOVW    BUFFER_PTR,-            ; Set line size
  0536'CF          06FA  1213                  INI_RAB+RAB$W_RSZ
                   06FD  1214          $PUT    RAB = INI_RAB,-         ; Write a UETINIDEV.DAT line...
                   06FD  1215                  ERR = RMS_ERROR         ; ...for the UCB or MPM
     50    04 AB    3C  070C  1216          MOVZWL  UIDGNRC$W_SIZE(R11),R0  ; Pick up the record length...
        5B    50    C0  0710  1217          ADDL2   R0,R11                  ; ...to point to the next record
           04    91  0713  1218          CMPB    #UID$K_UCB_RTYPE,-      ; Is this also a UCB?
        06 AB          0715  1219                  UIDGNRC$B_TYPE(R11)
              BA    13  0717  1220          BEQL    10$                     ; BR if it is
           05    91  0719  1221          CMPB    #UID$K_MPM_RTYPE,-      ; Or is this also an MPM record?
        06 AB          071B  1222                  UIDGNRC$B_TYPE(R11)
              B4    13  071D  1223          BEQL    10$                     ; BR if it is
                   071F  1224
053C'CF    020C'CF    DE  071F  1225          MOVAL   END_MSG,INI_RAB+RAB$L_RBF ; Set up to write end-of-file line
  0536'CF    14    B0  0726  1226          MOVW    #END_MSGL,INI_RAB+RAB$W_RSZ
                   072B  1227          $PUT    RAB = INI_RAB,-         ; Write the end of file line
                   072B  1228                  ERR = RMS_ERROR
  0524'CF    06    28  073A  1229          MOVC3   #6,INI_RAB+RAB$W_RFA,-  ; Save the RFA of the END line...
  076A'CF          073F  1230                  END_RFA                 ; ...to make truncation faster
                   0742  1231          $FLUSH  RAB = INI_RAB,-         ; Make sure the device test sees...
                   0742  1232                  ERR = RMS_ERROR         ; ...everything we think we wrote
053C'CF    0043'CF    DE  0751  1233          MOVAL   BUFFER,INI_RAB+RAB$L_RBF ; Restore typical buffer address
              05    0758  1234          RSB
```

```
                                0759  1236          .SBTTL  BUILD_INIDEV Routines - CREATE_SUBPROCESS
                                0759  1237 ;
                                0759  1238 ; Make the sub-process which will see if the devices found above have any
                                0759  1239 ; testable units.  R9 still points to the start of the image name from
                                0759  1240 ; the UETSUPDEV global section.  R10 and R11 are in use from BUILD_INIDEV.
                                0759  1241 ;
                                0759  1242 CREATE_SUBPROCESS:
          13 0021'CF    00    E0  0759  1243          BBS     #CHRT_RPRTV,FLAG,10$    ; Skip separation if short report
                                075F  1244          $PUTMSG_S MSGVEC = BLANK_LINE_PTR,- ; Separate all of this process'...
                                075F  1245                    ACTRTN = ACTRTN           ; ...msg's from the previous one's
                                0772  1246 10$:     $QIOW_S CHAN = MBX1_CHAN,-      ; Tell process's SYS$INPUT what to test
                                0772  1247                   EFN = #SS_SYNCH_EFN,-
                                0772  1248                  FUNC = #IO$_WRITEVBLK!IO$M_NOW,-
                                0772  1249                  IOSB = QUAD_STATUS,-
                                0772  1250                    P1 = @DEVDSC+4,-
                                0772  1251                    P2 = DEVDSC
            28 035B'CF    E8  0797  1252          BLBS    QUAD_STATUS,30$         ; BR if all went OK
          52   035B'CF    3C  079C  1253          MOVZWL  QUAD_STATUS,R2          ; Set up arglist so we can appease...
                 04C5    30  07A1  1254          BSBW     AO_CHECK               ; ...LIB$SIGNAL's use of $PUTMSG
                   52    DD  07A4  1255          PUSHL   R2                      ; Save flags and FAO count for $PUTMSG
                   02    13  07A6  1256          BEQL    20$                     ; BR if there are no FAO args
                   01    DD  07A8  1257          PUSHL   #1                      ; There are args - save dummy for count
                         07AA  1258 20$:
          52   8E    04    C1  07AA  1259          ADDL3   #4,(SP)+,R2             ; Pop bogus arg and get arg count
          7E   035B'CF    3C  07AE  1260          MOVZWL  QUAD_STATUS,-(SP)       ; Save the error status
               049B'CF    DF  07B3  1261          PUSHAL  MBXW_QIO_FAIL           ; Explain what went wrong
                   01    DD  07B7  1262          PUSHL   #1
           00741132 8F    DD  07B9  1263          PUSHL   #UETP$_TEXT!STS$K_ERROR
                   52    DD  07BF  1264          PUSHL   R2                      ; Arg count for ERROR_EXIT
                 06D2    31  07C1  1265          BRW     ERROR_EXIT
                         07C4  1266 30$:
      02AD'CF   69    02A5'CF   28  07C4  1267          MOVC3   TEST_DSC,(R9),TEST_IMAGE ; Set up image filespec to test device
            02BE'CF    02A5'CF   3C  07CC  1268          MOVZWL  TEST_DSC,LOGNAM_DSC     ; SYS$OUTPUT for that image...
02C6'CF     02AD'CF    02A5'CF   28  07D3  1269          MOVC3   TEST_DSC,TEST_IMAGE,LOGNAM ; ...will be the image name...
      02C6'CF    02BE'CF    2E   3A  07DD  1270          LOCC    #^A/./,LOGNAM_DSC,LOGNAM ; ...but have a file type...
                 61    01B3'CF   D0  07E5  1271          MOVL    LOGEXT,(R1)             ; ...of of .LOG
                         07EA  1272          $CREPRC_S IMAGE  = TEST_DSC,-   ; Create the subprocess to test device
                                07EA  1273                    PIDADR = PIDADR,-
                                07EA  1274                    INPUT  = TEST_NAME,-
                                07EA  1275                    OUTPUT = LOGNAM_DSC,-
                                07EA  1276                    ERROR  = SUBPROC_ERROR,-
                                07EA  1277                    QUOTA  = QUOTA_LIST,-
                                07EA  1278                    MBXUNT = MBX_UNIT
                                081A  1279          $SETIMR_S DAYTIM = THREEMIN,-   ; Set up a timer to catch test hangs
                                081A  1280                    ASTADR = KILL_SUBPROCESS,-
                                081A  1281                    REQIDT = PIDADR
                                082F  1282          $QIOW_S CHAN = MBXCHN,-         ; Set up read for termination mailbox
                                082F  1283                   EFN = #SS_SYNCH_EFN,-
                                082F  1284                  FUNC = #IO$_READVBLK,-
                                082F  1285                  IOSB = QUAD_STATUS,-
                                082F  1286                    P1 = MBX_BUF,-
                                082F  1287                    P2 = #MBX_SIZE
                                0854  1288          $CANTIM_S REQIDT = PIDADR       ; QIOW finished, we're no longer hung
                   02BA'CF    D4  0861  1289          CLRL    PIDADR                  ; Indicate to us that proc is finished
                3F 035B'CF    E8  0865  1290          BLBS    QUAD_STATUS,50$         ; BR if all went OK
                   02D4 8F    3C  086A  1291          MOVZWL  #SS$_OPINCOMPL,-        ; Assume for now that we $CANCELled...
                   01A9'CF        086E  1292                  MBX_BUF+ACC$L_FINALSTS  ; ...the QIOW
```

```
035B'CF    0830 8F   B1  0871          CMPW    #SS$_CANCEL,QUAD_STATUS ; But did we really?
                2F   13  0878          BEQL    50$                     ; BR if so, we can recover
     035B'CF    2C   B1  087A          CMPW    #SS$_ABORT,QUAD_STATUS  ; We get SS$_ABORT if I/O in progress
                28   13  087F          BEQL    50$                     ; BR if so, we can recover
     52   035B'CF    3C  0881          MOVZWL  QUAD_STATUS,R2          ; Set up arglist so we can appease...
              03E0   30  0886          BSBW    FAO_CHECK               ; ...LIB$SIGNAL's use of $PUTMSG
                52   DD  0889          PUSHL   R2                      ; Save flags and FAO count for $PUTMSG
                02   13  088B          BEQL    40$                     ; BR if there are no FAO args
                01   DD  088D          PUSHL   #1                      ; There are args - save dummy for count
                         088F  40$:
     52    8E   04   C1  088F          ADDL3   #4,(SP)+,R2             ; Pop bogus arg and get arg count
     7E   035B'CF    3C  0893          MOVZWL  QUAD_STATUS,(SP)        ; Save the error status
           04D1'CF   DF  0898          PUSHAL  CREPRC_CIO_FAIL         ; Explain what went wrong
                01   DD  089C          PUSHL   #1
      00741132 8F    DD  089E          PUSHL   #UETP$_TEXT!STS$K_ERROR
                52   DD  08A4          PUSHL   R2                      ; Arg count for ERROR_EXIT
              05ED   31  08A6          BRW     ERROR_EXIT
                         08A9  50$:
     52   01A9'CF    D0  08A9          MOVL    MBX_BUF+ACC$L_FINALSTS,R2 ; Save termination status
                08   12  08AE          BNEQ    60$                       ; BR if we got a real status
          52    2C   D0  08B0          MOVL    #SS$_ABORT,R2             ; No status, must have been...
                         08B3                                            ; ...$DELPRC, so give a dummy status
     01A9'CF    52   D0  08B3          MOVL    R2,MBX_BUF+ACC$L_FINALSTS ; Save status here, too
                         08B8  60$:
             6F 52   E8  08B8          BLBS    R2,80$                    ; BR if process completed OK
                         08BB          $FAO_S  CTRSTR = SUBPROCESS_FAIL,- ; Give some indication now...
                         08BB                  OUTLEN = BUFFER_PTR,-      ; ...if the subprocess failed...
                         08BB                  OUTBUF = FAO_BUF,-         ; ...so that we don't just...
                         08BB                      P1 = #TEST_DSC,-       ; ...die suddenly...
                         08BB                      P2 = #DEVDSC           ; ...if there's some other problem
              038C   30  08DA          BSBW    FAO_CHECK               ; See if error takes any FAO args
                52   DD  08DD          PUSHL   R2                      ; Save flags/FAO count for LIB$SIGNAL
                02   13  08DF          BEQL    70$                     ; BR if there are no FAO args
                01   DD  08E1          PUSHL   #1                      ; There are args - save dummy for count
                         08E3  70$:
     52    8E   04   C1  08E3          ADDL3   #4,(SP)+,R2             ; Pop bogus arg and get arg count
           01A9'CF   DD  08E7          PUSHL   MBX_BUF+ACC$L_FINALSTS
  6E   10000000 8F   CA  08EB          BICL    #STS$M_INHIB_MSG,(SP)   ; Allow message to be printed, always!
           003B'CF   DF  08F2          PUSHAL  BUFFER_PTR
      000F0001 8F    DD  08F6          PUSHL   #^XF0001
      00741132 8F    DD  08FC          PUSHL   #UETP$_TEXT!STS$K_ERROR
     0E61'CF    52   FB  0902          CALLS   R2,ERROR_MSG            ; Note that we don't exit on this error
                         0907          $QIO_S  CHAN = MBX1_CHAN,-      ; Throw away "SYS$INPUT" message,....
                         0907                  FUNC = #IO$_READVBLK!IO$M_NOW,-
                         0907                    P1 = @DEVDSC+4,-      ; ...it can screw up. .
                         0907                    P2 = DEVDSC           ; ...the next subprocess we create
                         092A  80$:
                     05  092A          RSB
```

```
                                                     .SBTTL  BUILD_INIDEV Routines - COPY_LOG_FILE
                                                     ;
                                                     ; Copy the log file of the process which just finished to our own log file.
                                                     ; If we are in "long report" mode, also copy the part of the log file between
                                                     ; the begining and ending "sentinels" (inclusive) to SYS$OUTPUT.  R10 and R11
                                                     ; are in use from BUILD_INIDEV.
                                                     ;
                                                     COPY_LOG_FILE:
                02BE'CF    90   092B  1350                 MOVB    LOGNAM_DSC,-                ; Set the file name size
                0704'CF         092F  1351                         TMPLOG_FAB+FAB$B_FNS
06FC'CF         02C6'CF    DE   0932  1352                 MOVAL   LOGNAM,TMPLOG_FAB+FAB$L_FNA ; Set the file name address
06B4'CF         0043'CF    DE   0939  1353                 MOVAL   BUFFER,LOG_RAB+RAB$L_RBF ; Set the buffer write address
                           0940  1354                 $OPEN   FAB = TMPLOG_FAB            ; Open the temp log file
                   58 50   E8   094B  1355                 BLBS    R0,20$                      ; BR if we can read it
          51    031E'CF    DE   094E  1356                 MOVAL   FNF_MSG,R1                  ; Assume the error was a missing log
     50    00000000'8F    D1   0953  1357                 CMPL    #RMS$_FNF,R0                ; Is the log file missing?
                     17    13   095A  1358                 BEQL    10$                         ; BR if so - a special case
          51    035D'CF    DE   095C  1359                 MOVAL   FLK_MSG,R1                  ; Assume now that the log was locked
     50    00000000'8F    D1   0961  1360                 CMPL    #RMS$_FLK,R0                ; Did the process not terminate?
                     09    13   0968  1361                 BEQL    10$                         ; BR if so - a special case
                06D0'CF    DF   096A  1362                 PUSHAL  TMPLOG_FAB                  ; There's some other error, so...
           0DAD'CF    01    FB   096E  1363                 CALLS   #1,RMS_ERROR                ; ...exit with a useful message
                           0973  1364 10$:
                           0973  1365                 $FAO_S  CTRSTR = (R1),-             ; Recover from this error,...
                           0973  1366                         OUTLEN = BUFFER_PTR,-       ; ...because it's probably...
                           0973  1367                         OUTBUF = FAO_BUF,-          ; ...a fault somewhere else,...
                           0973  1368                             P1 = #LOGNAM_DSC,-      ; ...but complain
                           0973  1369                             P2 = #DEVDSC
                003B'CF    DF   0990  1370                 PUSHAL  BUFFER_PTR                  ; Set up to give the message
             000F0001  8F    DD   0994  1371                 PUSHL   #^XF0001
             00741132  8F    DD   099A  1372                 PUSHL   #UETP$_TEXT!STS$K_ERROR
           0E61'CF    03    FB   09A0  1373                 CALLS   #3,ERROR_MSG
                      05    RSB  09A5  1374
                           09A6  1375 20$:
                           09A6  1376                 $CONNECT RAB = TMPLOG_RAB,-         ; Connect the temp log file
                           09A6  1377                          ERR = RMS_ERROR
      25 0021'CF    00    E0   09B5  1378                 BBS     #SHRT_RPRTV,FLAG,30$        ; Skip announcement in short report
01A1'DF         019D'CF    28   09BB  1379                 MOVC3   DEVDSC,@DEVDSC+4,-          ; Form message telling device to test...
                04B9'CF         09C2  1380                         TESTING_MSG_TEXT
                      08    A1   09C5  1381                 ADDW3   #TESTING_MSG_LENGTH,-       ; ...and its length
04A9'CF         019D'CF         09C7  1382                         DEVDSC,TESTING_MSG
                           09CD  1383                 $PUTMSG_S MSGVEC = LOG_BEGIN,-      ; Give msg saying what we're about to do
                           09CD  1384                           ACTRTN = ACTRTN
                           09E0  1385 30$:
                    008B   30   09E0  1386                 BSBW    COPY_RECORD                 ; Copy from temp log to our log file
                  69 50    E9   09E3  1387                 BLBC    R0,70$                      ; BR on any error
030C'CF         0304'CF    39   09E6  1388                 MATCHC  START_DESC,START_DESC+8,- ; Did we find beginning sentinel?
0397'CF         038F'CF         09ED  1389                         TEMP_BUFF_DESC,TEMP_BUFF_STR
                     EB    12   09F3  1390                 BNEQ    30$                         ; Loop if not
      11 0021'CF    00    E0   09F5  1391                 BBS     #SHRT_RPRTV,FLAG,40$        ; Sentinel to only log if short report
                           09FB  1392                 $PUTMSG_S MSGVEC = LOG_RCD          ; Copy sentinel line to SYS$OUTPUT
                           0A0C  1393 40$:
                      60   10   0A0C  1394                 BSBB    COPY_RECORD                 ; Copy from temp log to our log file
                  3E 50    E9   0A0E  1395                 BLBC    R0,70$                      ; BR on any error
0319'CF         0311'CF    39   0A11  1396                 MATCHC  STOP_DESC,STOP_DESC+8,- ; Did we find ending sentinel?
0397'CF         038F'CF         0A18  1397                         TEMP_BUFF_DESC,TEMP_BUFF_STR
                     13    13   0A1E  1398                 BEQL    50$                         ; Exit loop if we did
```

```
                          0A20  1399          $PUTMSG_S MSGVEC = LOG_RCD      ; Copy a line to SYS$OUTPUT
                    D9    0A31  1400          BRB     40$                     ; Loop for another temp log record
                          0A33  1401 50$:
          11 0021'CF    00    E0  0A33  1402          BBS     #SHRT_RPRTV,FLAG,60$    ; Sentinel to only log if short report
                          0A39  1403          $PUTMSG_S MSGVEC = LOG_RCD      ; Copy sentinel line to SYS$OUTPUT
                          0A4A  1404 60$:
                    22    10    0A4A  1405          BSBB    COPY_RECORD             ; Copy from temp log to our log file
                  FB 50   E8    0A4C  1406          BLBS    R0,60$                  ; Loop while there are more records
                          0A4F  1407 70$:
                          0A4F  1408          $CLOSE  FAB = TMPLOG_FAB,-      ; Close the temp log file
                          0A4F  1409                  ERR = RMS_ERROR
                          0A5E  1410          $ERASE  FAB = TMPLOG_FAB,-      ; Delete the temp log file
                          0A5E  1411                  ERR = RMS_ERROR
                    05    0A6D  1412          RSB                             ; Exit copy loop
```

```
                                0A6E  1414 ;
                                0A6E  1415 ; Massage a record from the (temp) log file of a device test process that's just
                                0A6E  1416 ; finished.  Massaging includes reading the record, indenting it from the left
                                0A6E  1417 ; margin, and forming an uppercase copy of the record so that it may be checked
                                0A6E  1418 ; for a beginning or ending sentinel.
                                0A6E  1419 ;
                                0A6E  1420 COPY_RECORD:
                                0A6E  1421          $GET    RAB = TMPLOG_RAB            ; Read a temp log record
                          13 50 E8    0A79  1422          BLBS    R0,20$                  ; BR if we have no error
                50 00000000'8F  D1    0A7C  1423          CMPL    #RMS$_EOF,R0            ; Is it EOF, i.e., reasonable error?
                             09 13    0A83  1424          BEQL    10$                     ; BR if so
                      0720'CF   DF    0A85  1425          PUSHAL  TMPLOG_RAB              ; We have some unknown error...
                  0DAD'CF   01  FB    0A89  1426          CALLS   #1,RMS_ERROR            ; ...so let RMS error handler eat it
                                05    0A8E  1427 10$:     RSB
                                0A8F  1428 20$:
                51  0748'CF   D0    0A8F  1429          MOVL    TMPLOG_RAB+RAB$L_RBF,R1 ; R1 and R0 are a string desc...
                50  0742'CF   3C    0A94  1430          MOVZWL  TMPLOG_RAB+RAB$W_RSZ,R0 ; ...for the remainder of the record
                    7E   50   B0    0A99  1431          MOVW    R0,-(SP)                ; Counts chars as indentation is done
                         1E   11    0A9C  1432          BRB     50$                     ; BR inside loop - indent string's start
                61  50   0D   3A    0A9E  1433 30$:     LOCC    #13,R0,(R1)             ; Is there a <RET> in rest of string?
                         35   13    0AA2  1434          BEQL    60$                     ; Exit loop if not - no more indent
                         50   D7    0AA4  1435          DECL    R0                      ; Found one. LOCC has us pointing at it
                         51   D6    0AA6  1436          INCL    R1                      ; Point past the <RET>
                    61   0A   91    0AA8  1437          CMPB    #10,(R1)                ; Is there a <LINEFEED>?
                         04   12    0AAB  1438          BNEQ    40$                     ; BR if we need not skip <LINEFEED>
                         50   D7    0AAD  1439          DECL    R0                      ; Must pass over <LF>...
                         51   D6    0AAF  1440          INCL    R1                      ; ...since they're new line to printers
                    61   09   91    0AB1  1441 40$:     CMPB    #9,(R1)                 ; Is there a <TAB> at start of line?
                         06   12    0AB4  1442          BNEQ    50$                     ; BR if not - we can start indenting
                         50   D7    0AB6  1443          DECL    R0                      ; Must pass over the <TAB>
                         51   D6    0AB8  1444          INCL    R1                      ; More of passing over the <TAB>
                         F5   11    0ABA  1445          BRB     40$                     ; Inner loop to find multiple <TAB>s
                         50   D5    0ABC  1446 50$:     TSTL    R0                      ; If we're at the end of the string...
                         19   13    0ABE  1447          BEQL    60$                     ; ...we can exit the outer loop
                         03   BB    0AC0  1448          PUSHR   #^M<R0,R1>              ; Save desc to rest of string
              04 A1   61    50    28    0AC2  1449          MOVC3   R0,(R1),INDENT(R1)      ; Indent the rest of the string
04 BE   04   20    00 8F    00    2C    0AC7  1450          MOVC5   #0,#0,#^A/ /,#INDENT,@4(SP) ; Fill indented spaces with blanks
                         03   BA    0ACF  1451          POPR    #^M<R0,R1>              ; Restore desc to rest of string
                    51   04   C0    0AD1  1452          ADDL2   #INDENT,R1              ; Point beyond the spaces just inserted
                    6E   04   A0    0AD4  1453          ADDW2   #INDENT,(SP)            ; Count total length incl. indentation
                         C5   11    0AD7  1454          BRB     30$                     ; Loop to see if we need indent again
                                0AD9  1455 60$:
                06AE'CF   6E    B0    0AD9  1456          MOVW    (SP),LOG_RAB+RAB$W_RSZ  ; Set log rec size from indented rec
                                0ADE  1457          $PUT    RAB = LOG_RAB,-         ; Copy the record
                                0ADE  1458                  ERR = RMS_ERROR
                003B'CF   8E    B0    0AED  1459          MOVW    (SP)+,BUFFER_PTR        ; Set size of indented record...
                     003B'CF  7F    0AF2  1460          PUSHAQ  BUFFER_PTR              ; ...so we can make it...
                     038F'CF  7F    0AF6  1461          PUSHAQ  TEMP_BUFF_DESC
          00000000'GF   02    FB    0AFA  1462          CALLS   #2,G^STR$UPCASE         ; ...uniform case for sentinel checking
                     50   01    9A    0B01  1463          MOVZBL  #SS$_NORMAL,R0
                                05    0B04  1464          RSB
```

```
                                0B05  1466          .SBTTL  BUILD_INIDEV Routines - FINISH_CONTROLLER
                                0B05  1467 ;
                                0B05  1468 ; We now have to check up on the subprocess we created to see what it really
                                0B05  1469 ; did to UETINIDEV.DAT.  If the process failed or found no testable units, the
                                0B05  1470 ; DDB and UCB lines will be marked untestable.  If it found testable units,
                                0B05  1471 ; write a line to UETCONT00.DAT so the device will be tested in the Device
                                0B05  1472 ; Test phase.  Remove the end-of-file record from UETINIDEV because there are
                                0B05  1473 ; probably more devices to check.  We'll rewrite the record if we really have
                                0B05  1474 ; finished testing.  For convenient access, R6 will point to INI_RAB and R7
                                0B05  1475 ; will point to CON_RAB.  R10 and R11 are in use from BUILD_INIDEV.
                                0B05  1476 ;
                                0B05  1477 FINISH_CONTROLLER:
          56   0514'CF    DE   0B05  1478          MOVAL   INI_RAB,R6              ; Set up...
          57   05F8'CF    DE   0B0A  1479          MOVAL   CON_RAB,R7              ; ...convenient RAB pointers
            1E A6    02   90   0B0F  1480          MOVB    #RAB$C_RFA,RAB$B_RAC(R6) ; Set RFA mode
      10 A6   0764'CF    06   28   0B13  1481          MOVC3   #6,DDB_RFA,RAB$W_RFA(R6) ; Set RFA to point to the DDB line
                                0B1A  1482          $GET    RAB = (R6),-            ; Go back to the DDB record
                                0B1A  1483                  ERR = RMS_ERROR
         0047'CF    4E 8F    91   0B27  1484          CMPB    #^A/N/,BUFFER+4         ; Any testable devices?
                       05    13   0B2D  1485          BEQL    10$                     ; BR if not
              2B 01A9'CF    E8   0B2F  1486          BLBS    MBX_BUF+ACC$L_FINALSTS,30$ ; All was OK if success status
                                0B34  1487 10$:
            1E A6    00    90   0B34  1488          MOVB    #RAB$C_SEQ,RAB$B_RAC(R6) ; Reset UETINIDEV to sequential access
                                0B38  1489 20$:
         0047'CF    4E 8F    90   0B38  1490          MOVB    #^A/N/,BUFFER+4         ; Set DDB or UCB not testable
                                0B3E  1491          $UPDATE RAB = (R6)              ; Update each record
                                0B47  1492          $GET    RAB = (R6),-            ; Get a UCB record
                                0B47  1493                  ERR = RMS_ERROR
      076A'CF   10 A6    06   29   0B54  1494          CMPC3   #6,RAB$W_RFA(R6),END_RFA ; Are all records updated?
                       DB    12   0B5B  1495          BNEQ    20$                     ; Loop if not
                       58    11   0B5D  1496          BRB     40$                     ; Skip success stuff if so
                                0B5F  1497 30$:
                                0B5F  1498          $FAO_S  CTRSTR = CONT_STR,-     ; Make the UETCONT00 record
                                0B5F  1499                  OUTBUF = FAO_ALT,-
                                0B5F  1500                  OUTLEN = ALTBUF_PTR,-
                                0B5F  1501                      P1 = #TEST_DSC,-
                                0B5F  1502                      P2 = #DEVDSC
         22 A7    0145'CF    B0   0B7E  1503          MOVW    ALTBUF_PTR,RAB$W_RSZ(R7) ; Set the record size
         28 A7    014D'CF    DE   0B84  1504          MOVAL   ALTBUF,RAB$L_RBF(R7)    ; Set the record address
                                0B8A  1505          $PUT    RAB = (R7),-            ; Write the record
                                0B8A  1506                  ERR = RMS_ERROR
                  0373'CF    D6   0B97  1507          INCL    TST_CNT                 ; One more device test to run
                    1E A6    02   90   0B9B  1508          MOVB    #RAB$C_RFA,RAB$B_RAC(R6) ; Set RFA mode in UETINIDEV.DAT
      10 A6   076A'CF    06   28   0B9F  1509          MOVC3   #6,END_RFA,RAB$W_RFA(R6) ; Point to the END line in UETINIDEV
                                0BA6  1510          $FIND   RAB = (R6),-            ; Get the END line as current
                                0BA6  1511                  ERR = RMS_ERROR
                    1E A6    00   90   0BB3  1512          MOVB    #RAB$C_SEQ,RAB$B_RAC(R6) ; Reset UETINIDEV to sequential access
                                0BB7  1513 40$:
                                0BB7  1514          $TRUNCATE RAB = (R6),-          ; Eat the END OF UETINDEV.DAT line
                                0BB7  1515                    ERR = RMS_ERROR
                                0BC4  1516          RSB
```

```
                        0BC5  1518          .SBTTL  Stop a Subprocess
                        0BC5  1519 ;++
                        0BC5  1520 ; FUNCTIONAL DESCRIPTION:
                        0BC5  1521 ;       We must terminate a subprocess either because it seems to be hung or
                        0BC5  1522 ;       because UETINIT01 is terminating.  Fix things up so we can try to
                        0BC5  1523 ;       continue with the next controller in the system, if need be.
                        0BC5  1524 ;
                        0BC5  1525 ; CALLING SEQUENCE:
                        0BC5  1526 ;       Called at AST level
                        0BC5  1527 ;
                        0BC5  1528 ; INPUT PARAMETERS:
                        0BC5  1529 ;       04(AP) - PID of the subprocess we must terminate
                        0BC5  1530 ;
                        0BC5  1531 ; IMPLICIT INPUTS:
                        0BC5  1532 ;       DEVDSC has the name of the controller we wanted to test.
                        0BC5  1533 ;       TEST_DSC has the name of the image to test the controller.
                        0BC5  1534 ;
                        0BC5  1535 ; OUTPUT PARAMETERS:
                        0BC5  1536 ;       NONE
                        0BC5  1537 ;
                        0BC5  1538 ; IMPLICIT OUTPUTS:
                        0BC5  1539 ;       NONE
                        0BC5  1540 ;
                        0BC5  1541 ; COMPLETION CODES:
                        0BC5  1542 ;       Value from ERROR_MSG.
                        0BC5  1543 ;
                        0BC5  1544 ; SIDE EFFECTS:
                        0BC5  1545 ;       Subprocess is deleted, if it was created.
                        0BC5  1546 ;       Error message given.
                        0BC5  1547 ;--
                        0BC5  1548
                        0BC5  1549 KILL_SUBPROCESS:
                  0FFC  0BC5  1550          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0BC7  1551
          04 AC   D5    0BC7  1552          TSTL    04(AP)                  ; Was the subprocess created?
             57   13    0BCA  1553          BEQL    10$                     ; BR if not
                        0BCC  1554          $DELPRC_S PIDADR = 04(AP)       ; Ensure that the subprocess dies
                        0BD8  1555          $FAO_S  CTRSTR = PROCESS_STOP_MSG,- ; Indicate what happened
                        0BD8  1556                  OUTBUF = FAO_BUF,-
                        0BD8  1557                  OUTLEN = BUFFER_PTR,-
                        0BD8  1558                  P1     = #TEST_DSC,-
                        0BD8  1559                  P2     = #DEVDSC,-
                        0BD8  1560                  P3     = #0
       003B'CF  DF      0BF9  1561          PUSHAL  BUFFER_PTR
 000F0001  8F   DD      0BFD  1562          PUSHL   #^XF0001
 00741132  8F   DD      0C03  1563          PUSHL   #UETP$_TEXT!STS$K_ERROR
0E61'CF    03   FB      0C09  1564          CALLS   #3,ERROR_MSG
                        0C0E  1565          $SETIMR_S DAYTIM = ONEMIN,-     ; Another timer will catch cases...
                        0C0E  1566                    ASTADR = PASS_OVER_SUBPROCESS,-
                        0C0E  1567                    REQIDT = PIDADR       ; ...where $DELPRC isn't enough
                        0C23  1568 10$:
                     04 0C23  1569          RET
```

```
                     0C24  1571          .SBTTL  Pass Over a Subprocess If $DELPRC Fails
                     0C24  1572 ;++
                     0C24  1573 ; FUNCTIONAL DESCRIPTION:
                     0C24  1574 ;       Image rundown has not yet terminated a process we believe to be hung
                     0C24  1575 ;       (KILL_SUBPROCESS).  We can do no more, so prevent further waiting for
                     0C24  1576 ;       the process' termination mailbox.
                     0C24  1577 ;
                     0C24  1578 ; CALLING SEQUENCE:
                     0C24  1579 ;       Called at AST level
                     0C24  1580 ;
                     0C24  1581 ; INPUT PARAMETERS:
                     0C24  1582 ;       04(AP) - PID of the subprocess we must terminate
                     0C24  1583 ;
                     0C24  1584 ; IMPLICIT INPUTS:
                     0C24  1585 ;       DEVDSC has the name of the controller we wanted to test.
                     0C24  1586 ;       TEST_DSC has the name of the image to test the controller.
                     0C24  1587 ;
                     0C24  1588 ; OUTPUT PARAMETERS:
                     0C24  1589 ;       NONE
                     0C24  1590 ;
                     0C24  1591 ; IMPLICIT OUTPUTS:
                     0C24  1592 ;       NONE
                     0C24  1593 ;
                     0C24  1594 ; COMPLETION CODES:
                     0C24  1595 ;       Value from ERROR_MSG.
                     0C24  1596 ;
                     0C24  1597 ; SIDE EFFECTS:
                     0C24  1598 ;       $QIOW is cancelled for termination mailbox.
                     0C24  1599 ;       Error message given.
                     0C24  1600 ;--
                     0C24  1601
                     0C24  1602 PASS_OVER_SUBPROCESS:
                OFFC 0C24  1603          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     0C26  1604
                     0C26  1605          $CANCEL_S CHAN = MBXCHN          ; Stop waiting for termination mailbox
                     0C32  1606          $FAO_S  CTRSTR = PASS_OVER_MSG,- ; Indicate what happened
                     0C32  1607                  OUTBUF = FAO_BUF,-
                     0C32  1608                  OUTLEN = BUFFER_PTR,-
                     0C32  1609                  P1     = #TEST_DSC,-
                     0C32  1610                  P2     = #DEVDSC,-
                     0C32  1611                  P3     = #0
         003B'CF     DF  0C53  1612          PUSHAL  BUFFER_PTR
     000F0001 8F     DD  0C57  1613          PUSHL   #^XF0001
     00741132 8F     DD  0C5D  1614          PUSHL   #UETP$_TEXT!STS$K_ERROR
    0E61'CF    03    FB  0C63  1615          CALLS   #3,ERROR_MSG
                     04  0C68  1616          RET
```

```
                     0C69  1618          .SBTTL  Check Whether a Message Needs $FAO Arguments
                     0C69  1619 ;++
                     0C69  1620 ; FUNCTIONAL DESCRIPTION:
                     0C69  1621 ;       To enable more useful error messages, we'd like to print out the
                     0C69  1622 ;       message associated with failures as well as the message we provide
                     0C69  1623 ;       ourself.  Some of those messages have $FAO arguments, the values
                     0C69  1624 ;       for which are lost.  Provide a $PUTMSG length of zero and a set of
                     0C69  1625 ;       appropriate flag bits so that messages which have no arguments get
                     0C69  1626 ;       printed in their entirety but those that do have arguments omit the
                     0C69  1627 ;       text part.
                     0C69  1628 ;
                     0C69  1629 ; CALLING SEQUENCE:
                     0C69  1630 ;       MOVL    status,R2
                     0C69  1631 ;       BSBW    FAO_CHECK
                     0C69  1632 ;
                     0C69  1633 ; IMPLICIT INPUTS:
                     0C69  1634 ;       R2 has the status to check.
                     0C69  1635 ;
                     0C69  1636 ; IMPLICIT OUTPUTS:
                     0C69  1637 ;       R2 returns a flag-length longword suitable for $PUTMSG or zero (to
                     0C69  1638 ;               indicate a null message).
                     0C69  1639 ;
                     0C69  1640 ; SIDE EFFECTS:
                     0C69  1641 ;       ALTBUF, ALTBUF_PTR and MSG_BLOCK get written into.
                     0C69  1642 ;--
                     0C69  1643
                     0C69  1644 FAO_CHECK:
                     0C69  1645          $GETMSG_S MSGID = R2,-          ; Check if we need flags to suppress...
                     0C69  1646                    MSGLEN = ALTBUF_PTR,-
                     0C69  1647                    BUFADR = FAO_ALT,-
                     0C69  1648                     FLAGS = #0,-
                     0C69  1649                    OUTADR = MSG_BLOCK
          0378'CF    95  0C80  1650          TSTB    MSG_BLOCK+1             ; ...any $FAO args in message text
                03   12  0C84  1651          BNEQ    10$                     ; BR if there are $FAO args
                52   D4  0C86  1652          CLRL    R2                      ; Set flags arg to look like null msg
                     05  0C88  1653          RSB
                     0C89  1654 10$:
52   000E0000 8F    D0  0C89  1655          MOVL    #^XE0000,R2             ; Suppress the text of the message
                     05  0C90  1656          RSB
```

```
0C91  1658          .SBTTL  System Service Exception Handler
0C91  1659 ;++
0C91  1660 ; FUNCTIONAL DESCRIPTION:
0C91  1661 ;       This routine is executed if a system service or RMS error occurs or
0C91  1662 ;       if a LIB$SIGNAL system service is used to output a message.
0C91  1663 ;
0C91  1664 ; CALLING SEQUENCE:
0C91  1665 ;       Entered via an exception from the system
0C91  1666 ;
0C91  1667 ; INPUT PARAMETERS:
0C91  1668 ;       ERROR_COUNT    = previous cumulative error count
0C91  1669 ;
0C91  1670 ;          AP ---->      +---------------+
0C91  1671 ;                        |       2       |
0C91  1672 ;                        |---------------|
0C91  1673 ;                        | SIGNL ARY PNT |
0C91  1674 ;                        |---------------|
0C91  1675 ;                        | MECH  ARY PNT |  ---------
0C91  1676 ;                        |---------------|         ^
0C91  1677 ;                        |       4       |         :
0C91  1678 ;                        |---------------|         :
0C91  1679 ;                        | ESTABLISH FP  |         :
0C91  1680 ;                        |---------------|         :
0C91  1681 ;                        |     DEPTH     |  Mechanism Array
0C91  1682 ;                        |---------------|         :
0C91  1683 ;                        |       R0      |         :
0C91  1684 ;                        |---------------|         v
0C91  1685 ;                        |       R1      |  ---------
0C91  1686 ;                        |---------------|         ^
0C91  1687 ;                        |       N       |         :
0C91  1688 ;                        |---------------|         :
0C91  1689 ;                        | CONDITION NAME|         :
0C91  1690 ;                        |---------------|         :
0C91  1691 ;                        | N-3 ADDITIONAL|    Signal Array
0C91  1692 ;                        | LONG WORD ARGS|         :
0C91  1693 ;                        |---------------|         :
0C91  1694 ;                        |       PC      |         :
0C91  1695 ;                        |---------------|         :
0C91  1696 ;                        |       PSL     |         v
0C91  1697 ;                        +---------------+  ---------
0C91  1698 ; IMPLICIT INPUTS:
0C91  1699 ;       NONE
0C91  1700 ; OUTPUT PARAMETERS:
0C91  1701 ;       NONE
0C91  1702 ;
0C91  1703 ; IMPLICIT OUTPUTS:
0C91  1704 ;       NONE
0C91  1705 ;
0C91  1706 ; COMPLETION CODES:
0C91  1707 ;       SS$_NORMAL if it's a UETP condition or RMS error.
0C91  1708 ;       Error status from exception, otherwise.
0C91  1709 ;
0C91  1710 ; SIDE EFFECTS:
0C91  1711 ;       May branch to ERROR_EXIT.
0C91  1712 ;       May print a message.
0C91  1713 ;--
0C91  1714
```

```
                        0C91  1715 SSERROR:
                   0FFC 0C91  1716          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0C93  1717
                        0C93  1718          $SETAST_S ENBFLG = #0               ; Disable AST delivery
                01      DD    0C9C  1719          PUSHL   #1                      ; Assume ASTs were enabled
           50   09      D1    0C9E  1720          CMPL    S^#SS$_WASSET,R0        ; Were ASTs enabled?
                02      13    0CA1  1721          BEQL    10$                     ; BR if they were
                6E      D4    0CA3  1722          CLRL    (SP)                    ; Set ASTs to remain disabled
                        0CA5  1723 10$:
                        0CA5  1724          $SETSFM_S ENBFLG = #0               ; Disable SS failure mode
                01      DD    0CAE  1725          PUSHL   #1                      ; Assume SS failure mode was enabled
           50   09      D1    0CB0  1726          CMPL    S^#SS$_WASSET,R0        ; Was SS failure mode enabled?
                02      13    0CB3  1727          BEQL    20$                     ; BR if it was
                6E      D4    0CB5  1728          CLRL    (SP)                    ; Set SS failure mode to remain off
                        0CB7  1729 20$:
      56   04 AC      D0    0CB7  1730          MOVL    CHF$L_SIGARGLST(AP),R6  ; Get the signal array pointer
      59   04 A6      7D    0CBB  1731          MOVQ    CHF$L_SIG_NAME(R6),R9   ; Get NAME in R9 and ARG1 in R10
                10      ED    0CBF  1732          CMPZV   #STS$V_FAC_NO,-         ; Is this a message from LIB$SIGNAL?
                0C            0CC1  1733                  #STS$S_FAC_NO,-
00000074 8F             59    0CC2  1734                  R9,#UETP$_FACILITY
                16      12    0CC8  1735          BNEQ    30$                     ; BR if this is not a UETP exception
           66   02      C2    0CCA  1736          SUBL2   #2,CHF$L_SIG_ARGS(R6)   ; Drop the PC and PSL
                        0CCD  1737          $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6),- ; Print the message
                        0CCD  1738                    ACTRTN = ACTRTN
                21      11    0CDE  1739          BRB     40$                     ; Restore ASTs and SS fail mode
                        0CE0  1740 30$:
59   0000045C 8F       D1    0CE0  1741          CMPL    #SS$_SSFAIL,R9          ; RMS failures are SysSvc failures
                32      12    0CE7  1742          BNEQ    50$                     ; BR if this can't be an RMS failure
                10      ED    0CE9  1743          CMPZV   #STS$V_FAC_NO,-         ; Is it an RMS failure?
                0C            0CEB  1744                  #STS$S_FAC_NO,-
           01   5A            0CEC  1745                  R10,#RMS$_FACILITY
                2B      12    0CEE  1746          BNEQ    50$                     ; BR if not
5A   F0000000 8F       CA    0CF0  1747          BICL2   #^XF0000000,R10         ; Strip control bits from status code
      08 A6    04      39    0CF7  1748          MATCHC  #4,CHF$L_SIG_ARG1(R6),- ; Is it an RMS failure for which...
                14            0CFB  1749                  #NRAT_LENGTH,-
          0074'CF             0CFC  1750                  NO_RMS_AST_TABLE        ; ...no AST can be delivered?
                1A      13    0CFF  1751          BEQL    50$                     ; BR if so - must give error here
                        0D01  1752 40$:
                01      BA    0D01  1753          POPR    #^M<R0>                 ; Restore SS failure mode...
                        0D03  1754          $SETSFM_S ENBFLG = R0               ; ...
                01      BA    0D0C  1755          POPR    #^M<R0>                 ; Restore AST enable...
                        0D0E  1756          $SETAST_S ENBFLG = R0               ; ...
           50   01      D0    0D17  1757          MOVL    S^#SS$_NORMAL,R0        ; Supply a standard status for exit
                04      04    0D1A  1758          RET                             ; Resume processing (or goto RMS_ERROR)
                        0D1B  1759 50$:
     0357'CF    59      D0    0D1B  1760          MOVL    R9,STATUS               ; Save the status
                58      D4    0D20  1761          CLRL    R8                      ; Assume for now it's not SS failure
59   0000045C 8F       D1    0D22  1762          CMPL    #SS$_SSFAIL,R9          ; But is it a System Service failure?
                38      12    0D29  1763          BNEQ    70$                     ; BR if not - no special case message
                        0D2B  1764          $GETMSG_S MSGID = R10,-             ; Get SS failure code associated text
                        0D2B  1765                    MSGLEN = BUFFER_PTR,-
                        0D2B  1766                    BUFADR = FAO_BUF,-
                        0D2B  1767                    FLAGS  = #14,-
                        0D2B  1768                    OUTADR = MSG_BLOCK
          0378'CF   95      0D42  1769          TSTB    MSG_BLOCK+1             ; Get FAO arg count for SS failure code
                16      13    0D46  1770          BEQL    60$                     ; Don't use $GETMSG if no $FAO args...
          003B'CF   DF      0D48  1771          PUSHAL  BUFFER_PTR              ; ...else build up...
```

```
                01   DD  0D4C  1772          PUSHL   #1                              ; ...a message describing...
        00741130 8F   DD  0D4E  1773          PUSHL   #UETP$_TEXT                     ; ...why the System Service failed
              00 5A   F0  0D54  1774          INSV    R10,#STS$V_SEVERITY,-           ; Give the message...
              6E 03           0D57  1775                  #STS$S_SEVERITY,(SP)        ; ...the correct severity code
              58 03   D0  0D59  1776          MOVL    #3,R8                           ; Count the number of args we pushed
                 05   11  0D5C  1777          BRB     70$
                          0D5E  1778 60$:
                 5A   DD  0D5E  1779          PUSHL   R10                             ; Save SS failure code
              58 01   D0  0D60  1780          MOVL    #1,R8                           ; Count the number of args we pushed
                          0D63  1781 70$:
           57 66 04   C5  0D63  1782          MULL3   #4,CHF$L_SIG_ARGS(R6),R7 ; Convert longwords to bytes
              5E 57   C2  0D67  1783          SUBL2   R7,SP                           ; Save the current signal array...
        6E 04 A6 57   28  0D6A  1784          MOVC3   R7,CHF$L_SIG_NAME(R6),(SP) ; ...on the stack
           7E 66 58   C1  0D6F  1785          ADDL3   R8,CHF$L_SIG_ARGS(R6),-(SP) ; Push the current arg count
                 0120 31  0D73  1786          BRW     ERROR_EXIT
```

```
                     0D76  1788          .SBTTL  $PUTMSG Action Routine
                     0D76  1789 ;++
                     0D76  1790 ; FUNCTIONAL DESCRIPTION:
                     0D76  1791 ;       This routine is called as the action routine from $PUTMSG to copy
                     0D76  1792 ;       messages to our own log file as well as have them written to SYS$OUTPUT.
                     0D76  1793 ;
                     0D76  1794 ; CALLING SEQUENCE:
                     0D76  1795 ;       Called by $PUTMSG system service if present as the ACTRTN parameter.
                     0D76  1796 ;
                     0D76  1797 ; INPUT PARAMETERS:
                     0D76  1798 ;       04(AP) - Descriptor for an ASCID string with the message
                     0D76  1799 ;
                     0D76  1800 ; IMPLICIT INPUTS:
                     0D76  1801 ;       NONE
                     0D76  1802 ;
                     0D76  1803 ; OUTPUT PARAMETERS:
                     0D76  1804 ;       NONE
                     0D76  1805 ;
                     0D76  1806 ; IMPLICIT OUTPUTS:
                     0D76  1807 ;       NONE
                     0D76  1808 ;
                     0D76  1809 ; COMPLETION CODES:
                     0D76  1810 ;       Always successful if it returns.
                     0D76  1811 ;
                     0D76  1812 ; SIDE EFFECTS:
                     0D76  1813 ;       Message copied to log file.
                     0D76  1814 ;       Program may exit, if RMS encounters an error.
                     0D76  1815 ;
                     0D76  1816 ;--
                     0D76  1817
                     0D76  1818 ACTRTN:
                0004 0D76  1819          .WORD   ^M<R2>
                     0D78  1820
   7E    06AE'CF    B0  0D78  1821          MOVW    LOG_RAB+RAB$W_RSZ,-(SP) ; Save vital info in case we're asynch
         06B4'CF    DD  0D7D  1822          PUSHL   LOG_RAB+RAB$L_RBF
     52    04 AC    D0  0D81  1823          MOVL    4(AP),R2                ; Get the message descriptor address
   06AE'CF    62    3C  0D85  1824          MOVZWL  (R2),LOG_RAB+RAB$W_RSZ  ; Get the message size
06B4'CF    04 A2    D0  0D8A  1825          MOVL    4(R2),LOG_RAB+RAB$L_RBF ; Set the message address
                     0D90  1826          $PUT    RAB = LOG_RAB,-
                     0D90  1827                  ERR = RMS_ERROR         ; Write the log file
         50    01    D0  0D9F  1828          MOVL    S^#SS$_NORMAL,R0        ; Set the return status code
   06B4'CF    8E    D0  0DA2  1829          MOVL    (SP)+,LOG_RAB+RAB$L_RBF ; Restore vital info
   06AE'CF    8E    B0  0DA7  1830          MOVW    (SP)+,LOG_RAB+RAB$W_RSZ
                    04  0DAC  1831          RET
```

```
                     ODAD  1833          .SBTTL  RMS Error Handler
                     ODAD  1834 ;++
                     ODAD  1835 ; FUNCTIONAL DESCRIPTION:
                     ODAD  1836 ;       This routine handles error returns from RMS calls.
                     ODAD  1837 ;
                     ODAD  1838 ; CALLING SEQUENCE:
                     ODAD  1839 ;       Called by RMS when a file processing error is found.  May also be
                     ODAD  1840 ;       called inline to process an RMS error.
                     ODAD  1841 ;
                     ODAD  1842 ; INPUT PARAMETERS:
                     ODAD  1843 ;       The FAB or RAB associated with the RMS call.
                     ODAD  1844 ;
                     ODAD  1845 ; IMPLICIT INPUTS:
                     ODAD  1846 ;       NONE
                     ODAD  1847 ;
                     ODAD  1848 ; OUTPUT PARAMETERS:
                     ODAD  1849 ;       NONE
                     ODAD  1850 ;
                     ODAD  1851 ; IMPLICIT OUTPUTS:
                     ODAD  1852 ;       Error message
                     ODAD  1853 ;
                     ODAD  1854 ; COMPLETION CODES:
                     ODAD  1855 ;       NONE
                     ODAD  1856 ;
                     ODAD  1857 ; SIDE EFFECTS:
                     ODAD  1858 ;       Program may exit, depending on severity of the error.
                     ODAD  1859 ;
                     ODAD  1860 ;--
                     ODAD  1861
                     ODAD  1862 RMS_ERROR:
                0FFC ODAD  1863          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                     ODAF  1864
   6D   FEDE CF   DE ODAF  1865          MOVAL   SSERROR,(FP)                ; Set up our exception handler
     56   04 AC   D0 ODB4  1866          MOVL    4(AP),R6                    ; See whether we're dealing with...
        66   03   91 ODB8  1867          CMPB    #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
             16   12 ODBB  1868          BNEQ    10$                         ; BR if it's a RAB
   57   03B9'CF   DE ODBD  1869          MOVAL   FILE,R7                     ; FAB-specific code:  text string...
        58   56   D0 ODC2  1870          MOVL    R6,R8                       ; ...address of FAB...
          0C A6   DD ODC5  1871          PUSHL   FAB$L_STV(R6)               ; ...STV field for error...
          08 A6   DD ODC8  1872          PUSHL   FAB$L_STS(R6)               ; ...STS field for error...
0357'CF   08 A6   D0 ODCB  1873          MOVL    FAB$L_STS(R6),STATUS        ; ...and save the error code
             15   11 ODD1  1874          BRB     RMS_COMMON                  ; FAB and RAB share other code
                     ODD3  1875 10$:
   57   03C5'CF   DE ODD3  1876          MOVAL   RECORD,R7                   ; RAB-specific code:  text string...
     58   3C A6   D0 ODD8  1877          MOVL    RAB$L_FAB(R6),R8            ; ...address of associated FAB...
          0C A6   DD ODDC  1878          PUSHL   RAB$L_STV(R6)               ; ...STV field for error...
          08 A6   DD ODDF  1879          PUSHL   RAB$L_STS(R6)               ; ...STS field for error...
0357'CF   08 A6   D0 ODE2  1880          MOVL    RAB$L_STS(R6),STATUS        ; ...and save the error code
                     ODE8  1881 RMS_COMMON:
     5A   34 A8   9A ODE8  1882          MOVZBL  FAB$B_FNS(R8),R10           ; Get the file name size
                     ODEC  1883          $FAO_S  CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
                     ODEC  1884                  OUTLEN = BUFFER_PTR,-
                     ODEC  1885                  OUTBUF = FAO_BUF,-
                     ODEC  1886                  P1     = R7,-
                     ODEC  1887                  P2     = R10,-
                     ODEC  1888                  P3     = FAB$L_FNA(R8)
        003B'CF   DF ODE6  1889          PUSHAL  BUFFER_PTR                  ; ...and arguments for ERROR_EXIT...
```

```
                    01   DD  0E0A  1890          PUSHL   #1                              ; ...
            00741130 8F   DD  0E0C  1891          PUSHL   #UETP$_TEXT                     ; ...
                    00   EF  0E12  1892          EXTZV   #STS$V_SEVERITY,-
                    03       0E14  1893                  #STS$S_SEVERITY,-
          59  0357'CF       0E15  1894                  STATUS,R9                       ; ...get the severity code...
              6E    59   88  0E19  1895          BISB2   R9,(SP)                         ; ...and add it into the signal name
                    05   DD  0E1C  1896          PUSHL   #5                              ; Current arg count
                  0075   31  0E1E  1897          BRW     ERROR_EXIT
```

```
                     0E21  1899          .SBTTL  CTRL/C Handler
                     0E21  1900 ;++
                     0E21  1901 ; FUNCTIONAL DESCRIPTION:
                     0E21  1902 ;       This routine handles CTRL/C AST's
                     0E21  1903 ;
                     0E21  1904 ; CALLING SEQUENCE:
                     0E21  1905 ;       Called via AST
                     0E21  1906 ;
                     0E21  1907 ; INPUT PARAMETERS:
                     0E21  1908 ;       NONE
                     0E21  1909 ;
                     0E21  1910 ; IMPLICIT INPUTS:
                     0E21  1911 ;       NONE
                     0E21  1912 ;
                     0E21  1913 ; OUTPUT PARAMETERS:
                     0E21  1914 ;       NONE
                     0E21  1915 ;
                     0E21  1916 ; IMPLICIT OUTPUTS:
                     0E21  1917 ;       NONE
                     0E21  1918 ;
                     0E21  1919 ; COMPLETION CODES:
                     0E21  1920 ;       NONE
                     0E21  1921 ;
                     0E21  1922 ; SIDE EFFECTS:
                     0E21  1923 ;       NONE
                     0E21  1924 ;
                     0E21  1925 ;--
                     0E21  1926
                     0E21  1927 CCASTHAND:
                0FFC 0E21  1928          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,P9,R10,R11> ; Entry mask
                     0E23  1929
          02BA'CF    DD   0E23  1930          PUSHL   PIDADR                      ; Clean up any subprocess...
      FD99 CF    01   FB   0E27  1931          CALLS   #1,KILL_SUBPROCESS          ; ...
          0398'CF    DF   0E2C  1932          PUSHAL  CNTRLCMSG                   ; Set message pointer
             01      DD   0E30  1933          PUSHL   #1                          ; Set arg count
       00741130 8F   DD   0E32  1934          PUSHL   #UETP$_TEXT!STS$K_WARNING ; Set signal name
             00      DD   0E38  1935          PUSHL   #0                          ; Indicate an abnormal termination
          000F'CF    DF   0E3A  1936          PUSHAL  TEST_NAME                   ; ...
             02      DD   0E3E  1937          PUSHL   #2                          ; ...
       007410E0 8F   DD   0E40  1938          PUSHL   #UETP$_ABENDD!STS$K_WARNING ; ...
  00000000'GF    07   FB   0E46  1939          CALLS   #7,G^LIB$SIGNAL             ; Output the message
                      D0   0E4D  1940          MOVL    #<STS$M_INHIB_MSG!-         ; Set the exit status
                           0E4E  1941                  SS$_CONTROLC-
                           0E4E  1942                  STS$K_SUCCESS+STS$K_WARNING>,-
0357'CF    10000650 8F        0E4E  1943                  STATUS
                           0E56  1944          $EXIT_S STATUS                      ; Terminate program cleanly
```

```
                                0E61  1946          .SBTTL  Error Message
                                0E61  1947 ;++
                                0E61  1948 ; FUNCTIONAL DESCRIPTION:
                                0E61  1949 ;       Print an error message preceded by the UETP error box message.
                                0E61  1950 ;
                                0E61  1951 ; CALLING SEQUENCE:
                                0E61  1952 ;       Form a LIB$SIGNAL set of message except for the error box message
                                0E61  1953 ;       CALLS   count-of-messages,ERROR_MSG
                                0E61  1954 ;
                                0E61  1955 ; INPUT PARAMETERS:
                                0E61  1956 ;       List of LIB$SIGNAL arguments to which more may be added.  There is no
                                0E61  1957 ;               limit on the number or type other than what LIB$SIGNAL imposes.
                                0E61  1958 ;
                                0E61  1959 ; IMPLICIT INPUTS:
                                0E61  1960 ;       NONE
                                0E61  1961 ;
                                0E61  1962 ; OUTPUT PARAMETERS:
                                0E61  1963 ;       NONE
                                0E61  1964 ;
                                0E61  1965 ; IMPLICIT OUTPUTS:
                                0E61  1966 ;       NONE
                                0E61  1967 ;
                                0E61  1968 ; COMPLETION CODES:
                                0E61  1969 ;       Value from LIB$SIGNAL.
                                0E61  1970 ;
                                0E61  1971 ; SIDE EFFECTS:
                                0E61  1972 ;       Message from LIB$SIGNAL.
                                0E61  1973 ;--
                                0E61  1974
                                0E61  1975 ERROR_MSG:
                         0FFC   0E61  1976          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                0E63  1977
                   50    6C    04   C5  0E63  1978          MULL3   #4,0(AP),R0             ; Figure bytes already used for msgs
                         5E    50   C2  0E67  1979          SUBL2   R0,SP                   ; Reserve stack space for them
                6E   04  AC    50   28  0E6A  1980          MOVC3   R0,4(AP),(SP)           ; Copy start of LIB$SIGNAL args
                         0353'CF    D6  0E6F  1981          INCL    ERROR_COUNT             ; Keep a count of recoverable errors
                         0353'CF    DD  0E73  1982          PUSHL   ERROR_COUNT             ; Form the rest of LIB$SIGNAL args
                         000F'CF    DF  0E77  1983          PUSHAL  TEST_NAME
                     00010002  8F    DD  0E7B  1984          PUSHL   #^X10002
                     00748022  8F    DD  0E81  1985          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR
                   50    6C    04   C1  0E87  1986          ADDL3   #4,0(AP),R0             ; Count total number of LIB$SIGNAL args
            00000000'GF    50   FB  0E8B  1987          CALLS   R0,G^LIB$SIGNAL
                         50    01   D0  0E92  1988          MOVL    #SS$_NORMAL,R0
                                04   04  0E95  1989          RET                             ; Continue with recovery
```

```
                     0E96  1991          .SBTTL  Error Exit
                     0E96  1992 ;++
                     0E96  1993 ; FUNCTIONAL DESCRIPTION:
                     0E96  1994 ;       This routine prints an error message and exits.
                     0E96  1995 ;
                     0E96  1996 ; CALLING SEQUENCE:
                     0E96  1997 ;       MOVx  error status value,STATUS
                     0E96  1998 ;       PUSHx error specific information on the stack
                     0E96  1999 ;       PUSHL current argument count
                     0E96  2000 ;       BRW   ERROR_EXIT
                     0E96  2001 ;
                     0E96  2002 ; INPUT PARAMETERS:
                     0E96  2003 ;       Arguments to LIB$SIGNAL, as above
                     0E96  2004 ;
                     0E96  2005 ; IMPLICIT INPUTS:
                     0E96  2006 ;       NONE
                     0E96  2007 ;
                     0E96  2008 ; OUTPUT PARAMETERS:
                     0E96  2009 ;       Message to SYS$OUTPUT and SYS$ERROR
                     0E96  2010 ;
                     0E96  2011 ; IMPLICIT OUTPUTS:
                     0E96  2012 ;       Program exit
                     0E96  2013 ;
                     0E96  2014 ; COMPLETION CODES:
                     0E96  2015 ;       NONE
                     0E96  2016 ;
                     0E96  2017 ; SIDE EFFECTS:
                     0E96  2018 ;       NONE
                     0E96  2019 ;
                     0E96  2020 ;--
                     0E96  2021
                     0E96  2022 ERROR_EXIT:
                     0E96  2023
                     0E96  2024          $SETAST_S ENBFLG = #0                   ; ASTs can play havoc with messages
   15 0021'CF    01    E0  0E9F  2025          BBS     #BEGIN_MSGV,FLAG,10$    ; BR if "begin" msg already printed
                 7E    D4  0EA5  2026          CLRL    -(SP)                   ; Set the time stamp flag
         000F'CF       DF  0EA7  2027          PUSHAL  TEST_NAME               ; Set the test name
                 02    DD  0EAB  2028          PUSHL   #2                      ; Push the argument count
      00741039 8F      DD  0EAD  2029          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
00000000'GF    04      FB  0EB3  2030          CALLS   #4,G^LIB$SIGNAL         ; Print the startup message
                             0EBA  2031 10$:
  038B'CF      08    8E    C1  0EBA  2032          ADDL3   (SP)+,#8,ARG_COUNT      ; Get total # args, pop partial count
         0353'CF       D6  0EC0  2033          INCL    ERROR_COUNT             ; Keep running error count
                 00    DD  0EC4  2034          PUSHL   #0                      ; Push the time parameter
         000F'CF       DF  0EC6  2035          PUSHAL  TEST_NAME               ; Push test name...
      000F0002 8F      DD  0ECA  2036          PUSHL   #^XF0002                ; ...arg count...
      007410E2 8F      DD  0ED0  2037          PUSHL   #UETP$_ABENDD!STS$K_ERROR ; ...and signal name
         0353'CF       DD  0ED6  2038          PUSHL   ERROR_COUNT             ; Finish off arg list...
         000F'CF       DF  0EDA  2039          PUSHAL  TEST_NAME               ; ...
      00010002 8F      DD  0EDE  2040          PUSHL   #^X10002                ; ...
      00748022 8F      DD  0EE4  2041          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
00000000'GF    038B'CF   FB  0EEA  2042          CALLS   ARG_COUNT,G^LIB$SIGNAL  ; Truly bitch
                             0EF3  2043
         0357'CF       D5  0EF3  2044          TSTL    STATUS                  ; Did we exit with an error code?
                 09    12  0EF7  2045          BNEQ    20$                     ; BR if we did
      007410E2 8F      D0  0EF9  2046          MOVL    #UETP$_ABENDD!STS$K_ERROR,- ; Supply a generic one otherwise
         0357'CF             0EFF  2047                  STATUS
```

```
                     0F02  2048 20$:
0357'CF    10000000 8F   C8  0F02  2049          BISL    #STS$M_INHIB_MSG,STATUS ; Don't print messages twice!
                     0F0B  2050          $EXIT_S STATUS                          ; Exit in error
```

```
                        0F16  2052          .SBTTL  Exit Handler
                        0F16  2053 ;++
                        0F16  2054 ; FUNCTIONAL DESCRIPTION:
                        0F16  2055 ;       This routine handles cleanup on exits.
                        0F16  2056 ;
                        0F16  2057 ; CALLING SEQUENCE:
                        0F16  2058 ;       Invoked automatically by $EXIT System Service.
                        0F16  2059 ;
                        0F16  2060 ; INPUT PARAMETERS:
                        0F16  2061 ;       Location STATUS contains the exit status, FLAG has synchronizing bits.
                        0F16  2062 ;
                        0F16  2063 ; IMPLICIT INPUTS:
                        0F16  2064 ;       NONE
                        0F16  2065 ;
                        0F16  2066 ; OUTPUT PARAMETERS:
                        0F16  2067 ;       NONE
                        0F16  2068 ;
                        0F16  2069 ; IMPLICIT OUTPUTS:
                        0F16  2070 ;       Various files are de-accessed, the process name is reset, and any
                        0F16  2071 ;       necessary synchronization with UETPDEV01 is carried out.
                        0F16  2072 ;
                        0F16  2073 ; COMPLETION CODES:
                        0F16  2074 ;       NONE
                        0F16  2075 ;
                        0F16  2076 ; SIDE EFFECTS:
                        0F16  2077 ;       NONE
                        0F16  2078 ;
                        0F16  2079 ;--
                        0F16  2080
                        0F16  2081 EXIT_HANDLER:
                  0FFC  0F16  2082          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0F18  2083
                        0F18  2084          $SETSFM_S ENBFLG = #0                       ; Turn off System Service failure mode
                        0F21  2085          $SETAST_S ENBFLG = #0                       ; We're finished - no more ASTs
                        0F2A  2086          $DELLOG_S LOGNAM = MODE,-
                        0F2A  2087                    TBLFLG = #1                       ; Clean up the logical names
                     00 DD  0F39  2088          PUSHL   #0                          ; Set the time flag
               000F'CF  DF  0F3B  2089          PUSHAL  TEST_NAME                   ; Push the test name
                     02 DD  0F3F  2090          PUSHL   #2                          ; Push arg count
            00741081  8F DD  0F41  2091          PUSHL   #UETP$_ENDEDD!STS$K_SUCCESS ; Push signal name
00000000'GF  04  FB  0F47  2092          CALLS   #4,G^LIB$SIGNAL             ; Output the message
                        0F4E  2093          $SETPRN_S PRCNAM = ACNT_NAME        ; Reset the process name
                     04 0F59  2094          RET                                 ; That's all folks!
                        0F5A  2095
                        0F5A  2096          .END    UETINIT01
```

```
$$.TAB                  = 00000720 R      03
$$.TABEND               = 00000764 R      03
$$.TMP                  = 00000000
$$.TMP1                 = 00000002
$$.TMP2                 = 000000CF
$$.TMPX                 = 00000027 R      04
$$.TMPX1                = 0000000D
$$T1                    = 00000000
$$T2                    = 00000006
ACC$L_FINALSTS          = 00000004
ACNT_NAME                 00000000 R      02
ACTRTN                    00000D76 R      05
ALTBUF                    0000014D R      03
ALTBUF_PTR                00000145 R      03
APPEND_UNITS              000005CE R      05
ARG_COUNT                 0000038B R      03
ASTCM                     00000005 R      03
BAD_PO_LIST               00000458 R      02
BEGIN_MSGM              = 00000002
BEGIN_MSGV              = 00000001
BIOLM                     0000000A R      03
BLANK_LINE                000001AB R      02
BLANK_LINE_PTR            0000018B R      02
BUFFER                    00000043 R      03
BUFFER_PTR                0000003B R      03
BUILD_INIDEV              0000027C R      05
CCASTRAND                 00000E21 R      05
CHF$L_SIGARGLST         = 00000004
CHF$L_SIG_ARG1          = 00000008
CHF$L_SIG_ARGS          = 00000000
CHF$L_SIG_NAME          = 00000004
CLSIODB_ARGLST            000000A4 R      02
CLSIODB_FAIL              00000418 R      02
CLSPTR                    000004A1 R      03
CNTRLCMSG                 00000398 R      02
COMMAND_ITMLST            00000088 R      02
CONTOO_OVERHEAD           00000130 R      02
CONT_DESC                 00000260 R      02
CONT_STR                  000001B7 R      02
CON_FAB                   000005A8 R      03
CON_RAB                   000005F8 R      03
COPY_LOG_FILE             0000092B R      05
COPY_RECORD               00000A6E R      05
CREATE_SUBPROCESS         00000759 R      05
CREPRC_QIO_FAIL           000004D1 R      02
CS1                       00000100 R      02
CS3                       00000112 R      02
DC$_TERM                = 00000042
DDB_CTRSTR                000001E9 R      02
DDB_LOOP                  00000533 R      05
DDB_RECORD                000003EA R      05
DDB_RFA                   00000764 R      03
DEV                       000002D7 R      03
DEVBUF                    000002DF R      03
DEVDSC                    0000019D R      03
DIB$K_LENGTH            = 00000074
DIB$W_UNIT              = 0000000C
DIOLM                     0000000F R      03
DO_SUMMARY                00000409 R      05
DVI$_DEVCLASS           = 00000004
DVI$_DEVNAM             = 00000020
END_MSG                   0000020C R      02
END_MSGL                = 00000014
END_RECORD                000002A0 R      05
END_RFA                   0000076A R      03
EQUA1                     0000004B R      02
ERROR_COUNT               00000353 R      03
ERROR_EXIT                00000E96 R      05
ERROR_MSG                 00000E61 R      05
EXIT_DESC                 0000037B R      03
EXIT_HANDLER              00000F16 R      05
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_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_TRN               = 00000004
FAB$V_UFO               = 00000011
FAB$V_UPD               = 00000003
FAB$V_UPI               = 00000006
FAB$W_GBC               = 00000048
FAO_ACT                   0000013D R      03
FAO_BUF                   00000033 R      03
FAO_CHECK                 00000C69 R      05
FILE                      000003B9 R      02
FINISH_CONTROLLER         00000B05 R      05
FLAG                      00000021 R      03
FLK_MSG                   0000035D R      02
FNF_MSG                   0000031E R      02
FORMAT_ERR_MSG            000002DE R      02
GETDVI_FAIL               000003F4 R      02
HEADER_REC                00000162 R      02
INADDRESS                 00000363 R      03
INDENT                  = 00000004
INI_FAB                   000004C4 R      03
INI_RAB                   00000514 R      03
IO$M_CTRLCAST           = 00000100
IO$M_NOW                = 00000040
IO$_READVBLK            = 00000031
IO$_SETMODE             = 00000023
IO$_WRITEVBLK           = 00000030
JPI$_ASTLM              = 00000409
```

```
JPI$_BIOLM                      =  00000310
JPI$_DIOLM                      =  00000313
JPI$_TQLM                       =  00000410
JPI$_UIC                        =  00000304
JPI_LIST                           00000220  R      02
KBNT_LENGTH                     =  00000040
KILL_SUBPROCESS                    00000BC5  R      05
KNOWN_BUT_NOT_TESTABLE             000000B8  R      02
LCLPTR                             00000491  R      03
LC_BITM                         =  00000020
LIB$SIGNAL                         ********    X    05
LOGEXT                             000001B3  R      02
LOGNAM                             000002C6  R      03
LOGNAM_DSC                         000002BE  R      03
LOG_BEGIN                          00000198  R      02
LOG_FAB                            0000063C  R      03
LOG_NAME                           0000014D  R      02
LOG_RAB                            0000068C  R      03
LOG_RCD                            00000023  R      03
MAX_DEV_DESIG                   =  0000000A
MAX_SUMM_LINE                   =  00000050
MAX_UNIT_DESIG                  =  00000005
MBXT_CHAN                          0000001F  R      03
MBXCHN                             0000001B  R      03
MBXW_QIO_FAIL                      00000498  R      02
MBX_BUF                            000001A5  R      03
MBX_SIZE                        =  00000100
MBX_UNIT                           0000001D  R      03
MODE                               00000031  R      02
MPMPTR                             00000499  R      03
MPM_AND_DDB_BOTH                   0000046C  R      05
MPM_CS                             00000122  R      02
MPM_LITERAL                        00000268  R      02
MPM_RECORD                         000003C5  R      05
MSG_BLOCK                          00000377  R      03
MSG_PTR                            0000002F  R      03
NONE                               0000029E  R      02
NONE_LEN                        =  00000004
NO_RMS_AST_TABLE                   00000074  R      02
NRAT_LENGTH                     =  00000014
NULL_RECORD                        0000028F  R      05
ONEMIN                             00000183  R      02
OUTADDRESS                         0000036B  R      03
OVERHEAD_LENGTH                 =  00000003
PASS_OVER_MSG                      000005C6  R      02
PASS_OVER_SUBPROCESS               00000C24  R      05
PATH_RECORD                        0000028F  R      05
PHASE_NAME                         0000013C  R      02
PIDADR                             000002BA  R      03
POTENTIALLY_OK                     000006AF  R      05
PQL$_ASTLM                      =  00000001
PQL$_BIOLM                      =  00000002
PQL$_DIOLM                      =  00000005
PQL$_LISTEND                    =  00000000
PQL$_TQELM                      =  00000009
PROCESS_STOP_MSG                   0000054B  R      02
QUAD_STATUS                        0000035B  R      03
QUOTA_LIST                         00000004  R      03
RAB$B_RAC                       =  0000001E
RAB$C_BID                       =  00000001
RAB$C_BLN                       =  00000044
RAB$C_RFA                       =  00000002
RAB$C_SEQ                       =  00000000
RAB$L_CTX                       =  00000018
RAB$L_FAB                       =  0000003C
RAB$L_RBF                       =  00000028
RAB$L_ROP                       =  00000004
RAB$L_STS                       =  00000008
RAB$L_STV                       =  0000000C
RAB$W_RFA                       =  00000010
RAB$W_RSZ                       =  00000022
RECORD                             000003C5  R      02
REC_SIZE                        =  00000028
REPORT_NAME                        0000003D  R      02
RMS$_BCN                           ********    X    02
RMS$_BUSY                          ********    X    02
RMS$_CDA                           ********    X    02
RMS$_EOF                           ********    X    05
RMS$_FAB                           ********    X    02
RMS$_FACILITY                   =  00000001
RMS$_FLK                           ********    X    05
RMS$_FNF                           ********    X    05
RMS$_RAB                           ********    X    02
RMS_COMMON                         00000DE8  R      05
RMS_ERROR                          00000DAD  R      05
RMS_ERR_STRING                     000003D3  R      02
SEC$M_EXPREG                    =  00020000
SEC$M_GBL                       =  00000001
SHR$_ABENDD                     =  000010E0
SHR$_BEGIND                     =  00001038
SHR$_ENDEDD                     =  00001080
SHR$_OPENIN                     =  00001098
SHR$_TEXT                       =  00001130
SHRT_RPRTM                      =  00000001
SHRT_RPRTV                      =  00000000
SID_RECORD                         000002A1  R      05
SKIP_CLUS_RECORDS                  000003B7  R      05
SKIP_RECORDS                       000004BD  R      05
SS$_ABORT                       =  0000002C
SS$_CANCEL                      =  00000830
SS$_CONTROLC                    =  00000651
SS$_NORMAL                      =  00000001
SS$_OPINCOMPL                   =  000002D4
SS$_SSFAIL                      =  0000045C
SS$_WASSET                      =  00000009
SSERROR                            00000C91  R      05
SS_SYNCH_EFN                    =  00000003
START_DESC                         00000304  R      02
STATUS                             00000357         03
STOP_DESC                          00000311  R      02
STR$UPCASE                         ********    X    05
STS$K_ERROR                     =  00000002
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$V_SEVERITY                  = 00000000
SUBPROCESS_FAIL                   0000050B R      02
SUBPROC_ERROR                     00000056 R      02
SUC_EXIT                          00000671 R      05
SUMM_HEADER                       000002A2 R      02
SUMM_OUTPUT                       0000063A R      05
SUPDEV_CLUS_DENOSU                0000039B R      05
SUPDEV_DENOSU                     000004A1 R      05
SUPDEV_GBLSEC                     00000020 R      02
SUPDEV_MATCH                      00000479 R      05
SUP_FAB                           00000558 R      03
SYS$ASSIGN                        ********  GX    05
SYS$CANCEL                        ********  GX    05
SYS$CANTIM                        ********  GX    05
SYS$CLOSE                         ********  GX    05
SYS$CMKRNL                        ********  GX    05
SYS$COMMAND                       00000061 R      02
SYS$CONNECT                       ********  GX    05
SYS$CREATE                        ********  GX    05
SYS$CRELOG                        ********  GX    05
SYS$CREMBX                        ********  GX    05
SYS$CREPRC                        ********  GX    05
SYS$CRMPSC                        ********  GX    05
SYS$DCLEXH                        ********  GX    05
SYS$DELLOG                        ********  GX    05
SYS$DELPRC                        ********  GX    05
SYS$ERASE                         ********  GX    05
SYS$EXIT                          ********  GX    05
SYS$FAO                           ********   X    05
SYS$FIND                          ********  GX    05
SYS$FLUSH                         ********  GX    05
SYS$GET                           ********  GX    05
SYS$GETCHN                        ********  GX    05
SYS$GETDVI                        ********  GX    05
SYS$GETJPI                        ********  GX    05
SYS$GETMSG                        ********  GX    05
SYS$OPEN                          ********  GX    05
SYS$PUT                           ********  GX    05
SYS$PUTMSG                        ********  GX    05
SYS$QIO                           ********  GX    05
SYS$QIOW                          ********  GX    05
SYS$REWIND                        ********  GX    05
SYS$SETAST                        ********  GX    05
SYS$SETIMR                        ********  GX    05
SYS$SETPRN                        ********  GX    05
SYS$SETSFM                        ********  GX    05
SYS$TRNLOG                        ********  GX    05
SYS$TRUNCATE                      ********  GX    05
SYS$UPDATE                        ********  GX    05
TEMP_BUFF_DESC                    0000038F R      03
TEMP_BUFF_STR                     00000397 R      03
TESTABLE                          0000026C R      02
TESTABLE_LEN                    = 00000019

TESTING_MSG                       000004A9 R      03
TESTING_MSG_LENGTH              = 00000008
TESTING_MSG_TEXT                  000004B9 R      03
TEST_COUNT                        000001CC R      02
TEST_DSC                          000002A5 R      03
TEST_IMAGE                        000002AD R      03
TEST_NAME                         0000000F R      02
TEXT_BUFFER                     = 000000FA
THREEMIN                          0000017B R      02
TMPLOG_FAB                        000006D0 R      03
TMPLOG_RAB                        00000720 R      03
TQLM                              00000014 R      03
TST_CNT                           00000373 R      03
TST_CNT_STR                       000001DE R      02
TTCHAN                            00000019 R      03
UCB_CTRSTR                        000001FA R      02
UCB_N                             000005B9 R      05
UCB_RECORD                        0000028F R      05
UCB_T                             000005A3 R      05
UETINIT01                         00000000 RG     05
UETP                            = 00740000
UETP$CLSIODB                      ********    X   05
UETP$_ABENDD                    = 007410E0
UETP$_ABORTC                    = 0074832B
UETP$_BEGIND                    = 00741038
UETP$_COPY_LOG                  = 007480B1
UETP$_DENOSU                    = 00748333
UETP$_ENDEDD                    = 00741080
UETP$_ERBOXPROC                 = 00748020
UETP$_FACILITY                  = 00000074
UETP$_OPENIN                    = 00741098
UETP$_TEXT                      = 00741130
UIC                               00000000 R      03
UID$K_DDB_RTYPE                 = 00000003
UID$K_END_RTYPE                 = 00000006
UID$K_MPM_RTYPE                 = 00000005
UID$K_NULL_RTYPE                = 00000000
UID$K_UCB_RTYPE                 = 00000004
UIDDDB$T_NAME                   = 0000000B
UIDFLAG$M_DDB                   = 00000004
UIDFLAG$M_MPM                   = 00000010
UIDFLAG$M_SID                   = 00000001
UIDFLAG$M_UCB                   = 00000008
UIDGNRC$B_TYPE                  = 00000006
UIDGNRC$W_SIZE                  = 00000004
UIDMPM$W_NUMBER                 = 00000007
UIDSID$T_NODENAME               = 00000031
UIDUCB$B_DEVCLASS               = 00000009
UIDUCB$B_DEVTYPE                = 0000000A
UIDUCB$W_NUMBER                 = 00000007
UNTESTABLE                        00000285 R      02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
RODATA	0000063E (1598.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
RWDATA	00000770 (1904.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
$RMSNAM	00000034 (52.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
UETINIT01	00000F5A (3930.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.07	00:00:00.38
Command processing	111	00:00:00.70	00:00:03.81
Pass 1	668	00:00:31.42	00:01:01.50
Symbol table sort	4	00:00:03.33	00:00:07.14
Pass 2	420	00:00:07.77	00:00:16.55
Symbol table output	28	00:00:00.28	00:00:00.60
Psect synopsis output	1	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1263	00:00:43.61	00:01:30.07

The working set limit was 2000 pages.
177751 bytes (348 pages) of virtual memory were used to buffer the intermediate code.
There were 120 pages of symbol table space allocated to hold 2174 non-local and 65 local symbols.
2096 source lines were read in Pass 1, producing 43 object records in Pass 2.
76 pages of virtual memory were used to define 68 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]UETP.MLB;1	2
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	63
TOTALS (all libraries)	65

2550 GETS were required to define 65 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:UETINIT01/OBJ=OBJ$:UETINIT01 MSRC$:UETINIT01/UPDATE=(ENH$:UETINIT01)+EXECML$/LIB+SHRLIB$:UETP/LIB
```