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

```
UU      UU  EEEEEEEEEE  TTTTTTTTTT  TTTTTTTTTT  TTTTTTTTTT  YY      YY    SSSSSSSS    000000      000000
UU      UU  EEEEEEEEEE  TTTTTTTTTT  TTTTTTTTTT  TTTTTTTTTT  YY      YY    SSSSSSSS    000000      000000
UU      UU  EE              TT          TT          TT      YY      YY  SS          00      00  00      00
UU      UU  EE              TT          TT          TT      YY      YY  SS          00      00  00      00
UU      UU  EE              TT          TT          TT        YY  YY    SS          00    0000  00    0000
UU      UU  EE              TT          TT          TT        YY  YY    SS          00    0000  00    0000
UU      UU  EEEEEEEE        TT          TT          TT          YY        SSSSSS    00  00  00  00  00  00
UU      UU  EEEEEEEE        TT          TT          TT          YY        SSSSSS    00  00  00  00  00  00
UU      UU  EE              TT          TT          TT          YY              SS  0000    00  0000    00
UU      UU  EE              TT          TT          TT          YY              SS  0000    00  0000    00
UU      UU  EE              TT          TT          TT          YY              SS  00      00  00      00
UU      UU  EE              TT          TT          TT          YY              SS  00      00  00      00
UUUUUUUUUU  EEEEEEEEEE      TT          TT          TT          YY      SSSSSSSS      000000      000000
UUUUUUUUUU  EEEEEEEEEE      TT          TT          TT          YY      SSSSSSSS      000000      000000

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

```
0000     1          .TITLE UETTTYS00 VAX/VMS UETP DEVICE TEST FOR TERMINALS
0000     2          .IDENT  'V04-000'
0000     3          .ENABLE SUPPRESSION
0000     4  ;
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27  ;
0000    28
0000    29  ;++
0000    30  ; FACILITY:
0000    31  ;       This module will be distributed with VAX/VMS under the [SYSTEST]
0000    32  ;       account.
0000    33  ;
0000    34  ; ABSTRACT:
0000    35  ;       This program tests terminal devices  etc.
0000    36  ;
0000    37  ; ENVIRONMENT:
0000    38  ;       This program will run in user access mode, with interrupts enabled
0000    39  ;       at all times.  This program requires the following privileges and
0000    40  ;       quotas:
0000    41  ;
0000    42  ;--
0000    43  ;
0000    44  ; AUTHOR: Larry D. Jones,       CREATION DATE: January, 1981
0000    45  ;
0000    46  ; MODIFIED BY:
0000    47  ;
0000    48  ;       V03-009 RNH0008         Richard N. Holstein,    01-May-1984
0000    49  ;               Fix so that CTRL/C really stops the test when run in loop mode.
0000    50  ;               Fix to preserve R3 across MOVC3.  Fix to prevent RMS invalid
0000    51  ;               RAB when run in loop mode.
0000    52  ;
0000    53  ;       V03-008 RNH0007         Richard N. Holstein,    15-Feb-1984
0000    54  ;               Take advantage of new UETP message codes.  Fix SSERROR
0000    55  ;               interaction with RMS_ERROR.
0000    56  ;
0000    57  ;       V03-007 RNH0006         Richard N. Holstein,    19-Dec-1983
```

```
0000     58 ;                Give correct sentinels to Test Controller.
0000     59 ;
0000     60 ;        V03-006 RNH0005         Richard N. Holstein,    11-Nov-1983
0000     61 ;                Use decimal conversion routine for unit numbers.
0000     62 ;
0000     63 ;        V03-005 RNH0004         Richard N. Holstein,    11-Mar-1983
0000     64 ;                Don't signal ending message in EXIT_HANDLER.
0000     65 ;
0000     66 ;        V03-004 RNH0003         Richard N. Holstein,    28-Feb-1983
0000     67 ;                Allow for longer device names.
0000     68 ;
0000     69 ;        V03-003 LDJ0004         Larry D. Jones,         26-Feb-1983
0000     70 ;                Added generic support for ANSI standard terminals.
0000     71 ;
0000     72 ;        V03-002 RNH0002         Richard N. Holstein,    15-Oct-1982
0000     73 ;                Miscellaneous fixes listed in the V3B UETP Workplan.
0000     74 ;
0000     75 ;        V03-001 LDJ0003         Larry D. Jones,         17-Jun-1982
0000     76 ;                Fixed VT100 on line 5 exceeded quota bug.
0000     77 ;
0000     78 ;        V02-006 RNP0003         Robert N. Perron,       22-Jan-1982
0000     79 ;                Added watchdog timer to prevent test from hanging if we
0000     80 ;                get a hung device.
0000     81 ;
0000     82 ;        V02-005 RNP0002         Robert N. Perron,       31-Dec-1981
0000     83 ;                Fixed problem with terminals that have network logical
0000     84 ;                links.
0000     85 ;
0000     86 ;        V02-004 RNP0001         Robert N. Perron,       23-Dec-1981
0000     87 ;                Added two second pause after each page of output to give
0000     88 ;                other device tests a chance to run.
0000     89 ;
0000     90 ;        V02-003 RNH0001         Richard N. Holstein,    09-Oct-1981
0000     91 ;                Use secondary device characteristics to get around problems
0000     92 ;                when a device is spooled.  Don't test spooled devices.
0000     93 ;
0000     94 ;        V02-002 LDJ0002         Larry D. Jones,         29-Sep-1981
0000     95 ;                Added support for LA34, LA38, VT101, VT102 and VT131. Fixed
0000     96 ;                problem with disabling all lines on one unsupported terminal.
0000     97 ;                Fixed problem with VT100 one shot mode leaving terminal char.
0000     98 ;                set wrong.
0000     99 ;
0000    100 ;        V02-001 LDJ0001         Larry D. Jones,         22-Sep-1981
0000    101 ;                Fixed possible hang problem when running UETP from other than
0000    102 ;                the console.
0000    103 ;
0000    104 ;**
```

```
                        .SBTTL  Declarations
                ;
                ; INCLUDE FILES:
                ;
                ;       SYS$LIBRARY:LIB.MLB     for general definitions
                ;       SHRLIB$:UETP.MLB        for UETP definitions
                ;
                ;
                ; MACROS:
                ;
                        $CHFDEF                         ; Condition handler frame definitions
                        $DCDEF                          ; Device characteristics definitions
                        $DEVDEF                         ; Device definitions
                        $DIBDEF                         ; Device Information Block
                        $DVIDEF                         ; $GETDVI ITMLST item codes
                        $SECDEF                         ; Section definitions
                        $SHRDEF                         ; Shared messages
                        $SSDEF                          ; System Service status codes
                        $STSDEF                         ; Status return
                        $TTDEF                          ; Terminal definitions
                        $TT2DEF                         ; Extended terminal definitions
                        $UETUNTDEF                      ; UETP unit block offset definitions
                        $UETPDEF                        ; UETP
                ;
                ;
                        .MACRO TERMINAL,TERM_NAME,CLASS,TYPE,PREAMBLE,TERM_DATA,HEAD_LEN
                                .=PC1...                ; PC of terminal type table
                                .BYTE   CLASS,TYPE,HEAD_LEN ; TERM_NAME
                                .ADDRESS PC2...         ; ASCIC name address
                                .ADDRESS PREAMBLE       ; ASCIC preamble address
                                .ADDRESS TERM_DATA      ; ASCID terminal specific data address
                                PC1...=PC1...+15        ; bump to the next address
                                .=PC2...                ; point to the next ASCIC msg
                                                        ; make it's label and function ID
                TERM_NAME:
                                .ASCIC /TERM_NAME/
                                PC2...=.                ; update the string PC
                        .ENDM TERMINAL
                ;
                ; EQUATED SYMBOLS:
                ;
                ;   Facility number definitions:
00000001                        RMS$_FACILITY = 1

                ;   SHR message definitions:
00740000                        UETP = UETP$_FACILITY@STS$V_FAC_NO ; Define the UETP facility code
007410E0                        UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
00741038                        UETP$_BEGIND = UETP!SHR$_BEGIND
00741080                        UETP$_ENDEDD = UETP!SHR$_ENDEDD
00741098                        UETP$_OPENIN = UETP!SHR$_OPENIN
00741130                        UETP$_TEXT   = UETP!SHR$_TEXT

                ;   Internal flag bits...:
00000001                        TEST_OVERV    = 1               ; Set when test is over
00000002                        SAFE_TO_UPDV  = 2               ; Set if it's safe to update UETINIDEV
00000003                        SELF_TESTV    = 3               ; Set when only user terminal is tested
00000004                        BEGIN_MSGV    = 4               ; Set if "BEGIN" msg has been printed
```

```
00000005  0000   163          SCROL_CLRV   = 5                ; Set when a scroll area is defined
00000006  0000   164          TIMOUT_ERRV  = 6                ; Set if watch dog timer goes off
00000007  0000   165          CTRLC_SEENV  = 7                ; Set if test gets CTRL/C AST
          0000   166
          0000   167 ;    ...and corresponding masks:
00000002  0000   168          TEST_OVERM   = 1@TEST_OVERV
00000004  0000   169          SAFE_TO_UPDM = 1@SAFE_TO_UPDV
00000008  0000   170          SELF_TESTM   = 1@SELF_TESTV
00000010  0000   171          BEGIN_MSGM   = 1@BEGIN_MSGV
00000020  0000   172          SCROL_CLRM   = 1@SCROL_CLRV
00000040  0000   173          TIMOUT_ERRM  = 1@TIMOUT_ERRV
00000080  0000   174          CTRLC_SEENM  = 1@CTRLC_SEENV
          0000   175
          0000   176 ;    Miscellany:
00000020  0000   177          LC_BITM      = ^X20             ; Mask to convert lower case to upper
00000028  0000   178          REC_SIZE     = 40               ; UETINIDEV.DAT record size
00000084  0000   179          TEXT_BUFFER  = 132              ; Internal text buffer size
00000004  0000   180          EFN2         = 4                ; EFN used for three minute timer
00000003  0000   181          SS_SYNCH_EFN = 3                ; Synch miscellaneous system services
0000000A  0000   182          MAX_DEV_DESIG = 10              ; Longest possible controller name
00000005  0000   183          MAX_UNIT_DESIG= 5               ; Maximum unit number length
0000000D  0000   184          CR           = ^XD              ; Carriage return
0000000A  0000   185          LF           = ^XA              ; Line feed
0000001B  0000   186          ESC          = 27               ; Escape
0000000C  0000   187          FF           = 12               ; Form feed
00000013  0000   188          TERM_TYPE_CNT = 19              ; Number of terminal types known now
00000007  0000   189          MAX_NAME_SIZE = 7               ; Maximum terminal name size
0000000F  0000   190          MAX_PROC_NAME = 15              ; Maximum process name size
00000100  0000   191          HDR_OUT_SIZE  = 256             ; Maximum header output size
00000007  0000   192          SPL_UNITV    = 7                ; UETUNT$B_FLAG bit if unit is spooled
00000080  0000   193          SPL_UNITM    = 1@SPL_UNITV
          0000   194
          0000   195 ;    The following definitions are set depending on the device under test.
          0000   196 ;    Their names should not change because they are used in size calculations
          0000   197 ;    in the executable code. TSD_SIZE should be made at least as large as the
          0000   198 ;    largest terminal specific data size for any terminal in the test.
          0000   199 ;
          0000   200
00000101  0000   201          WRITE_SIZE   = 257              ; Size of device write buffer
00000600  0000   202          TSD_SIZE     = 1536             ; Size of device header buffer(3 pages)
          0000   203 ;
          0000   204 ;    Device dependent definitions
000001A4  0000   205          UETUNT$B_LINE    = UETUNT$C_DEVDEP          ; Current line number
000001A5  0000   206          UETUNT$B_HD_LEN  = UETUNT$B_LINE+1          ; Header length in lines
000001A6  0000   207          UETUNT$B_LENGTH  = UETUNT$B_HD_LEN+1        ; Page length
000001A7  0000   208          UETUNT$W_WIDTH   = UETUNT$B_LENGTH+1        ; Page width
000001AB  0000   209          UETUNT$L_CURHDR  = UETUNT$W_WIDTH+4         ; Current position in the TSD
000001AF  0000   210          UETUNT$K_CCTBL   = UETUNT$L_CURHDR+4        ; Carriage control size table addr
000001B7  0000   211          UETUNT$K_WRITE   = UETUNT$K_CCTBL+8         ; Terminal write buffer
000002B8  0000   212          UETUNT$K_HEADER  = UETUNT$K_WRITE+257       ; Header buffer
00000002  0000   213          UETUNT$V_2PL     = 2                        ; 2 page limit flag
00000004  0000   214          UETUNT$M_2PL     = 1@UETUNT$V_2PL
          0000   215
00000013  0000   216          DEVDEP_SIZE      = <UETUNT$K_WRITE--
          0000   217                             UETUNT$C_INDSIZ> ; Size of device dependent part of UETUNT
          0000   218
          0000   219          PAGES = <<UETUNT$C_INDSIZ+-     ; Add together all of the pieces...
```

```
                    0000   220
                    0000   221
                    0000   222
          00000005  0000   223
                    0000   224

DEVDEP_SIZE+-           ; ...which make up a UETP unit block...
WRITE_SIZE+-            ; ...to give to the $EXPREG service below
TSD_SIZE+-
5115/512>
```

```
                                0000   226          .SBTTL  Read-Only Data
                            00000000   227          .PSECT  RODATA,NOEXE,NOWRT,PAGE
                                0000   228
                                0000   229 ACNT_NAME:                                ; Process name on exit
53 45 54 53 59 53 00000008'010E0000'   0000   230          .ASCID  /SYSTEST/
                                  54   000E
                                000F   231
                                000F   232 TEST_NAME:                                ; This test name
59 54 54 54 45 55 00000017'010E0000'   000F   233          .ASCID  /UETTTYS00/
                              30 30 53   001D
                                0020   234
                                0020   235 SUPDEV_GBLSEC:                            ; How we access UETSUPDEV.DAT
50 55 53 54 45 55 00000028'010E0000'   0020   236          .ASCID  /UETSUPDEV/
                              56 45 44   002E
                                0031   237
                                0031   238 CONTROLLER:                               ; Logical name of controller
41 4E 4C 52 54 43 00000039'010E0000'   0031   239          .ASCID  /CTRLNAME/
                                 45 4D   003F
                                0041   240
                                0041   241 MODE:                                     ; Run mode logical name
      45 44 4F 4D 00000049'010E0000'   0041   242          .ASCID  /MODE/
                                004D   243
                                004D   244 NO_RMS_AST_TABLE:                         ; List of errors for which...
                    00C00000'   004D   245          .LONG   RMS$_BLN                ;  ..RMS cannot deliver an AST...
                    00000000'   0051   246          .LONG   RMS$_BUSY               ; ...even if one has an ERR= arg
                    00000000'   0055   247          .LONG   RMS$_CDA                ; Note that we can search table...
                    00000000'   0059   248          .LONG   RMS$_FAB                ; ...via MATCHC since <31:16>...
                    00000000'   005D   249          .LONG   RMS$_RAB                ; ...pattern can't be in <15:0>
                    00000014    0061   250 NRAT_LENGTH = .-NO_RMS_AST_TABLE
                                0061   251
                                0061   252 SYS$INPUT:                               ; Name of device from which...
4E 49 24 53 59 53 00000069'010E0000'   0061   253          .ASCID  /SYS$INPUT/        ; ...the test can be aborted
                              54 55 50   006F
                                0072   254
                                0072   255 INPUT_ITMLST:                            ; $GETDVI arg list for SYS$INPUT
                        0020 0040      0072   256          .WORD   64,DVI$_DEVNAM     ; We need the equivalence name
                0000000C'00000014'     0076   257          .LONG   BUFFER,BUFFER_PTR
                        00000000       007E   258          .LONG   0                  ; Terminate the list
                                0082   259
                                0082   260 TERM_ITMLST:                            ; $GETDVI arg list for terminal
                        001C 0004      0082   261          .WORD   4,DVI$_DEVDEPEND2  ; We need the extended term. char.
                00000000 00000251'     0086   262          .LONG   XTERM_CHAR,0
                        00000000       008E   263          .LONG   0                  ; Terminate the list
                                0092   264
                                0092   265 CS1:                                    ; Device class and type control string
21 20 42 58 32 21 0000009A'010E0000'   0092   266          .ASCID  /!2XB !2XB /
                              20 42 58 32   00A0
                                00A4   267
                                00A4   268 CS3:                                    ; Device class-only control string
2A 20 42 58 32 21 000000AC'010E0000'   00A4   269          .ASCID  /!2XB **/
                                 2A    00B2
                                00B3   270
                                00B3   271 CNTRLCMSG:
65 74 72 6F 62 41 000000BB'010E0000'   00B3   272          .ASCID  \Aborted via a user CTRL/C\
72 65 73 75 20 61 20 61 69 76 20 64    00C1
                  43 2F 4C 52 54 43 20 00CD
                                00D4   273
```

```
                                                00D4    274 NO_CTRLNAME:
6E 6F 63 20 6F 4E 000000DC'010E0000'            00D4    275         .ASCID  /No controller specified./
63 65 70 73 20 72 65 6C 6C 6F 72 74             00E2
                  2E 64 65 69 66 69             00EE
                                                00F4    276
                                                00F4    277 DEAD_CTRLNAME:
20 74 27 6E 61 43 000000FC'010E0000'            00F4    278         .ASCID  /Can't test controller !AS, marked as unusable in UETINIDEV.DAT./
6C 6F 72 74 6E 6F 63 20 74 73 65 74             0102
72 61 6D 20 2C 53 41 21 20 72 65 6C             010E
61 73 75 6E 75 20 73 61 20 64 65 6B             011A
4E 49 54 45 55 20 6E 69 20 65 6C 62             0126
         2E 54 41 44 2E 56 45 44 49             0132
                                                013B    279
                                                013B    280 NOUNIT_SELECTED:
69 6E 75 20 6F 4E 00000143'010E0000'            013B    281         .ASCID  /No units selected for testing./
20 64 65 74 63 65 6C 65 73 20 73 74             0149
2E 67 6E 69 74 73 65 74 20 72 6F 66             0155
                                                0161    282
                                                0161    283 ILLEGAL_REC:
61 67 65 6C 6C 49 00000169'010E0000'            0161    284         .ASCID  /Illegal record format in file UETINIDEV.DAT!/
72 6F 66 20 64 72 6F 63 65 72 20 6C             016F
20 65 6C 69 66 20 6E 69 20 74 61 6D             017B
41 44 2E 56 45 44 49 4E 49 54 45 55             0187
                              21 54             0193
                                                0195    285
                                                0195    286 PASS_MSG:
66 6F 20 64 6E 45 0000019D'010E0000'            0195    287         .ASCID  /End of pass !UL with !UL iterations at !%D./
69 77 20 4C 55 21 20 73 73 61 70 20             01A3
61 72 65 74 69 20 4C 55 21 20 68 74             01AF
44 25 21 20 74 61 20 73 6E 6F 69 74             01BB
                                    2E          01C7
                                                01C8    288
                                                01C8    289 HEAD_CTRSTR:
58 41 56 43 41 21 000001D0'010E0000'            01C8    290         .ASCID  \!ACVAX/VMS UETP terminal test UETTTYS00 for the !AC at !AS!/\
65 74 20 50 54 45 55 20 53 4D 56 2F             01D6
20 74 73 65 74 20 6C 61 6E 69 6D 72             01E2
6F 66 20 30 30 53 59 54 54 54 45 55             01EE
74 61 20 43 41 21 20 65 68 74 20 72             01FA
                  2F 21 53 41 21 20             0206
                                                020C    291
                                                020C    292 TIME_OUT_MSG:
65 63 69 76 65 44 00000214'010E0000'            020C    293         .ASCID  \Device off line or not testable.\
72 6F 20 65 6E 69 6C 20 66 66 6F 20             021A
6C 62 61 74 73 65 74 20 74 6F 6E 20             0226
                              2E 65             0232
                                                0234    294
                                                0234    295 TIMOUT_ERR_MSG:
6F 2D 65 6D 69 54 0000023C'010E0000'            0234    296         .ASCID  /Time-out error on device !AS./
20 6E 6F 20 72 6F 72 72 65 20 74 75             0242
   2E 53 41 21 20 65 63 69 76 65 64             024E
                                                0259    297
                                                0259    298 PROMPT:
64 20 72 65 6C 6C 6F 72 74 6E 6F 43             0259    299         .ASCII  /Controller designation?: /
3A 3F 6E 6F 69 74 61 6E 67 69 73 65             0265
                                    20          0271
                        00000019                0272    300         PMTSIZ = .-PROMPT
                                                0272    301
```

```
                                0272    302 INIDEV_UPDERR:                          ; Error during exit handler
20 72 6F 72 72 45 0000027A'010E0000' 0272    303         .ASCID  /Error updating UETINIDEV.DAT./
54 45 55 20 67 6E 69 74 61 64 70 75  0280
   2E 54 41 44 2E 56 45 44 49 4E 49  028C
                                0297    304
                                0297    305 THREEMIN:                               ; 3 minute delta time
                FFFFFFFF 94B62E00  0297    306         .LONG   -10*1000*1000*180,-1
                                029F    307
                                029F    308 TWOMIN:                                 ; 2 minute delta time
                FFFFFFFF B8797400  029F    309         .LONG   -10*1000*1000*120,-1
                                02A7    310
                                02A7    311 FIVESEC:                                ; 5 second delta time
                FFFFFFFF FD050F80  02A7    312         .LONG   -10*1000*1000*5,-1
                                02AF    313
                                02AF    314 TWOSEC:                                 ; 2 seconds delta time
                FFFFFFFF FECED300  02AF    315         .LONG   -10*1000*1000*2,-1
                                02B7    316
                                02B7    317 UNIT_DESC:                              ; Descriptor used to convert unit #
                        00000005  02B7    318         .LONG   5
                        0000001A' 02BB    319         .ADDRESS BUFFER+6
                                02BF    320
                                02BF    321 CONT_DESC:                              ; Descriptor used to convert controller...
                    0000 0028  02BF    322         .WORD   REC_SIZE,0              ; ...from lowercase to uppercase
                        00000014' 02C3    323         .ADDRESS BUFFER
                                02C7    324
                                02C7    325 FILE:                                   ; Fills in RMS_ERR_STRING
      65 6C 69 66 000002CF'010E0000' 02C7    326         .ASCID  /file/
                                02D3    327
                                02D3    328 RECORD:                                 ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 000002DB'010E0000' 02D3    329         .ASCID  /record/
                                02E1    330
                                02E1    331 RMS_ERR_STRING:                         ; Announces an RMS error
41 21 20 53 4D 52 000002E9'010E0000' 02E1    332         .ASCID  /RMS !AS error in file !AD/
66 20 6E 69 20 72 6F 72 72 65 20 53  02EF
                44 41 21 20 65 6C 69  02FB
                                0302    333
                                0302    334 HUNG_TERMINAL:
59 54 54 54 45 55 0000030A'010E0000' 0302    335         .ASCID  /UETTTYS00 test is hung!/
20 73 69 20 74 73 65 74 20 30 30 53  0310
                   21 67 6E 75 68  031C
                                0321    336
                                0321    337 NOSPOOLED:                              ; We found a spooled device
65 6C 62 61 6E 55 00000329'010E0000' 0321    338         .ASCID  /Unable to test !AS because it is spooled./
53 41 21 20 74 73 65 74 20 6F 74 20  032F
20 74 69 20 65 73 75 61 63 65 62 20  033B
   2E 64 65 6C 6F 6F 70 73 20 73 69  0347
                                0352    339
                                0352    340 SIZE_TBL:                               ; table of output record sizes
                            0000  0352    341         .WORD   0                   ; line size added in later
                            0000  0354    342         .WORD   0
                            0002  0356    343         .WORD   2
                            0001  0358    344         .WORD   1
                        00000008  035A    345         SIZE_TBL_LEN=.-SIZE_TBL
                                035A    346
                                035A    347 DATA_BUF:                               ; 256 bytes of all printable characters
                        00000021  035A    348         A=^A/!/
                                035A    349         .REPT 94
```

```
                 035A    350          .BYTE A
                 035A    351          A=A+1
              21 035A    352          .ENDR
00000021         03B8    353          A=^A/./
                 03B8    354          .REPT 94
                 03B8    355          .BYTE A
                 03B8    356          A=A+1
              21 03B8    357          .ENDR
00000021         0416    358          A=^A/!/
```

```
                     045A   366 ;
                     045A   367 ; Carriage control longword format.
                     045A   368 ;
                     045A   369 ;       +----------+----------+----------+----------+
                     045A   370 ;       .  postfix .  prefix  ! data Type! Fortran  !
                     045A   371 ;       +----------+----------+----------+----------+
                     045A   372 ;
                     045A   373 ;
                     045A   374 CC_TBL:
00 01 01 00          045A   375         .BYTE   0,1,1,0                 ; Prefix newline PosDfix null
01 01 01 00          045E   376         .BYTE   0,1,1,1                 ; Prefix-Postfix newline
00 00 02 00          0462   377         .BYTE   0,2,0,0                 ; Prefix-Postfix null
01 00 01 00          0466   378         .BYTE   0,1,0,1                 ; Prefix null Postfix newline
00 8D 02 00          046A   379         .BYTE   0,2,^X8D,0              ; Prefix 7 bit Postfix null
00 CD 02 00          046E   380         .BYTE   0,2,^XCD,0              ; Prefix 8 bit Postfix null
01 8D 01 00          0472   381         .BYTE   0,1,^X8D,1              ; Prefix 7 bit Postfix newline
01 CD 01 00          0476   382         .BYTE   0,1,^XCD,1              ; Prefix 8 bit Postfix newline
8D 00 01 00          047A   383         .BYTE   0,1,0,^X8D              ; Prefix null Postfix 7 bit
8D 01 03 00          047E   384         .BYTE   0,3,1,^X8D              ; Prefix newline Postfix 7 bit
8D 8D 03 00          0482   385         .BYTE   0,3,^X8D,^X8D           ; Prefix 7 bit Postfix 7 bit
8D CD 03 00          0486   386         .BYTE   0,3,^XCD,^X8D           ; Prefix 8 bit Postfix 7 bit
CD 00 01 00          048A   387         .BYTE   0,1,0,^XCD              ; Prefix null Postfix 8 bit
CD 01 03 00          048E   388         .BYTE   0,3,1,^XCD              ; Prefix newline Postfix 8 bit
CD 8D 03 00          0492   389         .BYTE   0,3,^X8D,^XCD           ; Prefix 7 bit Postfix 8 bit
CD CD 02 00          0496   390         .BYTE   0,2,^XCD,^XCD           ; Prefix 8 bit Postfix 8 bit
00 00 01 00          049A   391         .BYTE   0,1,0,0                 ; Prefix-Postfix null
    00000011         049E   392         CC_TBL_SIZE = <.-CC_TBL>/4
                     049E   393 ;
                     049E   394 ; The following table contains the device class and type of all devices supported
                     049E   395 ; by this test.  Any new devices must be added to this table or the device will
                     049E   396 ; be tested as a DT$_TTYUNKN. The only other information that has to be supplied
                     049E   397 ; to this program to test a new device is any device specific data. (See below)
                     049E   398 ; The table contains a byte for the device class, a byte for the device type, and
                     049E   399 ; an address of a device descriptor for the device specific data or zero if none
                     049E   400 ; exists.
                     049E   401 ;
                     049E   402
                     049E   403 TERM_NAMES:
00000523             049E   404         .BLKB   TERM_TYPE_CNT*MAX_NAME_SIZE     ; number of terminal macros * max by
                     0523   405
00000523             0523   406         PC1... = .
0000049E             0523   407         PC2... = TERM_NAMES
                     0523   408 TESDEV_TBL:
00000640             0523   409         .BLKB   15*TERM_TYPE_CNT                ; Allocate space for each macro
00000640             0640   410         SAVE_PC... =
                     0640   411         TERMINAL VT100,DC$_TERM,DT$_VT100,VT100_PREAMBLE,VT100_DATA,0
                     04A4   412         TERMINAL VT101,DC$_TERM,DT$_VT101,VT100_PREAMBLE,VT100_DATA,0
                     04AA   413         TERMINAL VT102,DC$_TERM,DT$_VT102,VT100_PREAMBLE,VT100_DATA,0
                     04B0   414         TERMINAL VT125,DC$_TERM,DT$_VT125,VT100_PREAMBLE,VT100_DATA,0
                     04B6   415         TERMINAL VT131,DC$_TERM,DT$_VT131,VT100_PREAMBLE,VT100_DATA,0
                     04BC   416         TERMINAL VT132,DC$_TERM,DT$_VT132,VT100_PREAMBLE,VT100_DATA,0
                     04C2   417         TERMINAL VT52,DC$_TERM,DT$_VT52,VT52_PREAMBLE,VT52_DATA,4
                     04C7   418         TERMINAL LA120,DC$_TERM,DT$_LA120,FORM_FEED,LA120_DATA,2
                     04CD   419         TERMINAL LA34,DC$_TERM,DT$_LA34,FORM_FEED,LA36_DATA,2
                     04D2   420         TERMINAL LA36,DC$_TERM,DT$_LA36,FORM_FEED,LA36_DATA,2
                     04D7   421         TERMINAL LA38,DC$_TERM,DT$_LA38,FORM_FEED,LA36_DATA,2
                     04DC   422         TERMINAL VT55,DC$_TERM,DT$_VT55,VT52_PREAMBLE,VT52_DATA,2
```

```
                04E1   423          TERMINAL VT5X,DC$_TERM,DT$_VT5X,VT52_PREAMBLE,VT52_DATA,2
                04E6   424          TERMINAL VT05,DC$_TERM,DT$_VT05,VT05_PREAMBLE,VT05_DATA,2
000005F5        04EB   425          .=PC1...
                05F5   426 UNKNOWN1:
                05F5   427          TERMINAL UNKNOWN,DC$_TERM,DT$_TTYUNKN,FORM_FEED,UNKN_DATA,2
                04F3   428          TERMINAL LA180,DC$_LP,DT$_LA180,FORM_FEED,LA180_DATA,2
                04F9   429          TERMINAL LA11,DC$_LP,DT$_LA11,FORM_FEED,LA11_DATA,2
                04FE   430          TERMINAL LP11,DC$_LP,DT$_LP11,FORM_FEED,LP11_DATA,2
                0503   431          TERMINAL LAX,DC$_LP,DT$_LAX,FORM_FEED,LAX_DATA,2
00000640        0507   432          . = SAVE_PC...
00000640        0640   433          TESDEV_TBL_END = .
                0640   434
                0640   435          .ALIGN  LONG
                0640   436 ;
                0640   437 ; The following is device specific data. To add data of this type do the following:
                0640   438 ;
                0640   439 ;       1.      Create an ASCIC string for the terminal data to be printed before
                0640   440 ;               the header (PREAMBLE).
                0640   441 ;
                0640   442 ;       2.      Create a string descriptor for the data.
                0640   443 ;
                0640   444 ;       3.      Enter the terminal data and update the TSD_SIZE constant if the
                0640   445 ;               new terminal specific data size is larger than the current value
                0640   446 ;               for TSD_SIZE.
                0640   447 ;
                0640   448 ;       4.      Enter the terminal information in the above table
                0640   449 ;
                0640   450 ; Terminal specific data for the VT100 terminal
                0640   451 ;
                0640   452
                0640   453 VT100_DATA:
        000003E1'  0640   454          .LONG   VT100L
        00000657'  0644   455          .ADDRESS VT100_OUT_DATA
                0648   456
                0648   457 VT100_PREAMBLE:
                0648   458 ;
                0648   459 ; Set-up and header line
                0648   460 ;
              0E'  0648   461          .BYTE   VT100_PRE_LEN
           37 1B   0649   462          .ASCII  <ESC>/7/        ; Save attributes
           3C 1B   064B   463          .ASCII  <ESC>/</        ; Enter ANSI mode
     4A 32 5B 1B   064D   464          .ASCII  <ESC>/[2J/      ; Clear the screen
48 31 3B 31 5B 1B  0651   465          .ASCII  <ESC>/[1;1H/    ; Go home
        0000000E   0657   466          VT100_PRE_LEN = .-VT100_PREAMBLE-1
                0657   467 VT100_OUT_DATA:
71 31 3B 30 5B 1B  0657   468          .ASCII  <ESC>/[0;1q/    ; LED 1
6D 31 3B 30 5B 1B  065D   469          .ASCII  <ESC>/[0;1m/    ; Set bold
                0663   470 ;
                0663   471 ; Set-up double height and double width line modes
                0663   472 ;
                0663   473          .REPT   3
                0663   474          .ASCII  <ESC>/#3/       ; Double height top
                0663   475          .ASCII  <ESC>/E/        ; Next line
                0663   476          .ASCII  <ESC>/#4/       ; Double height bottom
                0663   477          .ASCII  <ESC>/E/        ; Next line
     33 23 1B   0663   478          .ENDR
                0681   479          .REPT   3
```

```
                                                0681    480         .ASCII  <ESC>/#6/       ; Double width
                                                0681    481         .ASCII  <ESC>/E/        ; Next line
                                     36 23 1B   0681    482         .ENDR
                                                0690    483 ;
                                                0690    484 ; Write double height lines
                                                0690    485 ;
                                  66 31 3B 32 5B 1B     0690    486         .ASCII  <ESC>/[2;1f/    ; Goto line 2, col 1
4C 4B 4A 49 48 47 46 45 44 43 42 41             0696    487         .ASCII  'ABCDEFGHIJKLMNOPQRSTUVWXYZ-+/\[]01234567'<ESC>'E'
58 57 56 55 54 53 52 51 50 4F 4E 4D             06A2
33 32 31 30 5D 5B 5C 2F 2B 2D 5A 59             06AE
                  45 1B 37 36 35 34             06BA
4C 4B 4A 49 48 47 46 45 44 43 42 41             06C0    488         .ASCII  'ABCDEFGHIJKLMNOPQRSTUVWXYZ-+/\[]01234567'<ESC>'E'
58 57 56 55 54 53 52 51 50 4F 4E 4D             06CC
33 32 31 30 5D 5B 5C 2F 2B 2D 5A 59             06D8
                  45 1B 37 36 35 34             06E4
                  71 32 3B 30 5B 1B             06EA    489         .ASCII  <ESC>/[0;2q/    ; LED 2
                  6D 34 3B 30 5B 1B             06F0    490         .ASCII  <ESC>/[0;4m/    ; Set underscore
6C 6B 6A 69 68 67 66 65 64 63 62 61             06F6    491         .ASCII  /abcdefghijklmnopqrstuvwxyz_!`~{}/<ESC>/E/
78 77 76 75 74 73 72 71 70 6F 6E 6D             0702
      45 1B 7D 7B 7E 60 7C 5F 7A 79             070E
6C 6B 6A 69 68 67 66 65 64 63 62 61             0718    492         .ASCII  /abcdefghijklmnopqrstuvwxyz_!`~{}/<ESC>/E/
78 77 76 75 74 73 72 71 70 6F 6E 6D             0724
      45 1B 7D 7B 7E 60 7C 5F 7A 79             0730
            71 32 3B 31 3B 30 5B 1B             073A    493         .ASCII  <ESC>/[0;1;2q/  ; LED 1 and 2
            6D 34 3B 31 3B 30 5B 1B             0742    494         .ASCII  <ESC>/[0;1;4m/  ; Bold and underscore
                        30 28 1B                074A    495         .ASCII  <ESC>/(0/       ; Special graphics and line drawing set
6C 6B 6A 69 68 67 66 65 64 63 62 61             074D    496         .ASCII  /abcdefghijklmnopqrstuvwxyz_!`~{}/<ESC>/E/
78 77 76 75 74 73 72 71 70 6F 6E 6D             0759
      45 1B 7D 7B 7E 60 7C 5F 7A 79             0765
6C 6B 6A 69 68 67 66 65 64 63 62 61             076F    497         .ASCII  /abcdefghijklmnopqrstuvwxyz_!`~{}/<ESC>/E/
78 77 76 75 74 73 72 71 70 6F 6E 6D             077B
      45 1B 7D 7B 7E 60 7C 5F 7A 79             0787
                                                0791    498 ;
                                                0791    499 ; Write double width lines
                                                0791    500 ;
                  71 33 3B 30 5B 1B             0791    501         .ASCII  <ESC>/[0;3q/    ; LED 3
                  6D 35 3B 30 5B 1B             0797    502         .ASCII  <ESC>/[0;5m/    ; Blink
                              42 28 1B          079D    503         .ASCII  <ESC>/(B/       ; USASCII
4C 4B 4A 49 48 47 46 45 44 43 42 41             07A0    504         .ASCII  'ABCDEFGHIJKLMNOPQRSTUVWXYZ-+/\[]01234567'<ESC>'E'
58 57 56 55 54 53 52 51 50 4F 4E 4D             07AC
33 32 31 30 5D 5B 5C 2F 2B 2D 5A 59             07B8
                  45 1B 37 36 35 34             07C4
            71 33 3B 31 3B 30 5B 1B             07CA    505         .ASCII  <ESC>/[0;1;3q/  ; LED 1 and 3
            6D 35 3B 31 3B 30 5B 1B             07D2    506         .ASCII  <ESC>/[0;1;5m/  ; Bold and blink
6C 6B 6A 69 68 67 66 65 64 63 62 61             07DA    507         .ASCII  /abcdefghijklmnopqrstuvwxyz_!`~{}/<ESC>/E/
78 77 76 75 74 73 72 71 70 6F 6E 6D             07E6
      45 1B 7D 7B 7E 60 7C 5F 7A 79             07F2
            71 33 3B 32 3B 30 5B 1B             07FC    508         .ASCII  <ESC>/[0;2;3q/  ; LED 2 and 3
            6D 35 3B 34 3B 30 5B 1B             0804    509         .ASCII  <ESC>/[0;4;5m/  ; underscore and blink
                        30 28 1B                080C    510         .ASCII  <ESC>/(0/       ; Special graphics and line drawing set
6C 6B 6A 69 68 67 66 65 64 63 62 61             080F    511         .ASCII  /abcdefghijklmnopqrstuvwxyz_!`~{}/<ESC>/E/
78 77 76 75 74 73 72 71 70 6F 6E 6D             081B
      45 1B 7D 7B 7E 60 7C 5F 7A 79             0827
                                                0831    512 ;
                                                0831    513 ; Write normal mode graphics line
                                                0831    514 ;
      71 33 3B 32 3B 31 3B 30 5B 1B             0831    515         .ASCII  <ESC>/[0;1;2;3q/ ; LED 1, 2, and 3
```

```
      6D 35 3B 34 3B 31 3B 30 5B 1B  083B        .ASCII  <ESC>/[0;1;4;5m/ ; Bold, underscore, and blink
      6A 69 68 67 66 65 64 63 62 61  0845        .ASCII  /abcdefghij/
                  71 34 3B 30 5B 1B  084F        .ASCII  <ESC>/[0;4q/    ; LED 4
                  6D 37 3B 30 5B 1B  0855        .ASCII  <ESC>/[0;7m/    ; Reverse video
      74 73 72 71 70 6F 6E 6D 6C 6B  085B        .ASCII  /klmnopqrst/
            71 34 3B 31 3B 30 5B 1B  0865        .ASCII  <ESC>/[0;1;4q/  ; LED 1 and 4
            6D 37 3B 31 3B 30 5B 1B  086D        .ASCII  <ESC>/[0;1;7m/  ; Bold and reverse video
      7E 60 7C 5F 7A 79 78 77 76 75  0875        .ASCII  /uvwxyz_|`~/
            71 34 3B 32 3B 30 5B 1B  087F        .ASCII  <ESC>/[0;2;4q/  ; LED 2 and 4
            6D 37 3B 34 3B 30 5B 1B  0887        .ASCII  <ESC>/[0;4;7m/  ; Underscore and reverse
      68 67 66 65 64 63 62 61 7D 7B  088F        .ASCII  /{}abcdefgh/
      71 34 3B 32 3B 31 3B 30 5B 1B  0899        .ASCII  <ESC>/[0;1;2;4q/ ; LED 1, 2, and 4
      6D 37 3B 34 3B 31 3B 30 5B 1B  08A3        .ASCII  <ESC>/[0;1;4;7m/ ; Bold, underscore and reverse
      72 71 70 6F 6E 6D 6C 6B 6A 69  08AD        .ASCII  /ijklmnopqr/
            71 34 3B 33 3B 30 5B 1B  08B7        .ASCII  <ESC>/[0;3;4q/  ; LED 3 and 4
            6D 37 3B 35 3B 30 5B 1B  08BF        .ASCII  <ESC>/[0;5;7m/  ; Blink and reverse
      7C 5F 7A 79 78 77 76 75 74 73  08C7        .ASCII  /stuvwxyz_|/
      71 34 3B 33 3B 31 3B 30 5B 1B  08D1        .ASCII  <ESC>/[0;1;3;4q/ ; LED 1, 3, and 4
      6D 37 3B 35 3B 31 3B 30 5B 1B  08DB        .ASCII  <ESC>/[0;1;5;7m/ ; Bold, blink, and reverse
      66 65 64 63 62 61 7D 7B 7E 60  08E5        .ASCII  /`~{}abcdef/
      71 34 3B 33 3B 32 3B 30 5B 1B  08EF        .ASCII  <ESC>/[0;2;3;4q/ ; LED 2, 3, and 4
      6D 37 3B 35 3B 34 3B 30 5B 1B  08F9        .ASCII  <ESC>/[0;4;5;7m/ ; Underscore, blink, and reverse
      70 6F 6E 6D 6C 6B 6A 69 68 67  0903        .ASCII  /ghijklmnop/
                                     090D  ;
                                     090D  ; Test cursor control features
                                     090D  ;
31 3B 30 5B 1B 48 33 33 3B 37 5B 1B  090D        .ASCII  <ESC>/[7;33H/<ESC>/[0;1;2q/<ESC>/[0;1;4m&*()=:;?/
26 6D 34 3B 31 3B 30 5B 1B 71 32 3B  0919
               3F 3B 3A 3D 29 28 2A  0925
                     3F 41 31 5B 1B  092C        .ASCII  <ESC>/[1A?/
6D 34 3B 30 5B 1B 71 32 3B 30 5B 1B  0931        .ASCII  <ESC>/[0;2q/<ESC>/[0;4m/<ESC>/[1A^/
                     5E 41 31 5B 1B  093D
                     5E 41 31 5B 1B  0942        .ASCII  <ESC>/[1A^/
                     25 44 31 5B 1B  0947        .ASCII  <ESC>/[1D%/
                     24 44 32 5B 1B  094C        .ASCII  <ESC>/[2D$/
                     23 44 32 5B 1B  0951        .ASCII  <ESC>/[2D#/
                     40 44 32 5B 1B  0956        .ASCII  <ESC>/[2D@/
                     21 44 32 5B 1B  095B        .ASCII  <ESC>/[2D!/
                     39 44 32 5B 1B  0960        .ASCII  <ESC>/[2D9/
                     38 44 32 5B 1B  0965        .ASCII  <ESC>/[2D8/
         38 42 31 5B 1B 44 31 5B 1B  096A        .ASCII  <ESC>/[1D/<ESC>/[1B8/
71 32 3B 31 3B 30 5B 1B 44 31 5B 1B  0973        .ASCII  <ESC>/[1D/<ESC>/[0;1;2q/<ESC>/[0;1;4m/<ESC>/[1B&*()=:;/
42 31 5B 1B 6D 34 3B 31 3B 30 5B 1B  097F
               3B 3A 3D 29 28 2A 26  098B
5B 1B 71 32 3B 30 5B 1B 44 31 5B 1B  0992        .ASCII  <ESC>/[1D/<ESC>/[0;2q/<ESC>/[0;4m/<ESC>/[1A%/
         25 41 31 5B 1B 6D 34 3B 30  099E
                     24 44 32 5B 1B  09A7        .ASCII  <ESC>/[2D$/
                     23 44 32 5B 1B  09AC        .ASCII  <ESC>/[2D#/
                     4C 44 32 5B 1B  09B1        .ASCII  <ESC>/[2D@/
                     21 44 32 5B 1B  09B6        .ASCII  <ESC>/[2D!/
                     39 44 32 5B 1B  09BB        .ASCII  <ESC>/[2D9/
3B 30 5B 1B 42 35 5B 1B 43 36 5B 1B  09C0        .ASCII  <ESC>/[6C/<ESC>/[5B/<ESC>/[0;2;3q/<ESC>/[0;4;5m?/
6D 35 3B 34 3B 30 5B 1B 71 33 3B 32  09CC
                                 3F  09D8
                     3B 44 31 5B 1B  09D9        .ASCII  <ESC>/[1D;/
                     3A 44 32 5B 1B  09DE        .ASCII  <ESC>/[2D:/
                     3D 44 32 5B 1B  09E3        .ASCII  <ESC>/[2D=/
```

```
                     29 44 32 5B 1B   09E8    565          .ASCII  <ESC>/[2D)/
                     28 44 32 5B 1B   09ED    566          .ASCII  <ESC>/[2D(/
                     2A 44 32 5B 1B   09F2    567          .ASCII  <ESC>/[2D*/
                     26 44 32 5B 1B   09F7    568          .ASCII  <ESC>/[2D&/
3B 30 5B 1B 41 31 5B 1B 44 31 5B 1B   09FC    569          .ASCII  <ESC>/[1D/<ESC>/[1A/<ESC>/[0;1;3q/<ESC>/[0;1;5m89!@#$%^/
6D 35 3B 31 3B 30 5B 1B 71 33 3B 31   0A08
           5E 25 24 23 40 21 39 38    0A14
                  71 30 5B 1B        0A1C    570          .ASCII  <ESC>/[0q/       ; No LEDs
           72 34 32 3B 32 31 5B 1B    0A20    571          .ASCII  <ESC>/[12;24r/   ; Set scroll to 12 thru 24
              48 31 3B 32 31 5B 1B    0A28    572          .ASCII  <ESC>/[12;1H/    ; Go to scroll area
                     6D 30 5B 1B      0A2F    573          .ASCII  <ESC>/[0m/       ; Reset attributes
                        42 28 1B      0A33    574          .ASCII  <ESC>/(B/        ; Reset to USASCII
                           42 1B      0A36    575          .ASCII  <ESC>/8/         ; Restore cursor and attributes
                     000003E1         0A38    576          VT100L = .-VT100_OUT_DATA
                                      0A38    577
                                      0A38    578 VT100_ORIG_SCROL:                ; Return to full screen scroll
3B 33 32 5B 1B 72 34 32 3B 30 5B 1B   0A38    579          .ASCII  <ESC>/[0;24r/<ESC>/[23;1H/
                              48 31   0A44
                     0000000E         0A46    580          VOSL = .-VT100_ORIG_SCROL
                                      0A46    581
                                      0A46    582 ;
                                      0A46    583 ; Terminal specific data for the VT52 terminal
                                      0A46    584 ;
                                      0A46    585 VT52_DATA:
                     00000168'        0A46    586          .LONG   VT52L
                     00000A53'        0A4A    587          .ADDRESS VT52_OUT_DATA
                                      0A4E    588 VT52_PREAMBLE:
                           04'        0A4E    589          .BYTE   VT52_PRE_LEN
                        48 1B         0A4F    590          .ASCII  <ESC>/H/                ; Home
                        4A 1B         0A51    591          .ASCII  <ESC>/J/                ; Clear
                     00000004         0A53    592          VT52_PRE_LEN = .-VT52_PREAMBLE-1
                                      0A53    593 VT52_OUT_DATA:
                              46 1B   0A53    594          .ASCII  <ESC>/F/                ; Enter graphics mode
74 6B 6A 69 68 67 66 65 64 63 62 61   0A55    595          .ASCII  /abcdefghijktuvwxyz{|}~/
      7E 7D 7C 7B 7A 79 78 77 76 75   0A61
      7D 44 1B 44 1B 7E 35 22 59 1B   0A6B    596          .ASCII  <ESC>/Y/<34><53>/~/<ESC>/D/<ESC>/D}/
44 1B 7B 44 1B 44 1B 7C 44 1B 44 1B   0A75    597          .ASCII  <ESC>/D/<ESC>/D|/<ESC>/D/<ESC>/D{/<ESC>/D/<ESC>/Dz/
                        7A 44 1B      0A81
44 1B 78 44 1B 44 1B 79 44 1B 44 1B   0A84    598          .ASCII  <ESC>/D/<ESC>/Dy/<ESC>/D/<ESC>/Dx/<ESC>/D/<ESC>/Dw/
                        77 44 1B      0A90
44 1B 75 44 1B 44 1B 76 44 1B 44 1B   0A93    599          .ASCII  <ESC>/D/<ESC>/Dv/<ESC>/D/<ESC>/Du/<ESC>/D/<ESC>/Dt/
                        74 44 1B      0A9F
44 1B 6A 44 1B 44 1B 6B 44 1B 44 1B   0AA2    600          .ASCII  <ESC>/D/<ESC>/Dk/<ESC>/D/<ESC>/Dj/<ESC>/D/<ESC>/Di/
                        69 44 1B      0AAE
44 1B 67 44 1B 44 1B 68 44 1B 44 1B   0AB1    601          .ASCII  <ESC>/D/<ESC>/Dh/<ESC>/Dg/<ESC>/D/<ESC>/Df/
                        66 44 1B      0ABD
44 1B 64 44 1B 44 1B 65 44 1B 44 1B   0AC0    602          .ASCII  <ESC>/D/<ESC>/De/<ESC>/D/<ESC>/Dd/<ESC>/D/<ESC>/Dc/
                        63 44 1B      0ACC
59 1B 61 44 1B 44 1B 62 44 1B 44 1B   0ACF    603          .ASCII  <ESC>/D/<ESC>/Db/<ESC>/D/<ESC>/Da/<ESC>/Y/<33><54>
                              36 21   0ADB
1B 44 1B 6D 41 1B 73 42 1B 44 1B 6C   0ADD    604          .ASCII  /l/<ESC>/D/<ESC>/Bs/<ESC>/Am/<ESC>/D/<ESC>/Br/
                              72 42   0AE9
1B 6F 41 1B 71 44 1B 42 1B 6E 41 1B   0AEB    605          .ASCII  <ESC>/An/<ESC>/B/<ESC>/Dq/<ESC>/Ao/<ESC>/B/<ESC>/Dp/
                           70 44 1B 42  0AF7
1B 71 41 1B 6F 44 1B 42 1B 70 41 1B   0AFB    606          .ASCII  <ESC>/Ap/<ESC>/B/<ESC>/Do/<ESC>/Aq/<ESC>/B/<ESC>/Dn/
                           6E 44 1B 42  0B07
1B 73 41 1B 6D 44 1B 42 1B 72 41 1B   0B0B    607          .ASCII  <ESC>/Ar/<ESC>/B/<ESC>/Dm/<ESC>/As/<ESC>/B/<ESC>/Dl/
```

```
                                                     6C 44 1B 42  0B17
1B 71 41 1B 6D 44 1B 42                              1B 72 41 1B  0B1B    608          .ASCII  <ESC>/Ar/<ESC>/B/<ESC>/Dm/<ESC>/Aq/<ESC>/B/<ESC>/Dn/
                                                     6E 44 1B 42  0B27
1B 6F 41 1B 6F 44 1B 42                              1B 70 41 1B  0B2B    609          .ASCII  <ESC>/Ap/<ESC>/B/<ESC>/Do/<ESC>/Ao/<ESC>/B/<ESC>/Dp/
                                                     70 44 1B 42  0B37
1B 6D 41 1B 71 44 1B 42                              1B 6E 41 1B  0B3B    610          .ASCII  <ESC>/An/<ESC>/B/<ESC>/Dq/<ESC>/Am/<ESC>/B/<ESC>/Dr/
                                                     72 44 1B 42  0B47
62 61 41 1B 73 44 1B 42                              1B 6C 41 1B  0B4B    611          .ASCII  <ESC>/Al/<ESC>/B/<ESC>/Ds/<ESC>/Aabcdefghijktuvwxyz{|}~/
76 75 74 6B 6A 69 68 67 66 65 64 63                              0B57
                        7E 7D 7C 7B 7A 79 78 77                  0B63
68 67 66 65 64 63 62 61 45 22 59 1B                              0B6B    612          .ASCII  <FSC>/Y/<34><69>/abcdefghijktuvwxyz{|}~/<ESC>/Aabcdefghijktuvwxyz{|}
7C 7B 7A 79 78 77 76 75 74 6B 6A 69                              0B77
68 67 66 65 64 63 62 61 41 1B 7E 7D                              0B83
7C 7B 7A 79 78 77 76 75 74 6B 6A 69                              0B8F
                                    7E 7D                        0B9B
68 67 66 65 64 63 62 61 5B 22 59 1B                              0B9D    613          .ASCII  <ESC>/Y/<34><91>/abcdefghijktuvwxyz{|}~/
7C 7B 7A 79 78 77 76 75 74 6B 6A 69                              0BA9
                                    7E 7D                        0BB5
                                  0A 0D 47 1B                    0BB7    614          .ASCII  <ESC>/G/<CR><LF>        ; Exit graphics mode
                                     00000168                    0BBB    615          VT52L = .-VT52_OUT_DATA
                                                                 0BBB    616 ;
                                                                 0BBB    617 ; Terminal specific data for the VT05 terminal
                                                                 0BBB    618 ;
                                                                 0BBB    619 VT05_PREAMBLE:
                                                           05'   0BBB    620          .BYTE   VT05_PRE_LEN
                                                  1F 00 00 00 1D 0BBC    621          .BYTE   29,0,0,0,31
                                                     00000005    0BC1    622          VT05_PRE_LEN = .-VT05_PREAMBLE-1
                                                                 0BC1    623
                                                                 0BC1    624 VT05_DATA:
                                                     00000000    0BC1    625          .LONG   0                       ; There is no VT05 terminal specific data
                                                     00000BC1'   0BC5    626          .ADDRESS VT05_DATA
                                                                 0BC9    627 ;
                                                                 0BC9    628 ; Terminal specific data for the LA120 terminal
                                                                 0BC9    629 ;
                                                                 0BC9    630 LA120_DATA:
                                                     00000000    0BC9    631          .LONG   0                       ; There is no LA120 terminal specific data
                                                     00000BC9'   0BCD    632          .ADDRESS LA120_DATA
                                                                 0BD1    633 ;
                                                                 0BD1    634 ; Terminal specific data for the LA36 terminal
                                                                 0BD1    635 ;
                                                                 0BD1    636 LA36_DATA:
                                                     00000000    0BD1    637          .LONG   0                       ; There is no LA36 terminal specific data
                                                     00000BD1'   0BD5    638          .ADDRESS LA36_DATA
                                                                 0BD9    639 ;
                                                                 0BD9    640 ; Terminal specific data for the LA180 terminal
                                                                 0BD9    641 ;
                                                                 0BD9    642 LA180_DATA:
                                                     00000000    0BD9    643          .LONG   0                       ; There is no LA180 terminal specific data
                                                     00000BD9'   0BDD    644          .ADDRESS LA180_DATA
                                                                 0BE1    645 ;
                                                                 0BE1    646 ; Terminal specific data for the LA11 terminal
                                                                 0BE1    647 ;
                                                                 0BE1    648 LA11_DATA:
                                                     00000000    0BE1    649          .LONG   0                       ; There is no LA11 terminal specific data
                                                     00000BE1'   0BE5    650          .ADDRESS LA11_DATA
                                                                 0BE9    651 ;
                                                                 0BE9    652 ; Terminal specific data for the an UNKNOWN terminal
```

```
                    0BE9   653 ;
                    0BE9   654 UNKN_PREAMBLE:
                 00 0BE9   655          .BYTE   0
                    0BEA   656
                    0BEA   657 UNKN_DATA:
00000000            0BEA   658          .LONG   0                       ; There is no UNKNOWN terminal
00000BC9'           0BEE   659          .ADDRESS LA120_DATA             ; specific data
                    0BF2   660 ;
                    0BF2   661 ; Terminal specific data for the LP11 terminal
                    0BF2   662 ;
                    0BF2   663 LP11_DATA:
00000000            0BF2   664          .LONG   0                       ; There is no LP11 terminal specific data
00000BF2'           0BF6   665          .ADDRESS LP11_DATA
                    0BFA   666 ;
                    0BFA   667 ; Terminal specific data for the LAX terminal
                    0BFA   668 ;
                    0BFA   669 LAX_DATA:
00000000            0BFA   670          .LONG   0                       ; There is no LAX terminal specific data
00000BFA'           0BFE   671          .ADDRESS LAX_DATA
                    0C02   672 FORM_FEED:
                 01 0C02   673          .BYTE   1                       ; Form feed ASCIC string
                 0C 0C03   674          .BYTE   FF
                    0C04   675
                    0C04   676 CRLF:                                    ; Carriage return and Line feed
              0A 0D 0C04   677          .BYTE   CR,LF
                    0C06   678 LFCR:                                    ; Line feed and Carriage return
              0D 0A 0C06   679          .BYTE   LF,CR
```

```
                                0C08   681          .SBTTL  Read/Write Data
                          00000000     682          .PSECT  RWDATA,WRT,NOEXE,PAGE
                                0000   683
                                0000   684 TTCHAN:                                  ; Channel associated with ctrl. term.
                          0000  0000   685          .WORD   0
                                0002   686
                                0002   687 FLAG:                                    ; Miscellaneous flag bits
                          0000  0002   688          .WORD   0                       ; (See Equated Symbols for definitions)
                                0004   689
                                0004   690 FAO_BUF:                                 ; FAO output string descriptor
                     0000 0084  0004   691          .WORD   TEXT_BUFFER,0
                      00000014' 0008   692          .ADDRESS BUFFER
                                000C   693
                                000C   694 BUFFER_PTR:                             ; Fake .ASCID buffer for misc. strings
                     0000 0084  000C   695          .WORD   TEXT_BUFFER,0           ; A word for length, a word for desc.
                      00000014' 0010   696          .ADDRESS BUFFER
                                0014   697
                                0014   698 BUFFER:                                 ; FAO output and other misc. buffer
                      00000098  0014   699          .BLKB   TEXT_BUFFER
                                0098   700
                                0098   701 DEVDSC:                                 ; Device name descriptor
                     0000 000A  0098   702          .WORD   MAX_DEV_DESIG,0
                      000000B7' 009C   703          .ADDRESS DEV_NAME
                                00A0   704
                                00A0   705 PROCESS_NAME:                           ; Process name
 53 59 54 54 000000A8 010E0000' 00A0   706          .ASCID  /TTYS/
                           0000000B  00AC   707          PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
                           000000B7  00AC   708          .BLKB   PROCESS_NAME_FREE
                                00B7   709
                                00B7   710 DEV_NAME:                               ; Device name buffer
                           000000C6  00B7   711          .BLKB   MAX_DEV_DESIG+MAX_UNIT_DESIG
                           0000000F  00C6   712          NAME_LEN = .-DEV_NAME
                                00C6   713
                                00C6   714 DATA_DSC:                               ; Descriptor for STR$UPCASE
                           00000101  00C6   715          .LONG   WRITE_SIZE
                           00000000' 00CA   716          .ADDRESS 0
                                00CE   717 DIB:                                    ; Device Information Block
                     0000 0074  00CE   718          .WORD   DIB$K_LENGTH,0
                      000000D6' 00D2   719          .ADDRESS DIBBUF
                                00D6   720 DIBBUF:
                           0000014A  00D6   721          .BLKB   DIB$K_LENGTH
                                014A   722
                                014A   723 DIB_SEC:                                ; Device Information Block (secondary)
                     0000 0074  014A   724          .WORD   DIB$K_LENGTH,0
                      00000152' 014E   725          .ADDRESS DIBBUF_SEC
                                0152   726 DIBBUF_SEC:
                           000001C6  0152   727          .BLKB   DIB$K_LENGTH
                                01C6   728
                                01C6   729 ERROR_COUNT:                            ; Cumulative error count at runtime
                           00000000  01C6   730          .LONG   0
                                01CA   731
                                01CA   732 STATUS:                                 ; Status value on program exit
                           00000000  01CA   733          .LONG   0
                                01CE   734
                                01CE   735 QUAD_STATUS:                            ; IO status block for misc sys. svcs.
 00000000 00000000              01CE   736          .QUAD   0
                                01D6   737
```

```
                    01D6   738 INADDRESS:                                ; $CRMPSC address storage
00000000 00000000   01D6   739         .LONG   0,0
                    01DE   740 OUTADDRESS:
00000000 00000000   01DE   741         .LONG   0,0
                    01E6   742
                    01E6   743 UNIT_NUMBER:                             ; Current dev unit number
            0000    01E6   744         .WORD   0
                    01E8   745
                    01E8   746 DEVNAM_LEN:                              ; Current device name length
            0000    01E8   747         .WORD   0
                    01EA   748
                    01EA   749 RANDOM1:                                 ; Random word #1
        AAAAAAAA    01EA   750         .LONG   ^XAAAAAAAA
                    01EE   751
                    01EE   752 RANDOM2:                                 ; Random word #2
        A72EA72E    01EE   753         .LONG   ^XA72EA72E
                    01F2   754
                    01F2   755 ITERATION:                               ; # of times all tests were executed
        00000000    01F2   756         .LONG   0
                    01F6   757
                    01F6   758 MSG_BLOCK:                               ; Auxiliary $GETMSG info
        000001FA    01F6   759         .BLKB   4
                    01FA   760
                    01FA   761 PASS:                                    ; Pass count
        00000000    01FA   762         .LONG   0
                    01FE   763
                    01FE   764 EXIT_DESC:                               ; Exit handler descriptor
        00000000    01FE   765         .LONG   0
        00000E15'   0202   766         .ADDRESS EXIT_HANDLER
        00000001    0206   767         .LONG   1
        000001CA'   020A   768         .ADDRESS STATUS
                    020E   769
                    020E   770 ARG_COUNT:                               ; Argument counter used by ERROR_EXIT
        00000000    020E   771         .LONG   0
                    0212   772
                    0212   773 RMSRUNDWN_BUF:                   ; Return buf for RMSRUNDWN close failures
       0000 0016    0212   774         .WORD   22,0
        0000021A'   0216   775         .ADDRESS RUNDWN_BUF
                    021A   776
                    021A   777 RUNDWN_BUF:
        00000230    021A   778         .BLKB 22
                    0230   779
                    0230   780 ;
                    0230   781 ; Head of self-relative UETP unit block queue.
                    0230   782 ;
                    0230   783         .ALIGN QUAD
                    0230   784
                    0230   785 UNIT_LIST:                                ; Head of unit block circular list
00000000 00000000   0230   786         .QUAD   0
                    0238   787
                    0238   788 NEW_NODE:                                ; Newly aquired node address
00000000 00000000   0238   789         .QUAD   0
                    0240   790
                    0240   791 HEAD_BUF:                                ; Buffer descriptor for the header message
        00000084    0240   792         .LONG   132
        00000000'   0244   793         .ADDRESS 0
                    0248   794 HEAD_LENGTH:                            ; Length of header record
```

```
00000000  0248   795          .LONG   0
          024C   796 SPARE:
       00 024C   797          .BYTE   0                ; Spot used to store byte while rotating dat
          024D   798 CUR_UNTBLK:                      ; Memory storage for the current unit block
00000000  024D   799          .LONG   0
          0251   800
          0251   801 XTERM_CHAR:                      ; Extended terminal characteristics
00000000  0251   802          .LONG
```

```
                     804          .SBTTL  RMS-32 Data Structures
                     805          .ALIGN  LONG
                     806
                     807 SYSIN_FAB:                                  ; Allocate FAB for SYS$INPUT
                     808          $FAB-
                     809          FNM = <SYS$INPUT>
                     810
                     811 SYSIN_RAB:                                  ; Allocate RAB for SYS$INPUT
                     812          $RAB-
                     813          FAB = SYSIN_FAB,-
                     814          ROP = PMT,-
                     815          PBF = PROMPT,-
                     816          PSZ = PMTSIZ,-
                     817          UBF = DEV_NAME,-
                     818          USZ = NAME_LEN
                     819
                     820 INI_FAB:                                    ; Allocate FAB for UETINIDEV
                     821          $FAB-
                     822          FAC = <GET,PUT,UPD>,-
                     823          RAT = CR,-
                     824          SHR = <GET,PUT,UPI>,-
                     825          FNM = <UETINIDEV.DAT>
                     826
                     827 INI_RAB:                                    ; Allocate RAB for UETINIDEV
                     828          $RAB-
                     829          FAB = INI_FAB,-
                     830          RBF = BUFFER,-
                     831          UBF = BUFFER,-
                     832          USZ = REC_SIZE
                     833
                     834 DDC_RFA:                                    ; RFA storage for INI_RAB
00000386             835          .BLKB   6
                     836
                     837          .ALIGN  LONG
                     838 SUP_FAB:                                    ; Allocate FAB for UETSUPDEV
                     839          $FAB-
                     840          FAC = GET,-
                     841          SHR = <UPI,GET>,-
                     842          RAT = CR,-
                     843          FOP = UFO,-
                     844          FNM = <UETSUPDEV.DAT>
                     845
                     846 ;
                     847 ; Dummy FAB and RAB to copy to the UETP unit blocks
                     848 ; The following FAB and RAB must be contiguous and in this order!
                     849 ;
                     850
                     851 DUMMY_FAB:
                     852          $FAB    RAT = <PRN>,-
                     853                  RFM = <VFC>,-
                     854                  FAC = <BRO,PUT>
                     855
                     856 DUMMY_RAB:
                     857          $RAB    RSZ = WRITE_SIZE,-
                     858                  ROP = <ASY,BIO>,-
                     859                  RHB = 0
```

```
                     046C   861          .SBTTL  Main Program
                 00000000   862          .PSECT  TTYS,EXE,NOWRT,PAGE
                     0000   863
                     0000   864          .DEFAULT DISPLACEMENT,WORD
                     0000   865
                     0000   866 ;+
                     0000   867 ;       Start up the terminal test.  This entails some overhead necessary to
                     0000   868 ;       cope with both expected and unforseen conditions, figuring out just
                     0000   869 ;       what devices are to be tested, making sure we can test the indicated
                     0000   870 ;       devices and setting up writeable space for each device to be tested.
                     0000   871 ;-
                     0000   872
                0000 0000   873 .ENTRY UETTTYS00,^M<>                       ; Entry mask
                     0002   874
          6D  0C0F'CF  DE   0002   875          MOVAL   SSERROR,(FP)          ; Declare exception handler
                     0007   876          $SETSFM_S ENBFLG = #1              ; Enable system service failure mode
                     0010   877          $DCLEXH_S DESBLK = EXIT_DESC       ; Declare an exit handler
                     001B   878
                     001B   879          $OPEN   FAB = SYSIN_FAB,-          ; Open SYS$INPUT
                     001B   880                  ERR = RMS_ERROR
                     002A   881          $CONNECT RAB = SYSIN_RAB,-         ; Connect RAB to SYS$INPUT
                     002A   882                   ERR = RMS_ERROR
                  02 E1   0039   883          BBC     S^#DEV$V_TRM,-         ; BR if SYS$INPUT is NOT a terminal
             3F 0298'CF   003B   884                  SYSIN_FAB+FAB$L_DEV,10$
                     003F   885          $GETDVI_S DEVNAM = SYS$INPUT,-     ; Get the name of the device...
                     003F   886                    EFN    = #SS_SYNCH_EFN,- ; ...from which the test is run.
                     003F   887                    ITMLST = INPUT_ITMLST,-  ; We have a special need here,...
                     003F   888                    IOSB   = QUAD_STATUS     ; ...get it now & save it in BUFFER
             5C 01CE'CF  E9   005B   889          BLBC    QUAD_STATUS,20$       ; Abort if we can't get it
                     0060   890          $TRNLOG_S LOGNAM = CONTROLLER,- ; Allow terminal user to specify...
                     0060   891                    RSLLEN = DEVNAM_LEN,- ; ...a logial name...
                     0060   892                    RSLBUF = DEVDSC       ; ...for the controller to test
                01   50  D1   0079   893          CMPL    R0,#SS$_NORMAL        ; Was a controller specified?
                     56  13   007C   894          BEQL    PROC_CONT_NAME        ; BR if it was - go process it
                         007E   895 10$:
                         007E   896          $GET    RAB = SYSIN_RAB,-     ; Read SYS$INPUT...
                         007E   897                  ERR = RMS_ERROR       ; ...for the controller name
                02CA'CF  B0   008D   898          MOVW    SYSIN_RAB+RAB$W_RSZ,- ; Save the name length
                01E8'CF       0091   899                  DEVNAM_LEN
                     3E  12   0094   900          BNEQ    PROC_CONT_NAME        ; BR if we got something
            01CE'CF  14  D0   0096   901          MOVL    #SS$_BADPARAM,QUAD_STATUS ; Set up default if we BR below
                     02  E1   009B   902          BBC     S^#DEV$V_TRM,-        ; BR if SYS$INPUT is NOT a terminal
              1B 0298'CF   009D   903                  SYSIN_FAB+FAB$L_DEV,20$
            0002'CF  08  88   00A1   904          BISB2   #SELF_TESTM,FLAG      ; Set self test mode
          000C'CF  5F 8F  3B   00A6   905          SKPC    #^A/_/,BUFFER_PTR,-   ; Strip off leading underscores
                0014'CF       00AC   906                  BUFFER
            01E8'CF  50  B0   00AF   907          MOVW    R0,DEVNAM_LEN         ; Save the device name length
        00B7'CF  61  50  28   00B4   908          MOVC3   R0,(R1),DEV_NAME      ; Use physical terminal name
                     18  11   00BA   909          BRB     PROC_CONT_NAME        ; Continue with testing
                         00BC   910 20$:
        01CA'CF  01CE'CF  B0   00BC   911          MOVW    QUAD_STATUS,STATUS    ; Save the exit status
                00D4'CF  DF   00C3   912          PUSHAL  NO_CTRLNAME           ; Prepare for message...
                     01  DD   00C7   913          PUSHL   #1                    ; ...arg count
              00741132  8F  DD   00C9   914          PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...signal name
                     03  DD   00CF   915          PUSHL   #3                    ; ...arg count
                   0CC1  31   00D1   916          BRW     ERROR_EXIT            ; ...go tell of bad setup
                         00D4   917
```

```
                                    00D4   918 PROC_CONT_NAME:
    0098'CF    01E8'CF    3C    00D4   919          MOVZWL  DEVNAM_LEN,DEVDSC        ; Set the device name length
               0098'CF    DF    00DB   920          PUSHAL  DEVDSC                  ; Make sure...
               0098'CF    DF    00DF   921          PUSHAL  DEVDSC                  ; ...that the specified controller...
     00000000'GF    02    FB    00E3   922          CALLS   #2,G^STR$UPCASE         ; ...is all uppercase for later comaparison
52   0098'CF    01    C1    00EA   923          ADDL3   #1,DEVDSC,R2            ; Estimate the eventual...
     00A0'CF    52    A0    00F0   924          ADDW2   R2,PROCESS_NAME         ; ...process name length (incl. "_")
                      DE    00F5   925          MOVAL   PROCESS_NAME+8-         ; Locate first available byte...
                              00F6   926                  +MAX_PROC_NAME-         ; ...in process name handle...
          50   00AC'CF            00F6   927                  -PROCESS_NAME_FREE,R0   ; ...for device name
                      0B    C3    00FA   928          SUBL3   #PROCESS_NAME_FREE,-    ; Will the device name fit...
               51    52           00FC   929                  R2,R1                   ; ...in the remaining space?
                      08    15    00FE   930          BLEQ    10$                     ; BR if it will
               50    51    C2    0100   931          SUBL2   R1,R0                   ; Overwrite handle otherwise...
     00A0'CF    0F    B0    0103   932          MOVW    #MAX_PROC_NAME,PROCESS_NAME ; ...and define the maximum length
                              0108   933 10$:
          80    5F    8F    90    0108   934          MOVB    #^A/ /,(R0)+            ; Separate handle from device name
60   00B7'CF    0098'CF    28    010C   935          MOVC3   DEVDSC,DEV_NAME,(R0)    ; Concatenate handle with device name
                     7E    D4    0114   936          CLRL    -(SP)                   ; Set the time stamp flag
               000F'CF    DF    0116   937          PUSHAL  TEST_NAME               ; Set the test name
                      02    DD    011A   938          PUSHL   #2                      ; Push the argument count
          00741039    8F    DD    011C   939          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
     00000000'GF    04    FB    0122   940          CALLS   #4,G^LIB$SIGNAL         ; Print the startup message
          0002'CF    10    A8    0129   941          BISW2   #BEGIN_MSGM,FLAG        ; Set flag so we don't print it again
                              012E   942          $SETPRN_S PRCNAM = PROCESS_NAME ; Set the process name to UETTTYS00_x
                              0139   943
                      02    E1    0139   944          BBC     S^#DEV$V_TRM,-          ; BR if SYS$INPUT is NOT a terminal
          79 0298'CF         013B   945                  SYSIN_FAB+FAB$L_DEV,20$
                              013F   946          $ASSIGN_S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C ASTs if we are
                              013F   947                    CHAN   = TTCHAN
                              0150   948          $QIOW_S CHAN     = TTCHAN,-     ; Enable CTRL/C AST's...
                              0150   949                  FUNC     = #IO$_SETMODE!IO$M_CTRLCAST,-
                              0150   950                  P1       = CCASTHAND
               00A0'CF    DF    0171   951          PUSHAL  PROCESS_NAME            ; ...and tell the user...
                      01    DD    0175   952          PUSHL   #1                      ; ...
          0074832B    8F    DD    0177   953          PUSHL   #UETP$_ABORTC!STS$K_SUCCESS ; ...how to abort gracefully...
     00000000'GF    03    FB    017D   954          CALLS   #3,G^LIB$SIGNAL         ; ...
     2E 0002'CF    03    E1    0184   955          BBC     #SELF_TESTV,FLAG,20$    ; Skip this if not self test
                              018A   956          $GETDEV_S DEVNAM = DEVDSC,-
                              018A   957                    PRIBUF = DIB          ; Get the device characteristics
                    05CF    30    019F   958          BSBW    GET_NODE                ; Get a unit block and init it
                              01A2   959 ;
                              01A2   960 ; Set the device dependent parameters
                              01A2   961 ;
               00D6'CF    DF    01A2   962          PUSHAL  DIBBUF                  ; Push the device characteristics buffer
7E 0230'CF    00000230'8F    C1    01A6   963          ADDL3   #UNIT_LIST,UNIT_LIST,-(SP) ; Push the unit block address
          07D4'CF    02    FB    01B0   964          CALLS   #2,SET_DEVDEP           ; Set the device characteristics
                    0265    31    01B5   965          BRW     ALL_SET                 ; Skip useless checking
                              01B8   966 20$:
```

```
                                01B8   968 ;
                                01B8   969 ; From UETINIDEV.DAT and UETSUPDEV.DAT, get information which gives controller
                                01B8   970 ; and unit configuration and lets us know if the setup to run this test was
                                01B8   971 ; done correctly.
                                01B8   972 ;
                                01B8   973         $OPEN   FAB = INI_FAB,-          ; Open file 'UETINIDEV.DAT'
                                01B8   974                 ERR = RMS_ERROR
                                01C7   975         $CONNECT RAB = INI_RAB,-         ; Connect the RAB and FAB
                                01C7   976                  ERR = RMS_ERROR
                                01D6   977         $MGBLSC_S  INADR = INADDRESS,-   ; Connect to UETSUPDEV global section
                                01D6   978                    RETADR = OUTADDRESS,-
                                01D6   979                    GSDNAM = SUPDEV_GBLSEC,-
                                01D6   980                     FLAGS = #SEC$M_EXPREG
 00000978 8F    50    D1       01F5   981         CMPL    R0,#SS$_NOSUCHSEC        ; Was the section already there?
                37    12       01FC   982         BNEQ    30$                      ; BR if it was...
                                01FE   983         $OPEN   FAB = SUP_FAB,-          ; ...else open 'UETSUPDEV.DAT'
                                01FE   984                 ERR = RMS_ERROR
                                020D   985         $CRMPSC_S CHAN = SUP_FAB+FAB$L_STV,- ; Create the global section
                                020D   986                   INADR = INADDRESS,-
                                020D   987                  RETADR = OUTADDRESS,-
                                020D   988                  GSDNAM = SUPDEV_GBLSEC,-
                                020D   989                   FLAGS = #SEC$M_EXPREG!SEC$M_GBL
                                0235   990 30$:
   56   01E2'CF   01DE'CF    C3  0235   991         SUBL3   OUTADDRESS,OUTADDRESS+4,R6 ; Compute global section length
                                023D   992
                                023D   993 FIND_IT:
                                023D   994         $GET    RAB = INI_RAB,-          ; Get the first record
                                023D   995                 ERR = RMS_ERROR
             02BF'CF    DF      024C   996         PUSHAL  CONT_DESC                ; Make sure...
             02BF'CF    DF      0250   997         PUSHAL  CONT_DESC                ; ...that the controller name...
 00000000'GF    02    FB      0254   998         CALLS   #2,G^STR$UPCASE          ; ...is all uppercase letters
  0014'CF    44 8F    91      025B   999         CMPB    #^A/D/,BUFFER            ; Is this a DDB?
                27    13      0261  1000         BEQL    10$                      ; Go on if not
  0014'CF    45 8F    91      0263  1001         CMPB    #^A/E/,BUFFER            ; Is this the end of the file?
                D2    12      0269  1002         BNEQ    FIND_IT                  ; Continue on if not
          0098'CF    DF      026B  1003         PUSHAL  DEVDSC                   ; Push device not supported message
          00A0'CF    DF      026F  1004         PUSHAL  PROCESS_NAME             ; Parameters on the stack
                02    DD      0273  1005         PUSHL   #2
       00748333 8F    DD      0275  1006         PUSHL   #UETP$_DENOSU
                02    F0      027B  1007         INSV    #STS$K_ERROR,-           ; Set the severity code...
                00            027D  1008                 #STS$V_SEVERITY,-
             6E    03         027E  1009                 #STS$S_SEVERITY,(SP)
       01CA'CF    6E    D0      0280  1010         MOVL    (SP),STATUS              ; ...and save it as the exit status
                04    DD      0285  1011         PUSHL   #4
              0B0B    31      0287  1012         BRW     ERROR_EXIT               ; Exit in error
                            028A  1013 10$:
00B7'CF 001A'CF   01E8'CF    29  028A  1014         CMPC    DEVNAM_LEN,BUFFER+6,DEV_NAME ; Is this the right controller?
                A7    12      0294  1015         BNEQ    FIND_IT                  ; BR if not
     0380'CF   034C'CF    06  0296  1016         MOVC3   #6,INI_RAB+RAB$W_RFA,DDB_RFA ; Save the Record File Address
          0018'CF    54 8F    91  029E  1017         CMPB    #^A/T/,BUFFER+4          ; Can we test this controller?
                2F    13      02A4  1018         BEQL    FOUND_IT                 ; BR if we can...
                            02A6  1019         $FAO_S  CTRSTR = DEAD_CTRLNAME,- ; ...and yell at user if we can't
                            02A6  1020                 OUTLEN = BUFFER_PTR,-
                            02A6  1021                 OUTBUF = FAO_BUF,-
                            02A6  1022                 P1     = #DEVDSC
       01CA'CF    14    D0      02BF  1023         MOVL    #SS$_BADPARAM,STATUS     ; Set return status
             000C'CF    DF      02C4  1024         PUSHAL  BUFFER_PTR               ; ...
```

```
                 01   DD  02C8  1025          PUSHL   #1                              ; ...
          00741132 8F   DD  02CA  1026          PUSHL   #UETP$_TEXT!STS$K_ERROR         ; ...
                 03   DD  02D0  1027          PUSHL   #3                              ; ...
               0AC0   31  02D2  1028          BRW     ERROR_EXIT                      ; We can't test what we can't test
                      02D5  1029
                      02D5  1030 FOUND_IT:
                      02D5  1031          $GET    RAB = INI_RAB,-                 ; Get a record
                      02D5  1032                  ERR = RMS_ERROR
            02BF'CF   DF  02E4  1033          PUSHAL  CONT_DESC                       ; Make sure...
            02BF'CF   DF  02E8  1034          PUSHAL  CONT_DESC                       ; ...that this line...
00000000'GF   02   FB  02EC  1035          CALLS   #2,G^STR$UPCASE                 ; ...is all uppercase letters
 0014'CF   55 8F   91  02F3  1036          CMPB    #^A/U/,BUFFER                   ; Is this a UCB?
                24   13  02F9  1037          BEQL    30$                             ; BR if it is
 0014'CF   44 8F   91  02FB  1038          CMPB    #^A/D/,BUFFER                   ; Is this a DDB?
                19   13  0301  1039          BEQL    20$                             ; BR if yes
 0014'CF   45 8F   91  0303  1040          CMPB    #^A/E/,BUFFER                   ; Is this the end?
                11   13  0309  1041          BEQL    20$                             ; BR if yes
                      030B  1042 10$:
            0161'CF   DF  030B  1043          PUSHAL  ILLEGAL_REC                     ; Then this is an error in the record
                 01   DD  030F  1044          PUSHL   #1                              ; Push the error message
          00741132 8F   DD  0311  1045          PUSHL   #UETP$_TEXT!STS$K_ERROR         ; Push the signal name
                 03   DD  0317  1046          PUSHL   #3                              ; Push the temp arg count
               0A79   31  0319  1047          BRW     ERROR_EXIT                      ; Finish for good
                      031C  1048 20$:
               00FE   31  031C  1049          BRW     ALL_SET                         ; Found DDB or END
                      031F  1050 30$:
 0018'CF   54 8F   91  031F  1051          CMPB    #^A/T/,BUFFER+4                 ; Is the unit testable?
                AE   12  0325  1052          BNEQ    FOUND_IT                        ; BR if not
                 01   DD  0327  1053          PUSHL   #1                              ; Flag to ignore blanks when converting
                 02   DD  0329  1054          PUSHL   #2                              ; Set byte size of results
            01E6'CF   DF  032B  1055          PUSHAL  UNIT_NUMBER                     ; Set address to receive word
            02B7'CF   DF  032F  1056          PUSHAL  UNIT_DESC                       ; Push string address
      00000000'GF   04   FB  0333  1057          CALLS   #4,G^OTS$CVT_TI_L               ; Convert ASCII unit # to decimal
                CE 50   E9  033A  1058          BLBC    R0,10$                          ; Don't allow bogus unit to pass
            05   20   3B  033D  1059          SKPC    #^A/ /,#MAX_UNIT_DESIG,-        ; Find out where unit number really is
            001A'CF       0340  1060                  BUFFER+6
                 50   D7  0343  1061          DECL    R0                              ; Units must all be at least one digit
        61   50   30   3B  0345  1062          SKPC    #^A/0/,R0,(R1)                  ; Skip leading zeroes on the unit
                 50   D6  0349  1063          INCL    R0                              ; Compensate for DECL above
0098'CF   01E8'CF   50   A1  034B  1064          ADDW3   R0,DEVNAM_LEN,DEVDSC            ; Calculate device'unit string length
      52   01E8'CF   3C  0353  1065          MOVZWL  DEVNAM_LEN,R2                   ; Offset to unit number in DEVDSC
00B7'C2   61   50   28  0358  1066          MOVC3   R0,(R1),DEV_NAME(R2)            ; Append unit number to device
                      035E  1067          $GETDEV_S DEVNAM = DEVDSC,-             ; Get the device characteristics
                      035E  1068                    PRIBUF = DIB,-
                      035E  1069                    SCDBUF = DIB_SEC
                 06   E1  0375  1070          BBC     #DEV$V_SPL,-
          0E 0152'CF       0377  1071                  DIB$L_DEVCHAR+DIBBUF_SEC,35$    ; Check for the spool bit
                 08   BB  037B  1072          PUSHR   #^M<R3>                         ; Save pointer to end of device name
      0152'CF    0074 8F   28  037D  1073          MOVC3   #DIB$K_LENGTH,DIBBUF_SEC,- ; Use secondary buffer if spooled
            00D6'CF       0384  1074                  DIBBUF
                 08   BA  0387  1075          POPR    #^M<R3>                         ; Restore pointer to end of device
            0098'CF   D6  0389  1076 35$:      INCL    DEVDSC                          ; Don't forget the : on the end
            63   3A   90  038D  1077          MOVB    #^A/:/,(R3)                     ; Stick in the colon
        57   00DA'CF   9A  0390  1078          MOVZBL  DIBBUF+DIB$B_DEVCLASS,R7        ; Save the device class
        58   00DB'CF   9A  0395  1079          MOVZBL  DIBBUF+DIB$B_DEVTYPE,R8         ; Save the device type
                      039A  1080          $FAO_S  CTRSTR = CS1,-
                      039A  1081                  OUTBUF = FAO_BUF,-
```

```
                                039A  1082                P1      = R7,-
                                039A  1083                P2      = R8                    ; Make it into a string
01DE'DF    56  0014'CF  06   39  03AF  1084        MATCHC  #6,BUFFER,R6,@OUTADDRESS ; Find the device class and type
                        1E   13  03B8  1085        BEQL    40$                     ; BR if it was found
                                03BA  1086        $FAO_S  CTRSTR = CS3,-          ; Try for full class support
                                03BA  1087                OUTBUF = FAO_BUF,-
                                03BA  1088                  P1 = R7
01DE'DF    56  0014'CF  06   39  03CD  1089        MATCHC  #6,BUFFER,R6,@OUTADDRESS ; Find the device class only
                        0D   12  03D6  1090        BNEQ    50$                     ; BR if not found
                                03D8  1091 40$:
           55  000F'CF  9A   03D8  1092        MOVZBL  TEST_NAME,R5            ; Get the test name length
0017'CF    63  55       29   03DD  1093        CMPC3   R5,(R3),TEST_NAME+8     ; Are we the right test?
                        1F   13  03E3  1094        BEQL    60$                     ; BR if yes
                                03E5  1095 50$:
                0098'CF  DF   03E5  1C96        PUSHAL  DEVDSC                  ; Push device not supported message
                00A0'CF  DF   03E9  1097        PUSHAL  PROCESS_NAME            ; Parameters on the stack
                        02   DD  03ED  1098        PUSHL   #2                      ; Push the argument count
             00748333   8F   DD  03EF  1099        PUSHL   #UETP$_DENOSU
                        02   F0  03F5  1100        INSV    #STS$K_ERROR,-
                        00       03F7  1101                #STS$V_SEVERITY,-
                    6E  03       03F8  1102                #STS$S_SEVERITY,(SP)    ; Set the severity code...
                01CA'CF  6E   D0  03FA  1103        MOVL    (SP),STATUS             ; ...and save it as the exit status
                        04   DD  03FF  1104        PUSHL   #4                      ; Push the partial arg count...
                    0991   31   0401  1105        BRW     ERROR_EXIT              ; ...and split this scene
                                0404  1106 60$:
                    0040 8F   BB   0404  1107        PUSHR   #^M<R6>                 ; Save across GET_NODE & SET_DEVDEP
                       0366   30   0408  1108        BSBW    GET_NODE                ; Get a unit block and init it
                                040B  1109 ;
                                040B  1110 ; Set the device dependent parameters
                                040B  1111 ;
                    00D6'CF    DF   040B  1112        PUSHAL  DIBBUF                  ; Push device information block adr
                        56     DD   040F  1113        PUSHL   R6                      ; Push unit block address
                07D4'CF    02   FB   0411  1114        CALLS   #2,SET_DEVDEP           ; Set the device dependent stuff
                    0040 8F    BA   0416  1115        POPR    #^M<R6>                 ; Saved across GET_NODE & SET_DEVDEP
                       FEB8    31   041A  1116        BRW     FOUND_IT                ; Do the next UCB
```

```
                     041D  1118 ;
                     041D  1119 ; Arrive here when we have the device configuration.  In normal or loop forever
                     041D  1120 ; mode, set a timer far enough in the future such that we can do a reasonable
                     041D  1121 ; set of tests before the timer expires, but if our device gets hung, the
                     041D  1122 ; program won't waste too much time before noticing.  Let one-shot mode be a
                     041D  1123 ; special case.
                     041D  1124 ;
                     041D  1125 ALL_SET:
          0230'CF    D5  041D  1126         TSTL    UNIT_LIST                       ; Anything to test?
                16   12  0421  1127         BNEQ    10$                             ; BR if yes
          013B'CF    DF  0423  1128         PUSHAL  NOUNIT_SELECTED                 ; Else set up the error message...
                01   DD  0427  1129         PUSHL   #1                              ; ...argument count...
           00741132  8F  DD  0429  1130         PUSHL   #UETP$_TEXT!STS$K_ERROR         ; ...signal name...
                03   DD  042F  1131         PUSHL   #3                              ; ...and parameter count
          01CA'CF    14  D0  0431  1132         MOVL    #SS$_BADPARAM,STATUS            ; Set return status
                095C 31  0436  1133         BRW     ERROR_EXIT                      ; ...and give up, complaining
                     0439  1134 10$:
          0002'CF    04  A8  0439  1135         BISW2   #SAFE_TO_UPDM,FLAG              ; OK safe to update UETINIDEV.DAT now
                     043E  1136 20$:
                     043E  1137         $TRNLOG_S LOGNAM = MODE,-                   ; Get the run mode
                     043E  1138                   RSLLEN = BUFFER_PTR,-
                     043E  1139                   RSLBUF = FAO_BUF
          0014'CF    20  8A  0457  1140         BICB2   #LC_BITM,BUFFER             ; Convert to upper case
     0014'CF    4F  8F  91  045C  1141         CMPB    #^A/O/,BUFFER               ; Is this a one shot?
                08   12  0462  1142         BNEQ    TIME_IT                         ; BR if not
          0002'CF    02  A8  0464  1143         BISW2   #TEST_OVERM,FLAG            ; End after one iteration
                012D 31  0469  1144         BRW     ONE_SHOT                        ; Skip the SETIMR
                     046C  1145 TIME_IT:
                     046C  1146         $SETIMR_S DAYTIM = THREEMIN,-               ; Set timer AST to 3 minutes
                     046C  1147                   ASTADR = TIME_OUT,-
                     046C  1148                   EFN    = #EFN2
```

```
                        047F  1150          .SBTTL  Test the Terminals
                        047F  1151
 56  0230'CF  00000230'8F  C1  047F  1152          ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; Set the unit block address
                        0489  1153  RESTART:
                        0489  1154  ;*****************************************************************************
                        0489  1155  ;
                        0489  1156  ; Device test specific code goes here.
                        0489  1157  ;
                        0489  1158  ; At this point the device designation is in location DEV_NAME pointed to by
                        0489  1159  ; descriptor DEVDSC. The device is known to be supported and testable by this test.
                        0489  1160  ; To leave successfully BRW SUC_EXIT, to leave in error BRW ERROR_EXIT.
                        0489  1161  ;
                        0489  1162  ;*****************************************************************************
                        0489  1163
          01FA'CF   D5  0489  1164          TSTL    PASS                    ; Only create and connect on the first pass
               73   12  048D  1165          BNEQ    10$                     ; else branch
               07   E1  048F  1166          BBC     #SPL_UNITV,-            ; Is this unit spooled?
       03 0B A6         0491  1167                  UETUNT$B_FLAGS(R6),3$
             00AE   31  0494  1168          BRW     25$                     ; BR if so - try the next one
                        0497  1169  3$:
                        0497  1170          $CREATE  FAB = UETUNT$K_FAB(R6) ; Create the channel
            4E 50   E8  04A2  1171          BLBS    R0,5$                   ; Did we succeed? Br if yes...
          01C6'CF   D6  04A5  1172          INCL    ERROR_COUNT             ; Bump the error count
               50   DD  04A9  1173          PUSHL   R0                      ; Push the error code...
               50   DD  04AB  1174          PUSHL   R0                      ; ...and the error code...
            14 A6   9A  04AD  1175          MOVZBL  UETUNT$T_FILSPC(R6),-
          0098'CF         04B0  1176                  DEVDSC                  ; ...get the name size...
            15 A6   DE  04B3  1177          MOVAL   UETUNT$T_FILSPC+1(R6),-
          009C'CF         04B6  1178                  DEVDSC+4                ; ...get the name address...
          0098'CF   DF  04B9  1179          PUSHAL  DEVDSC                  ; ...and push the device designation...
          000F'CF   DF  04BD  1180          PUSHAL  TEST_NAME               ; ...and the test name...
      000F0003 8F   DD  04C1  1181          PUSHL   #^XF0003                ; ...and the arg count...
      00748194 8F   DD  04C7  1182          PUSHL   #UETP$_DEUNUS!STS$K_ERROR ; ...and the signal name...
          01C6'CF   DD  04CD  1183          PUSHL   ERROR_COUNT             ; ...and the error count...
          00A0'CF   DF  04D1  1184          PUSHAL  PROCESS_NAME            ; ...our own name...
      00010002 8F   DD  04D5  1185          PUSHL   #^X10002                ; ...and the argument count...
      00748022 8F   DD  04DB  1186          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...and the signal name...
00000000'GF    0A   FB  04E1  1187          CALLS   #10,G^LIB$SIGNAL        ; ...and print the error
            02   8A  04E8  1188          BICB2   #UETUNT$M_TESTABLE,-
         0B A6         04EA  1189                  UETUNT$B_FLAGS(R6)      ; Clear the testable bit
            01   88  04EC  1190          BISB2   #UETUNT$M_DONE,-
         0B A6         04EE  1191                  UETUNT$B_FLAGS(R6)      ; Set the done testing bit
          0052   31  04F0  1192          BRW     25$                     ; Skip testing
                        04F3  1193  5$:
                        04F3  1194          $CONNECT RAB = UETUNT$K_RAB(R6),-
                        04F3  1195                   ERR = RMS_ERROR        ; Connect the RAB
                        0502  1196  10$:
            02   88  0502  1197          BISB2   #UETUNT$M_TESTABLE,-
         0B A6         0504  1198                  UETUNT$B_FLAGS(R6)      ; Set the testable bit
       58  09EB'CF   DE  0506  1199          MOVAL   SERVICE_IO2,R8          ; Set the success service address
       57  0160 C6   DE  050B  1200          MOVAL   UETUNT$R_RAB(R6),R7     ; Get the RAB address
          028A C6   DE  0510  1201          MOVAL   UETUNT$K_HEADER+2(R6),-
            28 A7         0514  1202                  RAB$L_RBF(R7)           ; Set RAB write address
          02B8 C6   B0  0516  1203          MOVW    UETUNT$K_HEADER(R6),-
            22 A7         051A  1204                  RAB$W_RSZ(R7)           ; Set RAB write size
  0100 8F   22 A7   B1  051C  1205          CMPW    RAB$W_RSZ(R7),#HDR_OUT_SIZE ; Check the size
               0B   15  0522  1206          BLEQ    20$                     ; Br if small enough...
```

```
                22 A7    0100 8F    B0   0524  1207          MOVW    #HDR_OUT_SIZE,RAB$W_RSZ(R7) ; ...else make it small enough
                   58    09A1'CF    DE   052A  1208          MOVAL   SERVICE_IO1,R8              ; It will take more than one bite
                                           052F  1209                                              ; to output this header
                                           052F  1210 20$:
                         01A5 C6      90   052F  1211          MOVB    UETUNT$B_HD_LEN(R6),-
                         01A4 C6           0533  1212                  UETUNT$B_LINE(R6)           ; Save it's length in lines
                                           0536  1213          $WRITE  RAB = (R7),-
                                           0536  1214                  ERR = RMS_ERROR,-
                                           0536  1215                  SUC = (R8)                  ; Write the header async
                                           0545  1216 25$:
                            56   66      C0   0545  1217          ADDL2   (R6),R6                 ; Get the next unit block address
                00000230'8F     56      D1   0548  1218          CMPL    R6,#UNIT_LIST           ; Is this the end?
                                03      13   054F  1219          BEQL    30$                     ; Br if yes...
                              FF35      31   0551  1220          BRW     RESTART                 ; else do the next one...
                                           0554  1221 30$:
    5B  0230'CF   00000230'8F     C1   0554  1222          ADDL3   #UNIT_LIST,UNIT_LIST,R11 ; Set the unit block list header
                                           055E  1223 40$:
                                02      93   055E  1224          BITB    #UETUNT$M_TESTABLE,-
                             0B AB           0560  1225                  UETUNT$B_FLAGS(R11)     ; Is this unit testable?
                                22      12   0562  1226          BNEQ    50$                     ; BR if yes
                             5B   6B     C0   0564  1227          ADDL2   (R11),R11               ; Next unit block
                00000230'8F     5B      D1   0567  1228          CMPL    R11,#UNIT_LIST          ; Are we full circle in the list?
                                EE      12   056E  1229          BNEQ    40$                     ; BR if not
                         013B'CF        DF   0570  1230          PUSHAL  NOUNIT_SELECTED         ; Else set up the error message...
                                01      DD   0574  1231          PUSHL   #1                      ; ...argument count...
                      00741132 8F       DD   0576  1232          PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...signal name...
                                03      DD   057C  1233          PUSHL   #3                      ; ...and parameter count
                      01CA'CF   14      D0   057E  1234          MOVL    #SS$_BADPARAM,STATUS    ; Set return status
                              080F      31   0583  1235          BRW     ERROR_EXIT              ; ...and give up, complaining
                                           0586  1236 50$:
                                           0586  1237          $HIBER_S                        ; Service I/O AST's
                                           058D  1238          $CANTIM_S                       ; Forget the watchdog timer
                                0158       31   0596  1239          BRW     SUC_EXIT                ; Loop until the test is over
                                           0599  1240
                                           0599  1241 ONE_SHOT:
    56  0230'CF   00000230'8F     C1   0599  1242          ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; Get the list address
                                           05A3  1243 ONE_SHOT_LOOP:
                                07      E1   05A3  1244          BBC     #SPL_UNITV,-            ; Is this unit spooled?
                             03 0B A6        05A5  1245                  UETUNT$B_FLAGS(R6),5$
                                00BE       31   05A8  1246          BRW     NEXT_TTY                ; BR if it is - try the next
                                           05AB  1247 5$:
                      024D'CF   56      D0   05AB  1248          MOVL    R6,CUR_UNTBLK           ; Save a copy of the adr in perm mem
                                           05B0  1249          $SETIMR_S DAYTIM = FIVESEC,-
                                           05B0  1250                    ASTADR = TIMED_OUT,-
                                           05B0  1251                    EFN    = #EFN2          ; Set a 5 second watch dog timer
                                14 A6      9A   05C3  1252          MOVZBL  UETUNT$T_FILSPC(R6),-
                             0098'CF           05C6  1253                  DEVDSC                  ; Get the name size
                                15 A6      DE   05C9  1254          MOVAL   UETUNT$T_FILSPC+1(R6),-
                             009C'CF           05CC  1255                  DEVDSC+4                ; Get the name address
                                           05CF  1256          $SETSFM_S ENBFLG = #0           ; Don't bail out on this error
                                           05D8  1257          $ASSIGN_S DEVNAM = DEVDSC,-
                                           05D8  1258                    CHAN   = UETUNT$W_CHAN(R6) ; Assign the channel
                      01CA'CF   50      D0   05E8  1259          MOVL    R0,STATUS               ; Save the status
                        03 01CA'CF      E8   05ED  1260          BLBS    STATUS,10$              ; If OK then go on...
                                0083       31   05F2  1261          BRW     ONESHOT_ERROR           ; ...else take care of failure.
                                           05F5  1262 10$:
                      57   02B8 C6      3C   05F5  1263          MOVZWL  UETUNT$K_HEADER(R6),R7  ; Get the header size
```

```
02BA C6   57  0C04'CF   02   39  05FA  1264          MATCHC  #2,CRLF,R7,-
                                0603  1265                  UETUNT$K_HEADER+2(R6)   ; Find the CR/LF. There has to be one
                    02BA C6   DE  0603  1266          MOVAL   UETUNT$K_HEADER+2(R6),-
                          53        0607  1267                  R8                      ; Get the header start address
                    58    53  C3  0608  1268          SUBL3   R8,R3,R8                ; Get the byte count to the start of...
                  024D'CF   56  D0  060C  1269          MOVL    R6,CUR_UNTBLK           ; Save the unit block address
                                0611  1270          $QIOW_S FUNC = #IO$_WRITEVBLK,-
                                0611  1271                  CHAN = UETUNT$W_CHAN(R6),-
                                0611  1272                  IOSB = UETUNT$K_RAB+RAB$L_STS(R6),-
                                0611  1273                  P1   = UETUNT$K_HEADER+2(R6),-
                                0611  1274                  P2   = R8               ; Output the header
                       0B 50  E8  0633  1275          BLBS    R0,20$                  ; BR if no error...
                  01CA'CF   50  D0  0636  1276          MOVL    R0,STATUS               ; Save the status
                        59  50  D0  063B  1277          MOVL    R0,R9
                         0037 31  063E  1278          BRW     ONESHOT_ERROR           ; ...else take care of failure.
                                0641  1279 20$:
                                0641  1280          $CANTIM_S                       ; Forget the watch dog timer
                                064A  1281          $SETSFM_S ENBFLG = #1           ; Turn on SS failure mode
                  59  0168 C6  3C  0653  1282          MOVZWL  UETUNT$K_RAB+RAB$L_STS(R6),R9 ; Get the IOSB value
                       0A 59  E8  0658  1283          BLBS    R9,30$                  ; If the IOSB has no error we are OK
                  01CA'CF   59  D0  065B  1284          MOVL    R9,STATUS               ; Save the error code
                         0015 31  0660  1285          BRW     ONESHOT_ERROR           ; Report the failure
                           04 11  0663  1286          BRB     NEXT_TTY                ; ...and look at the rest of the lines
                                0665  1287 30$:
                           02 88  0665  1288          BISB2   #UETUNT$M_TESTABLE,-
                       0B A6      0667  1289                  UETUNT$B_FLAGS(R6)      ; We have a good line here. Say so.
                                0669  1290 NEXT_TTY:
                        56  66  C0  0669  1291          ADDL2   (R6),R6                 ; Get the next unit block
                56  00000230'8F  D1  066C  1292          CMPL    #UNIT_LIST,R6           ; Are we all done?
                           7C 13  0673  1293          BEQL    SUC_EXIT                ; Br if yes
                         FF2B 31  0675  1294          BRW     ONE_SHOT_LOOP           ; Else test the next line
                                0678  1295 ONESHOT_ERROR:
                                0678  1296          $CANTIM_S                       ; Forget the watchdog timer
                  01C6'CF   D6  0681  1297          INCL    ERROR_COUNT             ; Bump the error count
                  01CA'CF   DD  0685  1298          PUSHL   STATUS                  ; Push the error code...
                  01CA'CF   DD  0689  1299          PUSHL   STATUS                  ; ...and the error code...
                  0098'CF   DF  068D  1300          PUSHAL  DEVDSC                  ; ...and the device designation...
                  000F'CF   DF  0691  1301          PUSHAL  TEST_NAME               ; ...and the test name...
              000F0003 8F   DD  0695  1302          PUSHL   #^XF0003                ; ...and the arg count...
              0074819A 8F   DD  069B  1303          PUSHL   #UETP$_DEUNUS!STS$K_ERROR ; ...and the signal name...
                  01C6'CF   DD  06A1  1304          PUSHL   ERROR_COUNT             ; ...and the error count...
                  00A0'CF   DF  06A5  1305          PUSHAL  PROCESS_NAME            ; ...our own name...
              00010002 8F   DD  06A9  1306          PUSHL   #^X10002                ; ...and the argument count...
              00748022 8F   DD  06AF  1307          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...and the signal name...
00000000'GF     0A  FB  06B5  1308          CALLS   #10,G^LIB$SIGNAL        ; ...and print the error
          0830 8F   59  B1  06BC  1309          CMPW    R9,#SS$_CANCEL          ; Did a timeout occur?
                    05  13  06C1  1310          BEQL    10$                     ; BR if it was
                  2C    59  B1  06C3  1311          CMPW    R9,#SS$_ABORT           ; Did a timeout occur?
                        13  12  06C6  1312          BNEQ    20$                     ; BR if not
                                06C8  1313 10$:
                  020C'CF   DF  06C8  1314          PUSHAL  TIME_OUT_MSG            ; Print the timeout error message
                        01  DD  06CC  1315          PUSHL   #1
              00741132 8F   DD  06CE  1316          PUSHL   #UETP$_TEXT!STS$K_ERROR
00000000'GF     03  FB  06D4  1317          CALLS   #3,G^LIB$SIGNAL
                                06DB  1318 20$:
                     FF8B 31  06DB  1319          BRW     NEXT_TTY                ; Go do the next unit
                                06DE  1320 TIMED_OUT:
```

```
                     0000  06DE  1321          .WORD   0                               ; Go here with watchdog timer timeout
          56  024D'CF    D0  06E0  1322          MOVL    CUR_UNTBLK,R6                   ; Get the unit block address
                         06E5  1323          $CANCEL_S CHAN = UETUNT$W_CHAN(R6) ; This is one IO that will never complete
                         04  06F0  1324          RET
                         06F1  1325
                         06F1  1326 SUC_EXIT:
          66 0002'CF     07  E0  06F1  1327          BBS     #CTRLC_SEENV,FLAG,10$   ; Exit now if user typed CTRL/C
                         06F7  1328          $TRNLOG_S LOGNAM = MODE,-
                         06F7  1329                    RSLLEN = BUFFER_PTR,-
                         06F7  1330                    RSLBUF = FAO_BUF        ; Get the run mode
             0014'CF     20  8A  0710  1331          BICB2   #LC_BITM,BUFFER         ; Convert to upper case
        0014'CF   4C 8F     91  0715  1332          CMPB    #^A7L/,BUFFER           ; Is this a loop for ever?
                    40      12  071B  1333          BNEQ    10$                     ; BR if not
             0002'CF     02  AA  071D  1334          BICW2   #TEST_OVERM,FLAG        ; Reset the termination flag
                  01FA'CF    D6  0722  1335          INCL    PASS                    ; Bump the pass count
                         0726  1336          $FAO_S  CTRSTR = PASS_MSG,-
                         0726  1337                  OUTLEN = BUFFER_PTR,-
                         0726  1338                  OUTBUF = FAO_BUF,-
                         0726  1339                  P1     = PASS,-
                         0726  1340                  P2     = ITERATION,-
                         0726  1341                  P3     = #0             ; Make the end of pass message
                    000C'CF    DF  0743  1342          PUSHAL  BUFFER_PTR              ; Push the string desc.
                         01    DD  0747  1343          PUSHL   #1                      ; Push arg count
                00741133 8F    DD  0749  1344          PUSHL   #UETP$_TEXT!STS$K_INFO  ; Push the signal name
           00000000'GF   03    FB  074F  1345          CALLS   #3,G^LIB$SIGNAL         ; Print the end of pass message
                  01F2'CF    D4  0756  1346          CLRL    ITERATION               ; Reset the iteration count
                     FD0F    31  075A  1347          BRW     TIME_IT                 ; Do the next pass
                         075D  1348 10$:
     01CA'CF    10000001 8F    D0  075D  1349          MOVL    #SS$_NORMAL!STS$M_INHIB_MSG,STATUS ; Set successful exit status
                         0766  1350          $EXIT_S STATUS                          ; Exit with the status
```

```
0771  1352          .SBTTL  GET_NODE Routine
0771  1353  ;++
0771  1354  ; FUNCTIONAL DESCRIPTION:
0771  1355  ; This routine dynamically allocates enough memory for a unit block,
0771  1356  ; a device dependent parameter area and I/O buffers. The unit block is inserted
0771  1357  ; into the queue header UNIT_LIST.  It then initializes the unit block.
0771  1358  ; The unit block format is as follows:
0771  1359  ;
0771  1360  ;                              +----------------+ -------
0771  1361  ;       UETUNT$L_FLINK         !                !       ^
0771  1362  ;                              +----------------+       !
0771  1363  ;       UETUNT$L_BLINK         !                !       !
0771  1364  ;                              +------------+---+       !
0771  1365  ;       UETUNT$B_TYPE                       !   !       !
0771  1366  ;                                  +-------+---+       !
0771  1367  ;       UETUNT$W_SIZE              !   !   !    contains DEVDEP_SIZE + UETUNT$C_INDSIZ
0771  1368  ;                              +---+-------+           !
0771  1369  ;       UETUNT$B_FLAGS         !   !                   !
0771  1370  ;                              +---+   +-------+       !
0771  1371  ;       UETUNT$W_CHAN                  !   !   !       !
0771  1372  ;                              +-------+-------+       !
0771  1373  ;       UETUNT$W_FUNC          !   !   !               !
0771  1374  ;                              +-------+-------+       +----- UETUNT$C_SIZE
0771  1375  ;       UETUNT$L_ITER          !                       !
0771  1376  ;                              +----------------+      !
0771  1377  ;       UETUNT$T_FILSPC        !                !      !
0771  1378  ;                              !/\/\/\/\/\/\/\/!      !
0771  1379  ;                              NAM$C_MAXRSS bytes     !
0771  1380  ;                              !/\/\/\/\/\/\/\/!      !
0771  1381  ;                              !                !      !
0771  1382  ;       UETUNT$K_FAB           !----------------!      !
0771  1383  ;                              !                !      !
0771  1384  ;                              !/\/\/\/\/\/\/\/!      !
0771  1385  ;                               FAB$C_BLN bytes       !
0771  1386  ;                              !/\/\/\/\/\/\/\/!      !
0771  1387  ;                              !                !      !
0771  1388  ;       UETUNT$K_RAB           +----------------+      !
0771  1389  ;                              !                !      !
0771  1390  ;                              !\/\/\/\/\/\/\/\!      !
0771  1391  ;                               RAB$C_BLN bytes       !
0771  1392  ;                              !\/\/\/\/\/\/\/\!      !
0771  1393  ;                              !                !      v
0771  1394  ;       UETUNT$K_DEVDEP        +------------+---+ -------
0771  1395  ;       UETUNT$B_LINE                       !   !       ^
0771  1396  ;                                      +---+---+       !
0771  1397  ;       UETUNT$B_HD_LEN                !   !           !
0771  1398  ;                                  +---+---+           !
0771  1399  ;       UETUNT$B_LENGTH            !   !               !
0771  1400  ;                              +---+---+               +----- DEVDEP_SIZE
0771  1401  ;       UETUNT$W_WIDTH         !   !                   !
0771  1402  ;                              +-------+-------+       !
0771  1403  ;       UETUNT$L_CURHDR        !               !       !
0771  1404  ;                              +----------------+      !
0771  1405  ;       UETUNT$K_CCTBL         !                !      v
0771  1406  ;       UETUNT$K_WRITE         +------------+---+ -------
```

```
                        ;       UETUNT$K_WRITE  +----------------+ -------
                        ;       WRITE buffer    !                !        ^
                        ;                       .\/\/\/\/\/\/\/\/!        !
                        ;                          257 bytes              +----- WRITE_SIZE
                        ;                       '/\/\/\/\/\/\/\/!        !
                        ;                       .                !        v
                        ;       UETUNT$K_HEADER +----------------+ -------
                        ;       HEADER buffer                    !        ^
                        ;                       !\/\/\/\/\/\/\/\/!        !
                        ;                          132 bytes              +----- HEADER_SIZE
                        ;                       !/\/\/\/\/\/\/\/!        !
                        ;                       !                !        v
                        ;                       +----------------+ -------
                        ;
                        ; CALLING SEQUENCE:
                        ;       BSBW    GET_NODE
                        ;
                        ; INPUT PARAMETERS:
                        ;       NONE
                        ;
                        ; IMPLICIT INPUTS:
                        ;       UNIT_LIST contains head of doubley linked circular list of unit blocks
                        ;
                        ; OUTPUT PARAMETERS:
                        ;       R6 = address of  new node
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;       New inited unit block is inserted into UNIT_LIST
                        ;
                        ; COMPLETION CODES:
                        ;       NONE
                        ;
                        ; SIDE EFFECTS:
                        ;       NONE
                        ;
                        ;--
                        ;
                        GET_NODE:
                        ;
                                $EXPREG_S PAGCNT = #PAGES,-
                                          RETADR = NEW_NODE     ; Get a new node of demand zero memory
0230'CF   0238'DF   5D  0782            INSQTI  @NEW_NODE,UNIT_LIST     ; Put the new node in the unit list
      56  0238'CF   D0  0789            MOVL    NEW_NODE,R6             ; Save a copy of its address
        08 A6   01  90  078E            MOVB    #1,UETUNT$B_TYPE(R6)    ; Set the structure type
           01B7 8F  B0  0792            MOVW    #UETUNT$C_IRDSIZ+DEVDEP_SIZE,-
             09 A6      0796                    UETUNT$W_SIZE(R6)       ; Set the structure size
  14 A6    0098'CF  90  0798            MOVB    DEVDSC,UETUNT$T_FILSPC(R6) ; Set the device name size
009C'DF    0098'CF  28  079E            MOVC3   DEVDSC,@DEVDSC+4,-
             15 A6      07A5                    UETUNT$T_FILSPC+1(R6)   ; Save the device name
           0094 8F  28  07A7            MOVC3   #FAB$C_BLN+RAB$C_BLN,-
0110 C6    03D8'CF      07AB                    DUMMY_FAB,UETUNT$C_FAB(R6) ; Save a FAB and a RAB away
      57   0110 C6  DE  07B1            MOVAL   UETUNT$K_FAB(R6),R7     ; Save the FAB address
      58   0160 C6  DE  07B6            MOVAL   UETUNT$K_RAB(R6),R8     ; Save the RAB address
        3C A8   57  D0  07BB            MOVL    R7,RAB$L_FAB(R8)        ; Set the FAB address in the RAB
             14 A6  90  07BF            MOVB    UETUNT$T_FILSPC(R6),-
             34 A7      07C2                    FAB$B_FNS(R7)           ; Set the FNS field in the FAB
             15 A6  DE  07C4            MOVAL   UETUNT$T_FILSPC+1(R6),-
```

```
                2C A7        07C7  1465                    FAB$L_FNA(R7)           ; Set the FNA field in the FAB
          18 A8   66   DE    07C9  1466          MOVAL     (R6),RAB$L_CTX(R8)      ; Set the UETUNT address in the RAB
             02BA C6   DE    07CD  1467          MOVAL     UETUNT$K_HEADER+2(R6),-
                28 A8        07D1  1468                    RAB$L_RBF(R8)           ; Set the buffer address
                             07D3  1469          RSB
```

```
                                07D4  1471          .SBTTL  SET_DEVDEP Routine
                                07D4  1472 ;++
                                07D4  1473 ; FUNCTIONAL DESCRIPTION:
                                07D4  1474 ;       This routine initializes the device dependent parameters in the unit
                                07D4  1475 ;       block specified.
                                07D4  1476 ;
                                07D4  1477 ; CALLING SEQUENCE:
                                07D4  1478 ;       PUSHAL  DIB                     ; Push the device information block adr
                                07D4  1479 ;       PUSHAL  UNIT_BLOCK              ; Push the unit block address
                                07D4  1480 ;       CALLS   #2,SET_DEVDEP           ; Set the device characteristics
                                07D4  1481 ;
                                07D4  1482 ; INPUT PARAMETERS:
                                07D4  1483 ;       4(AP) address of the unit block
                                07D4  1484 ;       8(AP) address of the device information block
                                07D4  1485 ;
                                07D4  1486 ; IMPLICIT INPUTS:
                                07D4  1487 ;       NONE
                                07D4  1488 ;
                                07D4  1489 ; OUTPUT PARAMETERS:
                                07D4  1490 ;       NONE
                                07D4  1491 ;
                                07D4  1492 ; IMPLICIT OUTPUTS:
                                07D4  1493 ;       Device characteristics are inited in the unit block
                                07D4  1494 ;
                                07D4  1495 ; COMPLETION CODES:
                                07D4  1496 ;       NONE
                                07D4  1497 ;
                                07D4  1498 ; SIDE EFFECTS:
                                07D4  1499 ;       NONE
                                07D4  1500 ;
                                07D4  1501 ;--
                                07D4  1502
                                07D4  1503 SET_DEVDEP:
                          0FFC  07D4  1504          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                07D6  1505
          56    04 AC     D0  07D6  1506          MOVL    4(AP),R6                ; Get the unit block address
          59    08 AC     D0  07DA  1507          MOVL    8(AP),R9                ; Get the DIB address
       5A    0523'CF     DE  07DE  1508          MOVAL   TESDEV_TBL,R10          ; Set the table address
          05 A9    21     91  07E3  1509          CMPB    #DT$_LA120,DIB$B_DEVTYPE(R9) ; Should only 2 pages be printed?
                   04     12  07E7  1510          BNEQ    10$                     ; Br if not...
          0B A6    04     88  07E9  1511          BISB2   #UETUNT$M_2PL,UETUNT$B_FLAGS(R6) ; ...else set the flag
                                07ED  1512 10$:
       04 A9    43 8F     91  07ED  1513          CMPB    #DC$_LP,DIB$B_DEVCLASS(R9) ; Should only 2 pages be printed?
                   04     12  07F2  1514          BNEQ    20$                     ; Br if not ...
          0B A6    04     88  07F4  1515          BISB2   #UETUNT$M_2PL,UETUNT$B_FLAGS(R6) ; ...else set the flag
                                07F8  1516 20$:
          04 A9    6A     B1  07F8  1517          CMPW    (R10),DIB$B_DEVCLASS(R9) ; Is this the right table entry?
                   58     13  07FC  1518          BEQL    30$                     ; Br if yes
FFF2 5A   0F    0640'8F     3D  07FE  1519          ACBW    #TESDEV_TBL_END,#15,R10,20$ ; Do it for the whole table
                                0806  1520          $GETDVI_S DEVNAM = DEVDSC,-     ; Get the name of the device...
                                0806  1521                    EFN    = #SS_SYNCH_EFN,- ; ...from which the test is run.
                                0806  1522                    ITMLST = TERM_ITMLST,- ; We have a special need here,...
                                0806  1523                    IOSB   = QUAD_STATUS   ; ...get it now & save it in BUFFER
          12 01CE'CF     E9  0822  1524          BLBC    QUAD_STATUS,25$         ; Abort if we can't get it
 0251'CF   01000000 8F     D3  0827  1525          BITL    #TT2$M_ANSICRT,XTERM_CHAR ; Is this an ANSI terminal?
                   1F     13  0830  1526          BEQL    27$                     ; BR if not...
       5A    0523'CF     DE  0832  1527          MOVAL   TESDEV_TBL,R10          ; ...otherwise set up as a VT100...
```

```
                     1D   11  0837  1528          BRB     30$                     ; ...and carry on
                                0839  1529 25$:
          01CA'CF    01CE'CF  B0  0839  1530          MOVW    QUAD_STATUS,STATUS      ; Save the exit status
                     00D4'CF  DF  0840  1531          PUSHAL  NO_CTRLNAME             ; Prepare for message...
                          01  DD  0844  1532          PUSHL   #1                      ; ...arg count
                   00741132  8F  DD  0846  1533          PUSHL   #UETP$_TEXT!STS$K_ERROR ; ...signal name
                          03  DD  084C  1534          PUSHL   #3                      ; ...arg count
                        0544  31  084E  1535          BRW     ERROR_EXIT              ; ...go tell of bad setup
                                0851  1536 27$:
                5A      05F5'CF  DE  0851  1537          MOVAL   UNKNOWN1,R10            ; Assume the lowest level of test
                                0856  1538 30$:
                     02BA C6     DE  0856  1539          MOVAL   UETUNT$K_HEADER+2(R6),-
                     0244'CF         085A  1540                  HEAD_BUF+4              ; Set the output address
                01A5 C6    02 AA  90  085D  1541          MOVB    2(R10),UETUNT$B_HD_LEN(R6) ; Save the header length in lines
          00000648'8F      07 AA  D1  0863  1542          CMPL    7(R10),#VT100_PREAMBLE  ; Is this a scroll region device?
                          04  12  086B  1543          BNEQ    35$                     ; BR if not
                   0B A6      20  88  086D  1544          BISB2   #SCROL_CLRM,UETUNT$B_FLAGS(R6) ; else set the scrol clear flag
                                0871  1545 35$:
                                0871  1546          $FAO_S  CTRSTR = HEAD_CTRSTR,-
                                0871  1547                  OUTLEN = HEAD_LENGTH,-
                                0871  1548                  OUTBUF = HEAD_BUF,-
                                0871  1549                  P1     = 7(R10),-
                                0871  1550                  P2     = 3(R10),-
                                0871  1551                  P3     = #DEVDSC        ; Form the header + preamble
                     0248'CF    B0  0890  1552          MOVW    HEAD_LENGTH,-
                     02B8 C6        0894  1553                  UETUNT$K_HEADER(R6)      ; Save the header/preamble length
                58   02BA C6    DE  0897  1554          MOVAL   UETUNT$K_HEADER+2(R6),R8 ; Get the header start address
                     0100 C8    DE  089C  1555          MOVAL   HDR_OUT_SIZE(R8),-
                     01AB C6        08A0  1556                  UETUNT$L_CURHDR(R6)      ; Set the current header address
                57   0248'CF    3C  08A3  1557          MOVZWL  HEAD_LENGTH,R7          ; Get the header length
                     58    57   C0  08A8  1558          ADDL2   R7,R8                   ; Make the start address for the TSD
                  57    0B AA    D0  08AB  1559          MOVL    11(R10),R7              ; Get the TSD desc address
               68   04 B7    67  28  08AF  1560          MOVC3   (R7),@4(R7),(R8)        ; Copy the TSD into the unit block
                02B8 C6     67  A0  08B4  1561          ADDW2   (R7),UETUNT$K_HEADER(R6) ; update the total header length
                                08B9  1562 40$:
                      0B A9     90  08B9  1563          MOVB    DIB$L_DEVDEPEND+3(R9),-
                     01A6 C6        08BC  1564                  UETUNT$B_LENGTH(R6)      ; Set the page length
                        06 A9    3C  08BF  1565          MOVZWL  DIB$W_DEVBUFSIZ(R9),-
                           57        08C2  1566                  R7                       ; Get the line size
                     01A7 C6    57  B0  08C3  1567          MOVW    R7,UETUNT$W_WIDTH(R6)    ; Save the line size
                           57    08  A0  08C8  1568          ADDW2   #SIZE_TBL_LEN,R7         ; Add in the CCTBL table length
          01AF C6    0352'CF    57  28  08CB  1569          MOVC3   R7,SIZE_TBL,UETUNT$K_CCTBL(R6) ; Init the CC tbl and write buffer
                           57    08  C2  08D3  1570          SUBL2   #SIZE_TBL_LEN,R7         ; Take size table back out
                     01AF C6    57  A0  08D6  1571          ADDW2   R7,UETUNT$K_CCTBL(R6)    ; Set the record lengths
                     01B1 C6    57  A0  08DB  1572          ADDW2   R7,UETUNT$K_CCTBL+2(R6)
                     01B3 C6    57  A0  08E0  1573          ADDW2   R7,UETUNT$K_CCTBL+4(R6)
                     01B5 C6    57  A0  08E5  1574          ADDW2   R7,UETUNT$K_CCTBL+6(R6)
                57    000001B7  8F  C0  08EA  1575          ADDL2   #UETUNT$K_WRITE,R7       ; Calculate the byte offset into the block
                           57    56  C0  08F1  1576          ADDL2   R6,R7                    ; Calculate the address
                67   0C06'CF    B0  08F4  1577          MOVW    LFCR,(R7)                ; Set up the termination characters
             08 A9    00000080  8F  D3  08F9  1578          BITL    #TT$M_LOWER,DIB$L_DEVDEPEND(R9) ; Is this a lower case terminal?
                              1F  12  0901  1579          BNEQ    50$                      ; Br if yes
          00C6'CF     00000101  8F  D0  0903  1580          MOVL    #WRITE_SIZE,DATA_DSC     ; Init the string descriptor
                     01B7 C6    DE  090C  1581          MOVAL   UETUNT$K_WRITE(R6),-
                     00CA'CF        0910  1582                  DATA_DSC+4
                     00C6'CF    DF  0913  1583          PUSHAL  DATA_DSC                 ; Set up STR$UPCASE destination
                     00C6'CF    DF  0917  1584          PUSHAL  DATA_DSC                 ; Set up source
```

```
00000000'GF   02   FB  091B  1585          CALLS   #2,G^STR$UPCASE          ; Convert data buffer to upper case
                       0922  1586 50$:
          10 A6    D6  0922  1587          INCL    UETUNT$L_ITER(R6)        ; Set to page 1
             06    E1  0925  1588          BBC     #DEV$V_SPL,-
          4D 69        0927  1589                  DIB$L_DEVCHAR(R9),60$    ; Check for the spool bit
          81 8F    88  0929  1590          BISB2   #SPL_UNITM!UETUNT$M_DONE,- ; We can't test spooled devices
          0B A6        092C  1591                  UETUNT$B_FLAGS(R6)
                       092E  1592          $FAO_S  CTRSTR = NOSPOOLED,-     ; Let the user know about it
                       092E  1593                  OUTLEN = BUFFER_PTR,-
                       092E  1594                  OUTBUF = FAO_BUF,-
                       092E  1595                  P1     = #DEVDSC
      01C6'CF    D6  0947  1596          INCL    ERROR_COUNT
      000C'CF    DF  094B  1597          PUSHAL  BUFFER_PTR
  000F0001 8F    DD  094F  1598          PUSHL   #^XF0001
  00741130 8F    DD  0955  1599          PUSHL   #UETP$_TEXT!STS$K_WARNING
      01C6'CF    DD  095B  1600          PUSHL   ERROR_COUNT
      00A0'CF    DF  095F  1601          PUSHAL  PROCESS_NAME
  00010002 8F    DD  0963  1602          PUSHL   #^X10002
  00748020 8F    DD  0969  1603          PUSHL   #UETP$_ERBOXPROC!STS$K_WARNING
00000000'GF   07   FB  096F  1604          CALLS   #7,G^LIB$SIGNAL
                       0976  1605 60$:
                   04  0976  1606          RET
```

```
                        0977  1608          .SBTTL SERVICE_IO
                        0977  1609 ;++
                        0977  1610 ; FUNCTIONAL DESCRIPTION:
                        0977  1611 ;       This routine services all asynchronous I/O completions for all unit
                        0977  1612 ;       numbers under test.  After the proper number of iterations or elapsed
                        0977  1613 ;       time have been completed the unit number is disabled for testing.
                        0977  1614 ;       This routine does the data checking as well as the next I/O initiation.
                        0977  1615 ;
                        0977  1616 ; CALLING SEQUENCE:
                        0977  1617 ;       Entered via an AST from the RMS
                        0977  1618 ;
                        0977  1619 ; INPUT PARAMETERS:
                        0977  1620 ;       4(AP) = address of RAB of unit completing I/O
                        0977  1621 ;
                        0977  1622 ; OUTPUT PARAMETERS:
                        0977  1623 ;       None
                        0977  1624 ;--
                        0977  1625
                        0977  1626 SERVICE_IO:
                   0FFC 0977  1627          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                        0979  1628
          5B    04 AC   D0   0979  1629          MOVL    4(AP),R11                       ; Get the RAB address
                        097D  1630 NEW_PAGE:
                        097D  1631          $DISCONNECT RAB = (R11),-
                        097D  1632                      ERR = RMS_ERROR                 ; Disconnect to set IO mode
04 AB   00000800 8F   C8   098A  1633          BISL2   #RAB$M_BIO,RAB$L_ROP(R11)       ; Set to block IO for the header
                        0992  1634          $CONNECT    RAB = (R11),-
                        0992  1635                      ERR = RMS_ERROR                 ; OK now hook it back up
                02'   11   099F  1636          BRB     SERVICE_IO1+2                   ; This is first time only for the
                        09A1  1637 ;
                        09A1  1638 ; This entry point is to service header AST's.
                        09A1  1639 ;
                        09A1  1640
                        09A1  1641 SERVICE_IO1:
                   0FFC 09A1  1642          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
      5B    04 AC   D0   09A3  1643          MOVL    4(AP),R11                       ; Get the RAB address
      5A    18 AB   D0   09A7  1644          MOVL    RAB$L_CTX(R11),R10              ; Save the unit block address
      58    F3 AF   DE   09AB  1645          MOVAL   SERVICE_IO1,R8                  ; Set the AST service routine addres
28 AB    01AB CA   D0   09AF  1646          MOVL    UETUNT$L_CURHDR(R10),RAB$L_RBF(R11) ; Set the transfer address
   57    02B8 CA   3C   09B5  1647          MOVZWL  UETUNT$K_HEADER(R10),R7         ; Get the header size
   59    02BA CA   DE   09BA  1648          MOVAL   UETUNT$K_HEADER+2(R10),R9       ; Get the header start address
         59    57   C0   09BF  1649          ADDL2   R7,R9                           ; Calculate the end address
00000100 8F    59   F1   09C2  1650          ACBL    R9,#HDR_OUT_SIZE,-
   000D 01AB CA         09C9  1651                  UETUNT$L_CURHDR(R10),10$        ; Output header in chunks
      59    28 AB   C2   09CE  1652          SUBL2   RAB$L_RBF(R11),R9               ; If not enough header left for 256.
      22 AB    59   B0   09D2  1653          MOVW    R9,RAB$W_RSZ(R11)               ; ...then set size to whats left...
   58    09EB'CF   DE   09D6  1654          MOVAL   SERVICE_IO2,R8                  ; ...and setup AST adr for data
                        09DB  1655 10$:
                        09DB  1656          $WRITE  RAB = (R11),-
                        09DB  1657                  SUC = (R8),-
                        09DB  1658                  ERR = RMS_ERROR                 ; Start the next header write
                   04   09EA  1659          RET
                        09EB  1660 ;
                        09EB  1661 ; This entry point is to set up for non-header data transfers.
                        09EB  1662 ;
                        09EB  1663
                        09EB  1664 SERVICE_IO2:
```

```
                         0FFC  09EB              .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
          5B    04 AC     D0   09ED              MOVL    4(AP),R11                      ; Get the RAB address
                               09F1              $DISCONNECT RAB = (R11),-
                               09F1                      ERR = RMS_ERROR                ; Disconnect to set IO mode
   04 AB    00000800 8F   CA   09FE              BICL2   #RAB$M_BIO,RAB$L_ROP(R11)      ; Set to record IO for the data
                               0A06              $CONNECT    RAB = (R11),-
                               0A06                      ERR = RMS_ERROR                ; OK now hook it back up
                   02'    11   0A13              BRB     SERVICE_IO3+2                  ; This is first time only for the
                               0A15
                               0A15       ;
                               0A15       ; This entry point is to service normal data transfers.
                               0A15       ;
                               0A15
                               0A15       SERVICE_IO3.
                         0FFC  0A15              .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
          5B    04 AC     D0   0A17              MOVL    4(AP),R11                      ; Get the RAB address
          5A    18 AB     D0   0A1B              MOVL    RAB$L_CTX(R11),R10             ; Save the unit block address
                   58     7C   0A1F              CLRQ    R8                             ; Clear the area
             01A5 CA      83   0A21              SUBB3   UETUNT$B_HD_LEN(R10),-
        58   01A4 CA           0A25                      UETUNT$B_LINE(R10),R8          ; Leave the non-header line cnt
                               0A29                                                     ; in R8
   58   58   58   11      7B   0A29              EDIV    #CC_TBL_SIZE,R8,R8,R8          ; Leave the CC_TBL index in R8
     57   045B'CF48       D0   0A2E              MOVL    CC_TBL+T[R8],R7                ; Get the size table index
             57   57      9A   0A34              MOVZBL  R7,R7                          ; Clean off the garbage
     28 AB   01B7 CA      DE   0A37              MOVAL   UETUNT$K_WRITE(R10),RAB$L_RBF(R11) ; Set the new transfer adr
   22 AB   01AF CA47      B0   0A3D              MOVW    UETUNT$K_CCTBL(R10)[R7],-
                               0A44                      RAB$W_RSZ(R11)                 ; Set the transfer size
   2C AB    045C'CF48     DE   0A44              MOVAL   CC_TB[+2[R8],RAB$L_RHB(R11)    ; Set the carriage control char's
        59   01A7 CA      D0   0A4B              MOVL    UETUNT$W_WIDTH(R10),R9         ; Get the line length
                   59     D7   0A50              DECL    R9                             ; Move it over one byte
  024C'CF    01B7 CA49    90   0A52              MOVB    UETUNT$K_WRITE(R10)[R9],SPARE  ; Rotate the data pattern...
      01B7 CA    59       28   0A5A              MOVC3   R9,UETUNT$K_WRITE(R10),-
             01B8 CA           0A5F                      UETUNT$K_WRITE+1(R10)          ; ...
   01B7 CA   024C'CF      90   0A62              MOVB    SPARE,UETUNT$K_WRITE(R10)      ;
             01A4 CA      96   0A69              INCB    UETUNT$B_LINE(R10)             ; One more line done
             01A4 CA      91   0A6D              CMPB    UETUNT$B_LINE(R10),-
             01A6 CA           0A71                      UETUNT$B_LENGTH(R10)           ; Is the page full?
                   03     13   0A74              BEQL    10$                            ; BR if yes
                 00B2     31   0A76              BRW     80$                            ; Br if not
                               0A79       10$:
             02BA CA      DE   0A79              MOVAL   UETUNT$K_HEADER+2(R10),-
             01AB CA           0A7D                      UETUNT$L_CURHDR(R10)           ; Reset header address
     26 0B AA    05       E1   0A80              BBC     #SCROL_CLRV,UETUNT$B_FLAGS(R10),20$ ; BR if no scroll to clear
     28 AB   0A38'CF      DE   0A85              MOVAL   VT100_ORIG_SCROL,RAB$L_RBF(R11) ; Set the reset scroll address
          22 AB   0E      B0   0A8B              MOVW    #VOSL,RAB$W_RSZ(R11)           ; Set the reset scroll length
                               0A8F              $PUT    RAB = (R11),-
                               0A8F                      ERR = RMS_ERROR                ; Reset the scroll region
                               0A9C              $WAIT   RAB = (R11)                    ; Wait for the I/O to finish
     28 AB   01B7 CA      DE   0AA5              MOVAL   UETUNT$K_WRITE(R10),RAB$L_RBF(R11) ;  Reset the write address
                               0AAB       20$:
                               0AAB              $SETIMR_S DAYTIM = TWOSEC,-            ; Pause in testing to give other
                               0AAB                      EFN     = #12                  ;  tests a chance to run
                               0ABA              $WAITFR_S EFN = #12
     2C 0002'CF    01     E1   0AC3              BBC     #TEST_OVERV,FLAG,50$           ; BR if we are done
          0B AA    01     88   0AC9              BISB2   #UETUNT$M_DONE,UETUNT$B_FLAGS(R10) ; ...else set our done flag
                               0ACD       30$:
 56   0230'CF   00000230'8F   C1   0ACD          ADDL3   #UNIT_LIST,UNIT_LIST,R6        ; Get the list header
```

```
                                0AD7  1722 40$:
62 0B A6    00      E1  0AD7  1723          BBC     #UETUNT$V_DONE,UETUNT$B_FLAGS(R6),90$ ; BR if any unit is not done
  56    66  56      C1  0ADC  1724          ADDL3   R6,(R6),R6                      ; Get the next unit block adr
00000230'8F 56      D1  0AE0  1725          CMPL    R6,#UNIT_LIST                   ; All done checking?
            EE      12  0AE7  1726          BNEQ    40$                             ; BR if yes
                        0AE9  1727          $WAKE_S                                 ; Else its all over with
                    04  0AF4  1728          RET                                     ; Tha-a-Tha-a-Thats all folks
                        0AF5  1729 50$:
        10 AA       D6  0AF5  1730          INCL    UETUNT$L_ITER(R10)              ; Bump iteration counter
    01F2'CF         D6  0AF8  1731          INCL    ITERATION                       ; and test wide iteration count
0C 0B AA    02      E1  0AFC  1732          BBC     #UETUNT$V_2PL,UETUNT$B_FLAGS(R10),60$ ; Br if no 2 page limit
   10 AA    03      D1  0B01  1733          CMPL    #3,UETUNT$L_ITER(R10)           ; See if 2 pages are out
            06      12  0B05  1734          BNEQ    60$                             ; Br if not yet
   0B AA    01      88  0B07  1735          BISB2   #UETUNT$M_DONE,UETUNT$B_FLAGS(R10) ; Set the done flag
            C0      11  0B0B  1736          BRB     30$                             ; Stop testing
                        0B0D  1737 60$:
    01A5 CA     90  0B0D  1738          MOVB    UETUNT$B_HD_LEN(R10),-
    01A4 CA         0B11  1739                  UETUNT$B_LINE(R10)              ; Init line count
    02B8 CA     B0  0B14  1740          MOVW    UETUNT$K_HEADER(R10),-
      22 AB         0B18  1741                  RAB$W_RSZ(R11)                  ; Set RAB write size
0100 8F   22 AB     B1  0B1A  1742          CMPW    RAB$W_RSZ(R11),#HDR_OUT_SIZE    ; Check the size
            06      15  0B20  1743          BLEQ    70$                             ; Br if small enough...
22 AB   0100 8F     B0  0B22  1744          MOVW    #HDR_OUT_SIZE,RAB$W_RSZ(R11)    ; ...else make it small enough
                        0B28  1745 70$:
          FE52      31  0B28  1746          BRW     NEW_PAGE                        ; Start a fresh page
                        0B2B  1747 80$:
                        0B2B  1748          $PUT    RAB = UETUNT$K_RAB(R10),-
                        0B2B  1749                  ERR = RMS_ERROR,-
                        0B2B  1750                  SUC = SERVICE_IO3               ; Kick off the next I/O operation
                        0B3E  1751 90$:
                    04  0B3E  1752          RET
```

```
                        0B3F  1754          .SBTTL  Timer Expiration Routine
                        0B3F  1755 ;++
                        0B3F  1756 ; FUNCTIONAL DESCRIPTION:
                        0B3F  1757 ;        This routine will be called when the three minute pass timer expires.
                        0B3F  1758 ;        The number of pages output by each unit is checked. If less then two
                        0B3F  1759 ;        pages were completed the device must be very slow or hung. To force
                        0B3F  1760 ;        completion of the slow units the line count is set to one less than
                        0B3F  1761 ;        the total page size. A two minute watchdog timer is then set - all
                        0B3F  1762 ;        units should finish unless they are hung. If the watchdog timer goes
                        0B3F  1763 ;        off an error message is output for all the units that did not finish.
                        0B3F  1764 ;
                        0B3F  1765 ; CALLING SEQUENCE:
                        0B3F  1766 ;        Called via AST at $SETIMR expiration.
                        0B3F  1767 ;
                        0B3F  1768 ; INPUT PARAMETERS:
                        0B3F  1769 ;        NONE
                        0B3F  1770 ;
                        0B3F  1771 ; IMPLICIT INPUTS:
                        0B3F  1772 ;        NONE
                        0B3F  1773 ;
                        0B3F  1774 ; OUTPUT PARAMETERS:
                        0B3F  1775 ;        NONE
                        0B3F  1776 ;
                        0B3F  1777 ; IMPLICIT OUTPUTS:
                        0B3F  1778 ;        NONE
                        0B3F  1779 ;
                        0B3F  1780 ; COMPLETION CODES:
                        0B3F  1781 ;        NONE
                        0B3F  1782 ;
                        0B3F  1783 ; SIDE EFFECTS:
                        0B3F  1784 ;        Sets a flag to indicate timer expiration.
                        0B3F  1785 ;
                        0B3F  1786 ;--
                        0B3F  1787
                        0B3F  1788 TIME_OUT:
                   0FFC 0B3F  1789          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0B41  1790
          0002'CF    02  A8  0B41  1791          BISW2   #TEST_OVERM,FLAG                    ; Set the test over flag
56  0230'CF   00000230'8F  C1  0B46  1792          ADDL3   #UNIT_LIST,UNIT_LIST,R6             ; Get the list header
                        0B50  1793 10$:
                    01  E1  0B50  1794          BBC     #UETUNT$V_TESTABLE,-                ; Br if unit is not testable
                 0E 0B  A6       0B52  1795                  UETUNT$B_FLAGS(R6),20$
               10 A6    03  D1  0B55  1796          CMPL    #3,UETUNT$L_ITER(R6)                ; 2 pages done? (starts with 1)
                    08  15  0B59  1797          BLEQ    20$                                 ; BR if more than 2 pages are out
                        0B5B  1798                                                          ; ... must be a fast terminal
          01A6 C6   01  83  0B5B  1799          SUBB3   #1,UETUNT$B_LENGTH(R6),-            ; Set line count equal to page
               01A4 C6      0B60  1800                  UETUNT$B_LINE(R6)                   ; ...length minus one
                        0B63  1801 20$:
               56   66  C0  0B63  1802          ADDL2   (R6),R6                             ; Get the next unit block adr
     00000230'8F    56  D1  0B66  1803          CMPL    R6,#UNIT_LIST                       ; All done checking?
                    E1  12  0B6D  1804          BNEQ    10$                                 ; BR if no
                        0B6F  1805          $SETIMR_S DAYTIM = TWOMIN,-                 ; Every unit should be done
                        0B6F  1806                    ASTADR = WATCHDOG_TIMER           ;  within 2 minutes
                    04  0B82  1807          RET
                        0B83  1808
                        0B83  1809 WATCHDOG_TIMER: ; We should never get here unless there is a hung device.
                        0B83  1810
```

```
                         0FFC  0B83  1811          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                               0B85  1812
    56   0230'CF   00000230'8F   C1  0B85  1813          ADDL3   #UNIT_LIST,UNIT_LIST,R6          ; Get the list header
                               0B8F  1814 10$:
                            01   E1  0B8F  1815          BBC     #UETUNT$V_TESTABLE,-            ; Br if unit is not testable
                      63 0B A6        0B91  1816                  UETUNT$B_FLAGS(R6),20$
                   5E 0B A6   00   E0  0B94  1817          BBS     #UETUNT$V_DONE,UETUNT$B_FLAGS(R6),20$ ; BR if this unit is done
                0098'CF    14 A6   9B  0B99  1818          MOVZBW  UETUNT$T_FILSPC(R6),DEVDSC      ; Setup device name length
                15 A6    0098'CF   28  0B9F  1819          MOVC3   DEVDSC,UETUNT$T_FILSPC+1(R6),-  ; Get device name
                         00B7'CF         0BA5  1820                  DEV_NAME
                                         0BA8  1821          $FAO_S  CTRSTR = TIMOUT_ERR_MSG,-               ; Prepare error message
                                         0BA8  1822                  OUTLEN = BUFFER_PTR,-
                                         0BA8  1823                  OUTBUF = FAO_BUF,-
                                         0BA8  1824                  P1     = #DEVDSC
                         01C6'CF   D6    0BC1  1825          INCL    ERROR_COUNT                     ; Bump the error count
                         000C'CF   DF    0BC5  1826          PUSHAL  BUFFER_PTR                      ; push error message
                   000F0001 8F     DD    0BC9  1827          PUSHL   #^XF0001                        ; push arg count
                   00741132 8F     DD    0BCF  1828          PUSHL   #UETP$_TEXT!STS$K_ERROR         ; push the signal name
                         01C6'CF   DD    0BD5  1829          PUSHL   ERROR_COUNT                     ; ...and the error count...
                         00A0'CF   DF    0BD9  1830          PUSHAL  PROCESS_NAME                    ; ...our own name...
                   00010002 8F     DD    0BDD  1831          PUSHL   #^X10002                        ; ...and the argument count...
                   00748022 8F     DD    0BE3  1832          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR    ; ...and the signal name...
             00000000'GF     07    FB    0BE9  1833          CALLS   #7,G^LIB$SIGNAL                 ; report the error
          0002'CF    0040 8F     A8    0BF0  1834          BISW2   #TIMOUT_ERRM,FLAG                 ; Set time out error flag
                               0BF7  1835 20$:
                      56   66   C0    0BF7  1836          ADDL2   (R6),R6                         ; Get the next unit block adr
          00000230'8F    56   D1    0BFA  1837          CMPL    R6,#UNIT_LIST                   ; All done checking?
                         8C   12    0C01  1838          BNEQ    10$                             ; BR if no
                                    0C03  1839          $WAKE_S
                              04    0C0E  1840          RET
                                    0C0F  1841
```

```
0C0F  1843          .SBTTL  System Service Exception Handler
0C0F  1844 ;++
0C0F  1845 ; FUNCTIONAL DESCRIPTION:
0C0F  1846 ;       This routine is executed if a software or hardware exception occurs or
0C0F  1847 ;       if a LIB$SIGNAL system service is used to output a message.
0C0F  1848 ;
0C0F  1849 ; CALLING SEQUENCE:
0C0F  1850 ;       Entered via an exception from the system
0C0F  1851 ;
0C0F  1852 ; INPUT PARAMETERS:
0C0F  1853 ;       ERROR_COUNT    = previous cumulative error count
0C0F  1854 ;
0C0F  1855 ;          AP ----->      +---------------+
0C0F  1856 ;                         |       2       |
0C0F  1857 ;                         |---------------|
0C0F  1858 ;                         | SIGNL ARY PNT |
0C0F  1859 ;                         |---------------|
0C0F  1860 ;                         | MECH  ARY PNT |
0C0F  1861 ;                         |---------------|  ---------
0C0F  1862 ;                         |       4       |         ^
0C0F  1863 ;                         |---------------|         |
0C0F  1864 ;                         | ESTABLISH FP  |         |
0C0F  1865 ;                         |---------------|         |
0C0F  1866 ;                         |     DEPTH     |  Mecnanism Array
0C0F  1867 ;                         |---------------|         |
0C0F  1868 ;                         |       R0      |         |
0C0F  1869 ;                         |---------------|         v
0C0F  1870 ;                         |       R1      |  ---------
0C0F  1871 ;                         |---------------|         ^
0C0F  1872 ;                         |       N       |         |
0C0F  1873 ;                         |---------------|         |
0C0F  1874 ;                         | CONDITION NAME|         |
0C0F  1875 ;                         |---------------|         |
0C0F  1876 ;                         | N-3 ADDITIONAL|    Signal Array
0C0F  1877 ;                         | LONG WORD ARGS|         |
0C0F  1878 ;                         |---------------|         |
0C0F  1879 ;                         |       PC      |         |
0C0F  1880 ;                         |---------------|         |
0C0F  1881 ;                         |       PSL     |         v
0C0F  1882 ;                         +---------------+  ---------
0C0F  1883 ; IMPLICIT INPUTS:
0C0F  1884 ;       NONE
0C0F  1885 ;
0C0F  1886 ; OUTPUT PARAMETERS:
0C0F  1887 ;       NONE
0C0F  1888 ;
0C0F  1889 ; IMPLICIT OUTPUTS:
0C0F  1890 ;       NONE
0C0F  1891 ;
0C0F  1892 ; COMPLETION CODES:
0C0F  1893 ;       SS$_NORMAL if it's a UETP condition or RMS error.
0C0F  1894 ;       Error status from exception, otherwise.
0C0F  1895 ;
0C0F  1896 ; SIDE EFFECTS:
0C0F  1897 ;       May branch to ERROR_EXIT.
0C0F  1898 ;       May print a message.
0C0F  1899 ;--
```

```
                        0C0F  1900 SSERROR:
                   0FFC 0C0F  1901         .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0C11  1902
                        0C11  1903         $SETAST_S ENBFLG = #0               ; Disable AST delivery
                     01 DD    0C1A  1904         PUSHL   #1                      ; Assume ASTs were enabled
                50   09 D1    0C1C  1905         CMPL    S^#SS$_WASSET,R0        ; Were ASTs enabled?
                     02 13    0C1F  1906         BEQL    10$                     ; BR if they were
                     6E D4    0C21  1907         CLRL    (SP)                    ; Set ASTs to remain disabled
                        0C23  1908 10$:
                        0C23  1909         $SETSFM_S ENBFLG = #0               ; Disable SS failure mode
                     01 DD    0C2C  1910         PUSHL   #1                      ; Assume SS failure mode was enabled
                50   09 D1    0C2E  1911         CMPL    S^#SS$_WASSET,R0        ; Was SS failure mode enabled?
                     02 13    0C31  1912         BEQL    20$                     ; BR if it was
                     6E D4    0C33  1913         CLRL    (SP)                    ; Set SS failure mode to remain off
                        0C35  1914 20$:
           56   04 AC D0      0C35  1915         MOVL    CHF$L_SIGARGLST(AP),R6  ; Get the signal array pointer
           59   04 A6 7D      0C39  1916         MOVQ    CHF$L_SIG_NAME(R6),R9   ; Get NAME in R9 and ARG1 in R10
                10 ED    0C3D  1917         CMPZV   #STS$V_FAC_NO,-         ; Is this a message from LIB$SIGNAL?
                   0C       0C3F  1918                 #STS$S_FAC_NO,-
00000074   8F       59       0C40  1919                 R9,#UETP$_FACILITY
                14 12    0C46  1920         BNEQ    30$                     ; BR if this is not a UETP exception
                66 02    C2    0C48  1921         SUBL2   #2,CHF$L_SIG_ARGS(R6)   ; Drop the PC and PSL
                        0C4B  1922         $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6) ; Print the message
                21 11    0C5A  1923         BRB     40$                     ; Restore ASTs and SS fail mode
                        0C5C  1924 30$:
59   0000045C   8F       D1    0C5C  1925         CMPL    #SS$_SSFAIL,R9          ; RMS failures are SysSvc failures
                32 12    0C63  1926         BNEQ    50$                     ; BR if this can't be an RMS failure
                10 ED    0C65  1927         CMPZV   #STS$V_FAC_NO,-         ; Is it an RMS failure?
                   0C       0C67  1928                 #STS$S_FAC_NO,-
                01 5A       0C68  1929                 R10,#RMS$_FACILITY
                2B 12    0C6A  1930         BNEQ    50$                     ; BR if not
5A   F0000000   8F       CA    0C6C  1931         BICL2   #^XF0000000,R10          ; Strip control bits from status code
      08 A6      04 39    0C73  1932         MATCHC  #4,CHF$L_SIG_ARG1(R6),- ; Is it an RMS failure for which...
                14       0C77  1933                 #NRAT_LENGTH,-
           004D'CF          0C78  1934                 NO_RMS_AST_TABLE        ; ...no AST can be delivered?
                1A 13    0C7B  1935         BEQL    50$                     ; BR if so - must give error here
                        0C7D  1936 40$:
                01 BA    0C7D  1937         POPR    #^M<R0>                 ; Restore SS failure mode...
                        0C7F  1938         $SETSFM_S ENBFLG = R0           ; ...
                01 BA    0C88  1939         POPR    #^M<R0>                 ; Restore AST enable...
                        0C8A  1940         $SETAST_S ENBFLG = R0           ; ...
           50   01 D0    0C93  1941         MOVL    S^#SS$_NORMAL,R0        ; Supply a standard status for exit
                04 0C96  1942         RET                             ; Resume processing (or goto RMS_ERROR)
                        0C97  1943 50$:
      01CA'CF   59 D0    0C97  1944         MOVL    R9,STATUS               ; Save the status
                58 D4    0C9C  1945         CLRL    R8                      ; Assume for now it's not SS failure
59   0000045C   8F       D1    0C9E  1946         CMPL    #SS$_SSFAIL,R9          ; But is it a System Service failure?
                38 12    0CA5  1947         BNEQ    70$                     ; BR if not - no special case message
                        0CA7  1948         $GETMSG_S MSGID = R10,-         ; Get SS failure code associated text
                        0CA7  1949                   MSGLEN = BUFFER_PTR,-
                        0CA7  1950                   BUFADR = FAO_BUF,-
                        0CA7  1951                   FLAGS  = #14,-
                        0CA7  1952                   OUTADR = MSG_BLOCK
           01F7'CF   95 0CBE  1953         TSTB    MSG_BLOCK+1             ; Get FAO arg count for SS failure code
                16 13    0CC2  1954         BEQL    60$                     ; Don't use $GETMSG if no $FAO args...
           000C'CF   DF 0CC4  1955         PUSHAL  BUFFER_PTR              ; ...else build up...
                01 DD    0CC8  1956         PUSHL   #1                      ; ...a message describing...
```

```
                00741130 8F    DD   0CCA  1957          PUSHL   #UETP$_TEXT                     ; ...why the System Service failed
                      00 5A    F0   0CD0  1958          INSV    R10,#STS$V_SEVERITY,-           ; Give the message...
                      6E 03         0CD3  1959                  #STS$S_SEVERITY,(SP)            ; ...the correct severity code
                      58 03    D0   0CD5  1960          MOVL    #3,R8                           ; Count the number of args we pushed
                         05    11   0CD8  1961          BRB     70$
                               0CDA  1962 60$:
                         5A    DD   0CDA  1963          PUSHL   R10                             ; Save SS failure code
                      58 01    D0   0CDC  1964          MOVL    #1,R8                           ; Count the number of args we pushed
                               0CDF  1965 70$:
                57    66 04    C5   0CDF  1966          MULL3   #4,CHF$L_SIG_ARGS(R6),R7 ; Convert longwords to bytes
                   5E 57       C2   0CE3  1967          SUBL2   R7,SP                           ; Save the current signal array...
             6E 04 A6 57       28   0CE6  1968          MOVC3   R7,CHF$L_SIG_NAME(R6),(SP) ; ...on the stack
                7E 66 58       C1   0CEB  1969          ADDL3   R8,CHF$L_SIG_ARGS(R6),-(SP) ; Push the current arg count
                      00A3     31   0CEF  1970          BRW     ERROR_EXIT
```

```
                        0CF2  1972          .SBTTL  RMS Error Handler
                        0CF2  1973 ;++
                        0CF2  1974 ; FUNCTIONAL DESCRIPTION:
                        0CF2  1975 ;       This routine handles error returns from RMS calls.
                        0CF2  1976 ;
                        0CF2  1977 ; CALLING SEQUENCE:
                        0CF2  1978 ;       Called by RMS when a file processing error is found.
                        0CF2  1979 ;
                        0CF2  1980 ; INPUT PARAMETERS:
                        0CF2  1981 ;       The FAB or RAB associated with the RMS call.
                        0CF2  1982 ;
                        0CF2  1983 ; IMPLICIT INPUTS:
                        0CF2  1984 ;       NONE
                        0CF2  1985 ;
                        0CF2  1986 ; OUTPUT PARAMETERS:
                        0CF2  1987 ;       NONE
                        0CF2  1988 ;
                        0CF2  1989 ; IMPLICIT OUTPUTS:
                        0CF2  1990 ;       Error message
                        0CF2  1991 ;
                        0CF2  1992 ; COMPLETION CODES:
                        0CF2  1993 ;       NONE
                        0CF2  1994 ;
                        0CF2  1995 ; SIDE EFFECTS:
                        0CF2  1996 ;       Program may exit, depending on severity of the error.
                        0CF2  1997 ;
                        0CF2  1998 ;--
                        0CF2  1999
                        0CF2  2000 RMS_ERROR:
                  0FFC  0CF2  2001          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0CF4  2002
      56   04 AC    D0  0CF4  2003          MOVL    4(AP),R6                ; See whether we're dealing with...
         66   03    91  0CF8  2004          CMPB    #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
              16    12  0CFB  2005          BNEQ    10$                     ; BR if it's a RAB
   57   02C7'CF    DE  0CFD  2006          MOVAL   FILE,R7                 ; FAB-specific code:  text string...
        58   56    D0  0D02  2007          MOVL    R6,R8                   ; ...address of FAB...
           0C A6    DD  0D05  2008          PUSHL   FAB$L_STV(R6)           ; ...STV field for error...
           08 A6    DD  0D08  2009          PUSHL   FAB$L_STS(R6)           ; ...STS field for error...
01CA'CF    08 A6    D0  0D0B  2010          MOVL    FAB$L_STS(R6),STATUS    ; ...and save the error code
              15    11  0D11  2011          BRB     COMMON                  ; FAB and RAB share other code
                        0D13  2012 10$:
   57   02D3'CF    DE  0D13  2013          MOVAL   RECORD,R7               ; RAB-specific code:  text string...
     58   3C A6    D0  0D18  2014          MOVL    RAB$L_FAB(R6),R8        ; ...address of associated FAB...
           0C A6    DD  0D1C  2015          PUSHL   RAB$L_STV(R6)           ; ...STV field for error...
           08 A6    DD  0D1F  2016          PUSHL   RAB$L_STS(R6)           ; ...STS field for error...
01CA'CF    08 A6    D0  0D22  2017          MOVL    RAB$L_STS(R6),STATUS    ; ...and save the error code
                        0D28  2018 COMMON:
      5A   34 A8    9A  0D28  2019          MOVZBL  FAB$B_FNS(R8),R10       ; Get the file name size
                        0D2C  2020          $FAO_S  CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
                        0D2C  2021                  OUTLEN = BUFFER_PTR,-
                        0D2C  2022                  OUTBUF = FAO_BUF,-
                        0D2C  2023                  P1     = R7,-
                        0D2C  2024                  P2     = R10,-
                        0D2C  2025                  P3     = FAB$L_FNA(R8)
         000C'CF    DF  0D46  2026          PUSHAL  BUFFER_PTR              ; ...and arguments for ERROR_EXIT...
              01    DD  0D4A  2027          PUSHL   #1                      ; ...
   00741130 8F    DD  0D4C  2028          PUSHL   #UETP$_TEXT             ; ...
```

```
                00   EF  0D52  2029          EXTZV   #STS$V_SEVERITY,-
                03       0D54  2030                  #STS$S_SEVERITY,-
          59    01CA'CF  0D55  2031                  STATUS,R9               ; ...get the severity code...
                6E   59  88    0D59  2032    BISB2   R9,(SP)                 ; ...and add it into the signal name
                     05  DD    0D5C  2033    PUSHL   #5                      ; Current arg count
                   0034  31    0D5E  2034    BRW     ERROR_EXIT
```

```
                        0D61  2036          .SBTTL  CTRL/C Handler
                        0D61  2037 ;++
                        0D61  2038 ; FUNCTIONAL DESCRIPTION:
                        0D61  2039 ;       This routine handles CTRL/C AST's
                        0D61  2040 ;
                        0D61  2041 ; CALLING SEQUENCE:
                        0D61  2042 ;       Called via AST
                        0D61  2043 ;
                        0D61  2044 ; INPUT PARAMETERS:
                        0D61  2045 ;       NONE
                        0D61  2046 ;
                        0D61  2047 ; IMPLICIT INPUTS:
                        0D61  2048 ;       NONE
                        0D61  2049 ;
                        0D61  2050 ; OUTPUT PARAMETERS:
                        0D61  2051 ;       NONE
                        0D61  2052 ;
                        0D61  2053 ; IMPLICIT OUTPUTS:
                        0D61  2054 ;       NONE
                        0D61  2055 ;
                        0D61  2056 ; COMPLETION CODES:
                        0D61  2057 ;       NONE
                        0D61  2058 ;
                        0D61  2059 ; SIDE EFFECTS:
                        0D61  2060 ;       NONE
                        0D61  2061 ;
                        0D61  2062 ;--
                        0D61  2063
                        0D61  2064 CCASTHAND:
                  0FFC  0D61  2065          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                        0D63  2066
          00B3'CF    DF 0D63  2067          PUSHAL  CNTRLCMSG               ; Set message pointer
                01    DD 0D67  2068          PUSHL   #1                      ; Set arg count
          00741130    8F DD 0D69  2069      PUSHL   #UETP$_TEXT!STS$K_WARNING ; Set signal name
                00    DD 0D6F  2070          PUSHL   #0                      ; Indicate an abnormal termination
          00A0'CF    DF 0D71  2071          PUSHAL  PROCESS_NAME            ; ...
                02    DD 0D75  2072          PUSHL   #2                      ; ...
          007410E0    8F DD 0D77  2073      PUSHL   #UETP$_ABENDD!STS$K_WARNING ; ...
    00000000'GF    07    FB 0D7D  2074      CALLS   #7,G^LIB$SIGNAL         ; Output the message
                        D0  0D84  2075          MOVL    #<STS$M_INHIB_MSG!-     ; Set the exit status
                        0D85  2076                      SS$_CONTROLC-
                        0D85  2077                      STS$K_SUCCESS+STS$K_WARNING>,-
01CA'CF    10000650    8F       0D85  2078                      STATUS
    0002'CF    0082    8F    A8 0D8D  2079      BISW2   #TEST_OVERM!CTRLC_SEENM,FLAG ; Set termination flags
                        04  0D94  2080          RET
```

```
                        0D95  2082          .SBTTL  Error Exit
                        0D95  2083 ;++
                        0D95  2084 ; FUNCTIONAL DESCRIPTION:
                        0D95  2085 ;       This routine prints an error message and exits.
                        0D95  2086 ;
                        0D95  2087 ; CALLING SEQUENCE:
                        0D95  2088 ;       MOVx  error status value,STATUS
                        0D95  2089 ;       PUSHx error specific information on the stack
                        0D95  2090 ;       PUSHL current argument count
                        0D95  2091 ;       BRW   ERROR_EXIT
                        0D95  2092 ;
                        0D95  2093 ; INPUT PARAMETERS:
                        0D95  2094 ;       Arguments to LIB$SIGNAL, as above
                        0D95  2095 ;
                        0D95  2096 ; IMPLICIT INPUTS:
                        0D95  2097 ;       NONE
                        0D95  2098 ;
                        0D95  2099 ; OUTPUT PARAMETERS:
                        0D95  2100 ;       Message to SYS$OUTPUT and SYS$ERROR
                        0D95  2101 ;
                        0D95  2102 ; IMPLICIT OUTPUTS:
                        0D95  2103 ;       Program exit
                        0D95  2104 ;
                        0D95  2105 ; COMPLETION CODES:
                        0D95  2106 ;       NONE
                        0D95  2107 ;
                        0D95  2108 ; SIDE EFFECTS:
                        0D95  2109 ;       NONE
                        0D95  2110 ;
                        0D95  2111 ;--
                        0D95  2112
                        0D95  2113 ERROR_EXIT:
                        0D95  2114
                        0D95  2115          $SETAST_S ENBFLG = #0              ; Disable AST's
     15 0002'CF    04    E0  0D9E  2116          BBS     #BEGIN_MSGV,FLAG,10$      ; BR if "begin" msg already printed
                     7E    D4  0DA4  2117          CLRL    -(SP)                     ; Set the time stamp flag
                000F'CF    DF  0DA6  2118          PUSHAL  TEST_NAME                 ; Set the test name
                     02    DD  0DAA  2119          PUSHL   #2                        ; Push the argument count
            00741039 8F    DD  0DAC  2120          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
         00000000'GF    04    FB  0DB2  2121          CALLS   #4,G^LIB$SIGNAL           ; Print the startup message
                        0DB9  2122 10$:
        020E'CF    08    8E    C1  0DB9  2123          ADDL3   (SP)+,#8,ARG_COUNT        ; Get total # args, pop partial count
                01C6'CF    D6  0DBF  2124          INCL    ERROR_COUNT               ; Keep running error count
                     00    DD  0DC3  2125          PUSHL   #0                        ; Push the time parameter
                00A0'CF    DF  0DC5  2126          PUSHAL  PROCESS_NAME              ; Push test name...
            000F0002 8F    DD  0DC9  2127          PUSHL   #^XF0002                  ; ...arg count...
            007410E2 8F    DD  0DCF  2128          PUSHL   #UETP$_ABENDD!STS$K_ERROR ; ...and signal name
                01C6'CF    DD  0DD5  2129          PUSHL   ERROR_COUNT               ; Finish off arg list...
                00A0'CF    DF  0DD9  2130          PUSHAL  PROCESS_NAME              ; ...
            00010002 8F    DD  0DDD  2131          PUSHL   #^X10002                  ; ...
            00748022 8F    DD  0DE3  2132          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
   00000000'GF    020E'CF    FB  0DE9  2133          CALLS   ARG_COUNT,G^LIB$SIGNAL    ; Truly bitch
                        0DF2  2134
                01CA'CF    D5  0DF2  2135          TSTL    STATUS                    ; Did we exit with an error code?
                     09    12  0DF6  2136          BNEQ    20$                       ; BR if we did
            007410E2 8F    D0  0DF8  2137          MOVL    #UETP$_ABENDD!STS$K_ERROR,- ; Supply a generic one otherwise
                01CA'CF        0DFE  2138                  STATUS
```

```
                     0E01  2139 20$:
01CA'CF    10000000 8F   C8  0E01  2140        BISL    #STS$M_INHIB_MSG,STATUS ; Don't print messages twice
                     0E0A  2141        $EXIT_S STATUS                          ; Exit in error
```

```
                     ;  J 6
0E15  2143          .SBTTL  Exit Handler
0E15  2144 ;++
0E15  2145 ; FUNCTIONAL DESCRIPTION:
0E15  2146 ;       This routine handles cleanup at exit.  If the MODE logical name is
0E15  2147 ;       equated to "ONE", the routine will update the test flag in the
0E15  2148 ;       UETINIDEV.DAT file depending on the UETUNT$M_TESTABLE flag state in the
0E15  2149 ;       UETUNT$B_FLAGS field of the unit block for each unit for the device
0E15  2150 ;       under test.
0E15  2151 ;
0E15  2152 ; CALLING SEQUENCE:
0E15  2153 ;       Invoked automatically by $EXIT System Service.
0E15  2154 ;
0E15  2155 ; INPUT PARAMETERS:
0E15  2156 ;       STATUS  contains the exit status.
0E15  2157 ;       FLAG    has synchronizing bits.
0E15  2158 ;       DDB_RFA contains the RFA of the DDB record for this device in UETINIDEV.
0E15  2159 ;
0E15  2160 ; IMPLICIT INPUTS:
0E15  2161 ;       UNIT_LIST points to the head of a doubly linked circular list of unit
0E15  2162 ;                 blocks for the device under test.
0E15  2163 ;
0E15  2164 ; OUTPUT PARAMETERS:
0E15  2165 ;       NONE
0E15  2166 ;
0E15  2167 ; IMPLICIT OUTPUTS:
0E15  2168 ;       Various files are de-accessed, the process name is reset, and any
0E15  2169 ;       necessary synchronization with UETPDEV01 is carried out.
0E15  2170 ;       If the MODE logical name is equated to "ONE", the routine will update
0E15  2171 ;       the test flag in the UETINIDEV.DAT file depending on the
0E15  2172 ;       UETUNT$M_TESTABLE flag state in the UETUNT$B_FLAGS field of the unit
0E15  2173 ;       block for each unit for the device under test.
0E15  2174 ;
0E15  2175 ; COMPLETION CODES:
0E15  2176 ;       NONE
0E15  2177 ;
0E15  2178 ; SIDE EFFECTS:
0E15  2179 ;       NONE
0E15  2180 ;
0E15  2181 ;--
0E15  2182
0E15  2183 EXIT_HANDLER:
                OFFC  0E15  2184          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
0E17  2185
0E17  2186          $SETSFM_S ENBFLG = #0           ; Turn off System Service failure mode
0E20  2187          $SETAST_S ENBFLG = #0           ; Disable AST's
0E29  2188          $TRNLOG_S LOGNAM = MODE,-       ; Get the run mode
0E29  2189                    RSLLEN = BUFFER_PTR,-
0E29  2190                    RSLBUF = FAO_BUF
   0014'CF    20   8A  0E42  2191          BICB2   #LC_BITM,BUFFER         ; Convert to upper case
0014'CF    4F 8F   91  0E47  2192          CMPB    #^A/O/,BUFFER           ; Is this a one shot?
              03   13  0E4D  2193          BEQL    10$                     ; BR if yes...
            00C1   31  0E4F  2194          BRW     END_UPDATE              ; ...else don't update UETINIDEV.DAT
0E52  2195 10$:
03 0002'CF    02   E0  0E52  2196          BBS     #SAFE_TO_UPDV,FLAG,20$  ; Only update if it's safe
            00B8   31  0E58  2197          BRW     END_UPDATE              ; Else forget it
0E5B  2198 20$:
03 0002'CF    03   E1  0E5B  2199          BBC     #SELF_TESTV,FLAG,30$    ; Is this a self test?
```

```
                    00AF   31  0E61             BRW     END_UPDATE                   ; BR if so - we don't access UETINIDEV
                               0E64       30$:
          5A   033C'CF     DE  0E64             MOVAL   INI_RAB,R10                  ; Set the RAB address
             1E AA   02    90  0E69             MOVB    #RAB$C_RFA,RAB$B_RAC(R10)    ; Set RFA mode
      10 AA  0380'CF 06    28  0E6D             MOVC3   #6,DDB_RFA,RAB$W_RFA(R10)    ; Set RFA to DDB line
                               0E74             $GET    RAB = (R10)                  ; Go back to the DDB record
                  75 50    E9  0E7D             BLBC    R0,UPDATE_FAILED             ; If failure then forget it
             1E AA   00    90  0E80             MOVB    #RAB$C_SEQ,RAB$B_RAC(R10)    ; Set back to sequential mode
  5B   0230'CF  00000230'8F C1 0E84             ADDL3   #UNIT_LIST,UNIT_LIST,R11     ; Set the unit block list header
                     59    D4  0E8E             CLRL    R9                           ; Init a counter
                               0E90       UNIT_LOOP:
                     01    E1  0E90             BBC     #UETUNT$V_TESTABLE,-         ; BR if this unit is not testable
               02 0B AB        0E92                     UETUNT$B_FLAGS(R11),10$
                     59    D6  0E95             INCL    R9                           ; Count testable units
                               0E97       10$:
                  5B 6B    C0  0E97             ADDL2   (R11),R11                    ; Next unit block
        00000230'8F  5B    D1  0E9A             CMPL    R11,#UNIT_LIST               ; Are we full circle in the list?
                     ED    12  0EA1             BNEQ    UNIT_LOOP                    ; BR if not
                     59    D5  0EA3             TSTL    R9                           ; Any testable units?
                     12    12  0EA5             BNEQ    20$                          ; BR if yes...
          0018'CF  4E 8F   90  0EA7             MOVB    #^A/N/,BUFFER+4              ; ...else disable the DDB record...
                               0EAD             $UPDATE RAB = (R10)                  ; ...here
                  3C 50    E9  0EB6             BLBC    R0,UPDATE_FAILED             ; If error then forget it
                               0EB9       20$:
                  5B 6B    C0  0EB9             ADDL2   (R11),R11                    ; Next unit block
        00000230'8F  5B    D1  0EBC             CMPL    R11,#UNIT_LIST               ; Are we full circle in the list?
                     4E    13  0EC3             BEQL    END_UPDATE                   ; BR if yes
                               0EC5             $GET    RAB = (R10)                  ; Get a record
                  24 50    E9  0ECE             BLBC    R0,UPDATE_FAILED             ; If error then forget it
             0014'CF 20    8A  0ED1             BICB2   #LC_BITM,BUFFER              ; Convert to uppercase
          0014'CF  55 8F   91  0ED6             CMPB    #^A/U/,BUFFER                ; Is it a UCB record?
                     35    12  0EDC             BNEQ    END_UPDATE                   ; BR if not
                     01    E0  0EDE             BBS     #UETUNT$V_TESTABLE,-         ; BR if this unit is testable...
               D6 0B AB        0EE0                     UETUNT$B_FLAGS(R11),20$
          0018'CF  4E 8F   90  0EE3             MOVB    #^A/N/,BUFFER+4              ; ...else disable the UCB record...
                               0EE9             $UPDATE RAB = (R10)                  ; ...here
                  C4 50    E8  0EF2             BLBS    R0,20$                       ; Look at the next record if no error
                               0EF5       UPDATE_FAILED:
                  0C AA    DD  0EF5             PUSHL   RAB$L_STV(R10)               ; Do a simple message...
                     50    DD  0EF8             PUSHL   R0                           ; ...to tell of the failure
                0272'CF    DF  0EFA             PUSHAL  INIDEV_UPDERR
                     01    DD  0EFE             PUSHL   #1
                     00    EF  0F00             EXTZV   #STS$V_SEVERITY,-            ; Copy the severity from RMS status...
             7E   50 03        0F02                     #STS$S_SEVERITY,R0,-(SP)
        6E   00741130 8F   C8  0F05             BISL2   #UETP$_TEXT,(SP)             ; ...to our message
        00000000'GF  05    FB  0F0C             CALLS   #5,G^LIB$SIGNAL
                               0F13       END_UPDATE:
                     00    DD  0F13             PUSHL   #0                           ; Set the time flag
                000F'CF    DF  0F15             PUSHAL  TEST_NAME                    ; Push the test name
                     02    DD  0F19             PUSHL   #2                           ; Push arg count
                     00    EF  0F1B             EXTZV   #STS$V_SEVERITY,-            ; Push the proper exit severity...
                     03        0F1D                     #STS$S_SEVERITY,-
             7E  01CA'CF       0F1E                     STATUS,-(SP)
        6E   00741080 8F   C8  0F22             BISL2   #UETP$_ENDEDD,(SP)           ; ...and use it in our message code
                     04    DD  0F29             PUSHL   #4
                  51 5E    D0  0F2B             MOVL    SP,R1
                               0F2E             $PUTMSG_S MSGVEC = (R1)              ; Output the message
```

```
                                0F3D  2257          $DISCONNECT  RAB = INI_RAB      ; Disconnect the RAB from the FAB
                                0F48  2258          $CLOSE       FAB = INI_FAB      ; Close the file
                                0F53  2259          $SETPRN_S PRCNAM = ACNT_NAME    ; Reset the process name
                     OD 0002'CF 06    E1  0F5E  2260          BBC     #TIMOUT_ERRV,FLAG,10$   ; BR if no units hung
                                00    DD  0F64  2261          PUSHL   #0                      ; We have a hung unit - cancel all
                           0212'CF    DF  0F66  2262          PUSHAL  RMSRUNDWN_BUF           ;  image and indirect I/O so we can
                00000000'GF    02    FB  0F6A  2263          CALLS   #2,G^SYS$RMSRUNDWN      ;  exit without hanging
                                           0F71  2264 10$:
                                04         0F71  2265          RET                             ; That's all folks!
                                           0F72  2266
                                           0F72  2267          .END    UETTTYS00
```

```
$$.TAB                  = 00000428 R      03
$$.TABEND               = 0000046C R      03
$$.TMP                  = 00000801
$$.TMP1                 = 00000001
$$.TMP2                 = 000000CF
$$.TMPX                 = 00000016 R      04
$$.TMPX1                = 0000000D
$$T1                    = 00000000
$$T2                    = 00000006
A                       = 00000065
ACNT_NAME                 00000000 R      02
ALL_SET                   0000041D R      05
ARG_COUNT                 0000020E R      03
BEGIN_MSGM              = 00000010
BEGIN_MSGV              = 00000004
BUFFER                    00000014 R      03
BUFFER_PTR                0000000C R      03
CCASTHAND                 00000D61 R      05
CC_TBL                    0000045A R      02
CC_TBL_SIZE             = 00000011
CHF$L_SIGARGLST         = 00000004
CHF$L_SIG_ARG1          = 00000008
CHF$L_SIG_ARGS          = 00000000
CHF$L_SIG_NAME          = 00000004
CNTRLCMSG                 000000B3 R      02
COMMON                    00000D28 R      05
CONTROLLER                00000031 R      02
CONT_DESC                 000002BF R      02
CR                      = 0000000D
CRLF                      00000C04 R      02
CS1                       00000092 R      02
CS3                       000000A4 R      02
CTRLC_SEENM             = 00000080
CTRLC_SEENV             = 00000007
CUR_UNTBLK                0000024D R      03
DATA_BUF                  0000035A R      02
DATA_DSC                  000000C6 R      03
DC$_LP                  = 00000043
DC$_TERM                = 00000042
DDB_RFA                   00000380 R      03
DEA5_CTRLNAME             000000F4 R      02
DEV$V_SPL               = 00000006
DEV$V_TRM               = 00000002
DEVDEP_SIZE             = 00000013
DEVDSC                    00000098 R      03
DEVNAM_LEN                000001E8 R      03
DEV_NAME                  000000B7 R      03
DIB                       000000CE R      03
DIB$B_DEVCLASS          = 00000004
DIB$B_DEVTYPE           = 00000005
DIB$K_LENGTH            = 00000074
DIB$L_DEVCHAR           = 00000000
DIB$L_DEVDEPEND         = 00000008
DIB$W_DEVBUFSIZ         = 00000006
DIBBUF                    000000D6 R      03
DIBBUF_SEC                00000152 R      03
DIB_SEC                   0000014A R      03
DT$_LA11                = 00000002
DT$_LA120               = 00000021
DT$_LA180               = 00000003
DT$_LA34                = 00000022
DT$_LA36                = 00000020
DT$_LA38                = 00000023
DT$_LAX                 = 00000020
DT$_LP11                = 00000001
DT$_TTYUNKN             = 00000000
DT$_VT05                = 00000001
DT$_VT100               = 00000060
DT$_VT101               = 00000061
DT$_VT102               = 00000062
DT$_VT125               = 00000064
DT$_VT131               = 00000065
DT$_VT132               = 00000066
DT$_VT52                = 00000040
DT$_VT55                = 00000041
DT$_VT5X                = 00000040
DUMMY_FAB                 000003D8 R      03
DUMMY_RAB                 00000428 R      03
DVI$_DEVDEPEND2         = 0000001C
DVI$_DEVNAM             = 00000020
EFN2                    = 00000004
END_UPDATE                00000F13 R      05
ERROR_COUNT               000001C6 R      03
ERROR_EXIT                00000D95 R      05
ESC                     = 0000001B
EXIT_DESC                 000001FE R      03
EXIT_HANDLER              00000E15 R      05
FAB$B_BID               = 00000000
FAB$B_FNS               = 00000034
FAB$C_BID               = 00000003
FAB$C_BLN               = 00000050
FAB$C_SEQ               = 00000000
FAB$C_VAR               = 00000002
FAB$C_VFC               = 00000003
FAB$L_ALQ               = 00000010
FAB$L_DEV               = 00000040
FAB$L_FNA               = 0000002C
FAB$L_FOP               = 00000004
FAB$L_STS               = 00000008
FAB$L_STV               = 0000000C
FAB$V_BRO               = 00000006
FAB$V_CHAN_MODE         = 00000002
FAB$V_CR                = 00000001
FAB$V_FILE_MODE         = 00000004
FAB$V_GET               = 00000001
FAB$V_LNM_MODE          = 00000000
FAB$V_PRN               = 00000002
FAB$V_PUT               = 00000000
FAB$V_UFO               = 00000011
FAB$V_UPD               = 00000003
FAB$V_UPI               = 00000006
FAB$W_GBC               = 00000048
FAO_BUF                   00000004 R      03
FF                      = 0000000C
```

```
FILE                    000002C7 R      02
FIND_IT                 0000023D R      05
FIVESEC                 000002A7 R      02
FLAG                    00000002 R      03
FORM_FEED               00000C02 R      02
FOUND_IT                000002D5 R      05
GET_NODE                00000771 R      05
HDR_OUT_SIZE          = 00000100
HEAD_BUF                00000240 R      03
HEAD_CTRSTR             000001C8 R      02
HEAD_LENGTH             00000248 R      03
HUNG_TERMINAL           00000302 R      02
ILLEGAL_REC             00000161 R      02
INADDRESS               000001D6 R      03
INIDEV_UPDERR           00000272 R      02
INI_FAB                 000002EC R      03
INI_RAB                 0000033C R      03
INPUT_ITMLST            00000072 R      02
IO$M_CTRLCAST           ********     X  05
IO$_SETMODE             ********     X  05
IO$_WRITEVBLK           ********     X  05
ITERATION               000001F2 R      03
LA11                    000004F9 R      02
LA11_DATA               00000BE1 R      02
LA120                   000004C7 R      02
LA120_DATA              00000BC9 R      02
LA180                   000004F3 R      02
LA180_DATA              00000BD9 R      02
LA34                    000004CD R      02
LA36                    000004D2 R      02
LA36_DATA               00000BD1 R      02
LA38                    000004D7 R      02
LAX                     00000503 R      02
LAX_DATA                00000BFA R      02
LC_BITM               = 00000020
LF                    = 0000000A
LFCR                    00000C06 R      02
LIB$SIGNAL              ********     X  05
LP11                    000004FE R      02
LP11_DATA               00000BF2 R      02
MAX_DEV_DESIG         = 0000000A
MAX_NAME_SIZE         = 00000007
MAX_PROC_NAME         = 0000000F
MAX_UNIT_DESIG        = 00000005
MODE                    00000041 R      02
MSG_BLOCK               000001F6 R      03
NAME_LEN              = 0000000F
NEW_NODE                00000238 R      03
NEW_PAGE                0000097D R      05
NEXT_TTY                00000669 R      05
NOSPOOLED               00000321 R      02
NOUNIT_SELECTED         0000013B R      02
NO_CTRLNAME             000000D4 R      02
NO_RMS_AST_TABLE        0000004D R      02
NRAT_LENGTH           = 00000014
ONESHOT_ERROR           00000678 R      05
ONE_SHOT                00000599 R      05
ONE_SHOT_LOOP           000005A3 R      05
OTS$CVT_TI_L            ********     X  05
OUTADDRESS              000001DE R      03
PAGES                 = 00000005
PASS                    000001FA R      03
PASS_MSG                00000195 R      02
PC1...                = 00000640 R      02
PC2...                = 00000507 R      02
PMTSIZ                = 00000019
PROCESS_NAME            000000A0 R      03
PROCESS_NAME_FREE     = 0000000B
PROC_CONT_NAME          000000D4 R      05
PROMPT                  00000259 R      02
QUAD_STATUS             000001CE R      03
RAB$B_PSZ             = 00000034
RAB$B_RAC             = 0000001E
RAB$C_BID             = 00000001
RAB$C_BLN             = 00000044
RAB$C_RFA             = 00000002
RAB$C_SEQ             = 00000000
RAB$L_CTX             = 00000018
RAB$L_FAB             = 0000003C
RAB$L_PBF             = 00000030
RAB$L_RBF             = 00000028
RAB$L_RHB             = 0000002C
RAB$L_ROP             = 00000004
RAB$L_STS             = 00000008
RAB$L_STV             = 0000000C
RAB$M_BIO             = 00000800
RAB$V_ASY             = 00000000
RAB$V_BIO             = 0000000B
RAB$V_PMT             = 0000001E
RAB$W_RFA             = 00000010
RAB$W_RSZ             = 00000022
RANDOM1                 000001EA R      03
RANDOM2                 000001EE R      03
RECORD                  000002D3 R      02
REC_SIZE              = 00000028
RESTART                 00000489 R      05
RMS$_BLN                ********     X  02
RMS$_BUSY               ********     X  02
RMS$_CDA                ********     X  02
RMS$_FAB                ********     X  02
RMS$_FACILITY         = 00000001
RMS$_RAB                ********     X  02
RMSRUNDWN_BUF           00000212 R      03
RMS_ERROR               00000CF2 R      05
RMS_ERR_STRING          000002E1 R      02
RUNDWN_BUF              0000021A R      03
SAFE_TO_UPDM          = 00000004
SAFE_TO_UPDV          = 00000002
SAVE_PC...            = 00000640 R      02
SCROL_CLRM            = 00000020
SCROL_CLRV            = 00000005
SEC$M_EXPREG          = 00020000
SEC$M_GBL             = 00000001
SELF_TESTM            = 00000008
```

```
SELF_TESTV                      = 00000003
SERVICE_IO                        00000977 R      05
SERVICE_IO1                       000009A1 R      05
SERVICE_IO2                       000009EB R      05
SERVICE_IO3                       00000A15 R      05
SET_DEVDEP                        000007D4 R      05
SHR$_ABENDD                     = 000010E0
SHR$_BEGIND                     = 00001038
SHR$_ENDEDD                     = 00001080
SHR$_OPENIN                     = 00001098
SHR$_TEXT                       = 00001130
SIZE_TBL                          00000352 R      02
SIZE_TBL_LEN                    = 00000008
SPARE                             0000024C R      03
SPL_UNITM                       = 00000080
SPL_UNITV                       = 00000007
SS$_ABORT                       = 0000002C
SS$_BADPARAM                    = 00000014
SS$_CANCEL                      = 00000830
SS$_CONTROLC                    = 00000651
SS$_NORMAL                      = 00000001
SS$_NOSUCHSEC                   = 00000978
SS$_SSFAIL                      = 0000045C
SS$_WASSET                      = 00000009
SSERROR                           00000C0F R      05
SS_SYNCH_EFN                    = 00000003
STATUS                            000001CA R      03
STR$UPCASE                        ********    X   05
STS$K_ERROR                     = 00000002
STS$K_INFO                      = 00000003
STS$K_SUCCESS                   = 00000001
STS$K_WARNING                   = 00000000
STS$M_INHIB_MSG                 = 10000000
STS$S_FAC_NO                    = 0000000C
STS$S_SEVERITY                  = 00000003
STS$V_FAC_NO                    = 00000010
STS$V_SEVERITY                  = 00000000
SUC_EXIT                          000006F1 R      05
SUPDEV_GBLSEC                     00000020 R      02
SUP_FAB                           00000388 R      03
SYS$ASSIGN                        ********  GX    05
SYS$CANCEL                        ********  GX    05
SYS$CANTIM                        ********  GX    05
SYS$CLOSE                         ********  GX    05
SYS$CONNECT                       ********  GX    05
SYS$CREATE                        ********  GX    05
SYS$CRMPSC                        ********  GX    05
SYS$DCLEXH                        ********  GX    05
SYS$DISCONNECT                    ********  GX    05
SYS$EXIT                          ********  GX    05
SYS$EXPREG                        ********  GX    05
SYS$FAO                           ********   X    05
SYS$GET                           ********  GX    05
SYS$GETDEV                        ********  GX    05
SYS$GETDVI                        ********  GX    05
SYS$GETMSG                        ********  GX    05
SYS$HIBER                         ********  GX    05
SYS$INPUT                         00000061 R      02
SYS$MGBLSC                        ********  GX    05
SYS$OPEN                          ********  GX    05
SYS$PUT                           ********  GX    05
SYS$PUTMSG                        ********  GX    05
SYS$QIOW                          ********  GX    05
SYS$RMSRUNDWN                     ********   X    05
SYS$SETAST                        ********  GX    05
SYS$SETIMR                        ********  GX    05
SYS$SETPRN                        ********  GX    05
SYS$SETSFM                        ********  GX    05
SYS$TRNLOG                        ********  GX    05
SYS$UPDATE                        ********  GX    05
SYS$WAIT                          ********  GX    05
SYS$WAITFR                        ********  GX    05
SYS$WAKE                          ********  GX    05
SYS$WRITE                         ********  GX    05
SYSIN_FAB                         00000258 R      03
SYSIN_RAB                         000002A8 R      03
TERM_ITMLST                       00000082 R      02
TERM_NAMES                        0000049E R      02
TERM_TYPE_CNT                   = 00000013
TESDEV_TBL                        00000523 R      02
TESDEV_TBL_END                  = 00000640 R      02
TEST_NAME                         0000000F R      02
TEST_OVERM                      = 00000002
TEST_OVERV                      = 00000001
TEXT_BUFFER                     = 00000084
THREEMIN                          00000297 R      02
TIMED_OUT                         000006DE R      05
TIME_IT                           0000046C R      05
TIME_OUT                          00000B3F R      05
TIME_OUT_MSG                      0000020C R      02
TIMOUT_ERRM                     = 00000040
TIMOUT_ERRV                     = 00000006
TIMOUT_ERR_MSG                    00000234 R      02
TSD_SIZE                        = 00000600
TT$M_LOWER                      = 00000080
TT2$M_ANSICRT                   = 01000000
TTCHAN                            00000000 R      03
TWOMIN                            0000029F R      02
TWOSEC                            000002AF R      02
UETP                            = 00740000
UETP$_ABENDD                    = 007410E0
UETP$_ABORTC                    = 0074832B
UETP$_BEGIND                    = 00741038
UETP$_DENOSU                    = 00748333
UETP$_DEUNUS                    = 0074819A
UETP$_ENDEDD                    = 00741080
UETP$_ERBOXPROC                 = 00748020
UETP$_FACILITY                  = 00000074
UETP$_OPENIN                    = 00741098
UETP$_TEXT                      = 00741130
UETTTYS00                         00000000 RG     05
UETUNT$B_FLAGS                  = 0000000B
UETUNT$B_HD_LEN                 = 000001A5
UETUNT$B_LENGTH                 = 000001A6
```

```
UETUNT$B_LINE                   = 000001A4
UETUNT$B_TYPE                   = 00000008
UETUNT$C_DEVDEP                 = 000001A4
UETUNT$C_FAB                    = 00000110
UETUNT$C_INDSIZ                 = 000001A4
UETUNT$K_CCTBL                  = 000001AF
UETUNT$K_FAB                    = 00000110
UETUNT$K_HEADER                 = 000002B8
UETUNT$K_RAB                    = 00000160
UETUNT$K_WRITE                  = 000001B7
UETUNT$L_CURHDR                 = 000001AB
UETUNT$L_ITER                   = 00000010
UETUNT$M_2PL                    = 00000004
UETUNT$M_DONE                   = 00000001
UETUNT$M_TESTABLE               = 00000002
UETUNT$T_FILSPC                 = 00000014
UETUNT$V_2PL                    = 00000002
UETUNT$V_DONE                   = 00000000
UETUNT$V_TESTABLE               = 00000001
UETUNT$W_CHAN                   = 0000000C
UETUNT$W_SIZE                   = 00000009
UETUNT$W_WIDTH                  = 000001A7
UNIT_DESC                         000002B7 R      02
UNIT_LIST                         00000230 R      03
UNIT_LOOP                         00000E90 R      05
UNIT_NUMBER                       000001E6 R      03
UNKNOWN                           000004EB R      02
UNKNOWN1                          000005F5 R      02
UNKN_DATA                         00000BEA R      02
UNKN_PREAMBLE                     00000BE9 R      02
UPDATE_FAILED                     00000EF5 R      05
VOSL                            = 0000000E
VT05                              000004E6 R      02
VT05_DATA                         00000BC1 R      02
VT05_PREAMBLE                     00000BBB R      02
VT05_PRE_LEN                    = 00000005
VT100                             0000049E R      02
VT100L                          = 000003E1
VT100_DATA                        00000640 R      02
VT100_ORIG_SCROL                  00000A38 R      02
VT100_OUT_DATA                    00000657 R      02
VT100_PREAMBLE                    00000648 R      02
VT100_PRE_LEN                   = 0000000E
VT101                             000004A4 R      02
VT102                             000004AA R      02
VT125                             000004B0 R      02
VT131                             000004B6 R      02
VT132                             000004BC R      02
VT52                              000004C2 R      02
VT52L                           = 00000168
VT52_DATA                         00000A46 R      02
VT52_OUT_DATA                     00000A53 R      02
VT52_PREAMBLE                     00000A4E R      02
VT52_PRE_LEN                    = 00000004
VT55                              000004DC R      02
VT5X                              000004E1 R      02
WATCHDOG_TIMER                    00000B83 R      05
WRITE_SIZE                      = 00000101
XTERM_CHAR                        00000251 R      03
```

. Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
RODATA	00000C08 (3080.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
RWDATA	0000046C (1132.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
$RMSNAM	00000023 (35.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
TTYS	00000F72 (3954.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE

Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	28	00:00:00.09	00:00:01.15
Command processing	108	00:00:00.74	00:00:06.80
Pass 1	714	00:00:30.01	00:01:05.94
Symbol table sort	1	00:00:02.79	00:00:05.85
Pass 2	982	00:00:08.33	00:00:18.56
Symbol table output	53	00:00:00.36	00:00:00.53
Psect synopsis output	6	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1894	00:00:42.36	00:01:38.90

The working set limit was 900 pages.
172877 bytes (338 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1868 non-local and 69 local symbols.
2267 source lines were read in Pass 1, producing 47 object records in Pass 2.
71 pages of virtual memory were used to define 63 macros.

! Macro library statistics !

Macro library name	Macros defined
_255DUA28:[UETP.OBJ]UETP.MLB;1	2
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	57
TOTALS (all libraries)	59

2145 GETS were required to define 59 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:UETTTYS00/OBJ=OBJ$:UETTTYS00 MSRC$:UETTTYS00/UPDATE=(ENH$:UETTTYS00)+EXECML$/LIB+LIB$:UETP/LIB
```