K 1

```
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  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
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP
UUUUUUUUUUUUUUU  EEEEEEEEEEEEEEE        TTT        PPP
```

FILEID**UETTTYS00


```
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
```

Table of contents

```
0000            .TITLE UETTTYS00 VAX/VMS UETP DEVICE TEST FOR TERMINALS
0000            .IDENT  'V04-000'
0000            .ENABLE SUPPRESSION
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000
0000    ;++
0000    ; FACILITY:
0000    ;       This module will be distributed with VAX/VMS under the [SYSTEST]
0000    ;       account.
0000    ;
0000    ; ABSTRACT:
0000    ;       This program tests terminal devices  etc.
0000    ;
0000    ; ENVIRONMENT:
0000    ;       This program will run in user access mode, with interrupts enabled
0000    ;       at all times.  This program requires the following privileges and
0000    ;       quotas:
0000    ;
0000    ;--
0000    ;
0000    ; AUTHOR: Larry D. Jones,        CREATION DATE: January, 1981
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-009 RNH0008         Richard N. Holstein,    01-May-1984
0000    ;               Fix so that CTRL/C really stops the test when run in loop mode.
0000    ;               Fix to preserve R3 across MOVC3.  Fix to prevent RMS invalid
0000    ;               RAB when run in loop mode.
0000    ;
0000    ;       V03-008 RNH0007         Richard N. Holstein,    15-Feb-1984
0000    ;               Take advantage of new UETP message codes.  Fix SSERROR
0000    ;               interaction with RMS_ERROR.
0000    ;
0000    ;       V03-007 RNH0006         Richard N. Holstein,    19-Dec-1983
```

```
0000   ;               Give correct sentinels to Test Controller.
0000   ;
0000   ;       V03-006 RNH0005         Richard N. Holstein,    11-Nov-1983
0000   ;               Use decimal conversion routine for unit numbers.
0000   ;
0000   ;       V03-005 RNH0004         Richard N. Holstein,    11-Mar-1983
0000   ;               Don't signal ending message in EXIT_HANDLER.
0000   ;
0000   ;       V03-004 RNH0003         Richard N. Holstein,    28-Feb-1983
0000   ;               Allow for longer device names.
0000   ;
0000   ;       V03-003 LDJ0004         Larry D. Jones,         26-Feb-1983
0000   ;               Added generic support for ANSI standard terminals.
0000   ;
0000   ;       V03-002 RNH0002         Richard N. Holstein,    15-Oct-1982
0000   ;               Miscellaneous fixes listed in the V3B UETP Workplan.
0000   ;
0000   ;       V03-001 LDJ0003         Larry D. Jones,         17-Jun-1982
0000   ;               Fixed VT100 on line 5 exceeded quota bug.
0000   ;
0000   ;       V02-006 RNP0003         Robert N. Perron,       22-Jan-1982
0000   ;               Added watchdog timer to prevent test from hanging if we
0000   ;               get a hung device.
0000   ;
0000   ;       V02-005 RNP0002         Robert N. Perron,       31-Dec-1981
0000   ;               Fixed problem with terminals that have network logical
0000   ;               links.
0000   ;
0000   ;       V02-004 RNP0001         Robert N. Perron,       23-Dec-1981
0000   ;               Added two second pause after each page of output to give
0000   ;               other device tests a chance to run.
0000   ;
0000   ;       V02-003 RNH0001         Richard N. Holstein,    09-Oct-1981
0000   ;               Use secondary device characteristics to get around problems
0000   ;               when a device is spooled.  Don't test spooled devices.
0000   ;
0000   ;       V02-002 LDJ0002         Larry D. Jones,         29-Sep-1981
0000   ;               Added support for LA34, LA38, VT101, VT102 and VT131. Fixed
0000   ;               problem with disabling all lines on one unsupported terminal.
0000   ;               Fixed problem with VT100 one shot mode leaving terminal char.
0000   ;               set wrong.
0000   ;
0000   ;       V02-001 LDJ0001         Larry D. Jones,         22-Sep-1981
0000   ;               Fixed possible hang problem when running UETP from other than
0000   ;               the console.
0000   ;
0000   ;**
```

```
                    0000          .SBTTL  Declarations
                    0000     ;
                    0000     ; INCLUDE FILES:
                    0000     ;
                    0000     ;        SYS$LIBRARY:LIB.MLB       for general definitions
                    0000     ;        SHRLIB$:UETP.MLB          for UETP definitions
                    0000
                    0000     ;
                    0000     ; MACROS:
                    0000     ;
                    0000          $CHFDEF                       ; Condition handler frame definitions
                    0000          $DCDEF                        ; Device characteristics definitions
                    0000          $DEVDEF                       ; Device definitions
                    0000          $DIBDEF                       ; Device Information Block
                    0000          $DVIDEF                       ; $GETDVI ITMLST item codes
                    0000          $SECDEF                       ; Section definitions
                    0000          $SHRDEF                       ; Shared messages
                    0000          $SSDEF                        ; System Service status codes
                    0000          $STSDEF                       ; Status return
                    0000          $TTDEF                        ; Terminal definitions
                    0000          $TT2DEF                       ; Extended terminal definitions
                    0000          $UETUNTDEF                    ; UETP unit block offset definitions
                    0000          $UETPDEF                      ; UETP
                    0000     ;
                    0000     ;
                    0000          .MACRO TERMINAL,TERM_NAME,CLASS,TYPE,PREAMBLE,TERM_DATA,HEAD_LEN
                    0000               .=PC1...                 ; PC of terminal type table
                    0000               .BYTE   CLASS,TYPE,HEAD_LEN ; TERM_NAME
                    0000               .ADDRESS PC2...          ; ASCIC name address
                    0000               .ADDRESS PREAMBLE        ; ASCIC preamble address
                    0000               .ADDRESS TERM_DATA       ; ASCID terminal specific data address
                    0000               PC1...=PC1...+15         ; bump to the next address
                    0000               .=PC2...                 ; point to the next ASCIC msg
                    0000                                        ; make it's label and function ID
                    0000     TERM_NAME:
                    0000               .ASCIC /TERM_NAME/
                    0000               PC2...=.                 ; update the string PC
                    0000          .ENDM TERMINAL
                    0000     ;
                    0000     ; EQUATED SYMBOLS:
                    0000     ;
                    0000     ;   Facility number definitions:
00000001            0000          RMS$_FACILITY = 1
                    0000
                    0000     ;   SHR message definitions:
00740000            0000          UETP = UETP$_FACILITY@STS$V_FAC_NO ; Define the UETP facility code
007410E0            0000          UETP$_ABENDD = UETP!SHR$_ABENDD ; Define the UETP message codes
00741038            0000          UETP$_BEGIND = UETP!SHR$_BEGIND
00741080            0000          UETP$_ENDEDD = UETP!SHR$_ENDEDD
00741098            0000          UETP$_OPENIN = UETP!SHR$_OPENIN
00741130            0000          UETP$_TEXT   = UETP!SHR$_TEXT
                    0000
                    0000     ;   Internal flag bits...:
00000001            0000          TEST_OVERV   = 1              ; Set when test is over
00000002            0000          SAFE_TO_UPDV = 2              ; Set if it's safe to update UETINIDEV
00000003            0000          SELF_TESTV   = 3              ; Set when only user terminal is tested
00000004            0000          BEGIN_MSGV   = 4              ; Set if "BEGIN" msg has been printed
```

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

```
                 0000   220         DEVDEP_SIZE+-            ; ...which make up a UETP unit block...
                 0000   221         WRITE_SIZE+-             ; ...to give to the $EXPREG service below
                 0000   222         TSD_SIZE+-
        00000005 0000   223         511>/512>
                 0000   224
```

```
                                        0000          .SBTTL  Read-Only Data
                                    00000000          .PSECT  RODATA,NOEXE,NOWRT,PAGE
                                        0000
                                        0000  ACNT_NAME:                                      ; Process name on exit
53 45 54 53 59 53 00000008'010E0000'    0000          .ASCID  /SYSTEST/
                                  54    000E
                                        000F
                                        000F  TEST_NAME:                                      ; This test name
59 54 54 54 45 55 00000017'010E0000'    000F          .ASCID  /UETTTYS00/
                            30 30 53    001D
                                        0020
                                        0020  SUPDEV_GBLSEC:                                  ; How we access UETSUPDEV.DAT
50 55 53 54 45 55 00000028'010E0000'    0020          .ASCID  /UETSUPDEV/
                            56 45 44    002E
                                        0031
                                        0031  CONTROLLER:                                     ; Logical name of controller
41 4E 4C 52 54 43 00000039'010E0000'    0031          .ASCID  /CTRLNAME/
                               45 4D    003F
                                        0041
                                        0041  MODE:                                           ; Run mode logical name
      45 44 4F 4D 00000049'010E0000'    0041          .ASCID  /MODE/
                                        004D
                                        004D  NO_RMS_AST_TABLE:                               ; List of errors for which...
                           00000000'    004D          .LONG   RMS$_BLN                        ; ...RMS cannot deliver an AST...
                           00000000'    0051          .LONG   RMS$_BUSY                       ; ...even if one has an ERR= arg
                           00000000'    0055          .LONG   RMS$_CDA                        ; Note that we can search table...
                           00000000'    0059          .LONG   RMS$_FAB                        ; ...via MATCHC since <31:16>...
                           00000000'    005D          .LONG   RMS$_RAB                        ; ...pattern can't be in <15:0>
                            00000014    0061  NRAT_LENGTH = .-NO_RMS_AST_TABLE
                                        0061
                                        0061  SYS$INPUT:                                      ; Name of device from which...
4E 49 24 53 59 53 00000069'010E0000'    0061          .ASCID  /SYS$INPUT/                     ; ...the test can be aborted
                            54 55 50    006F
                                        0072
                                        0072  INPUT_ITMLST:                                   ; $GETDVI arg list for SYS$INPUT
                           0020 0040    0072          .WORD   64,DVI$_DEVNAM                  ; We need the equivalence name
                0000000C'00000014'      0076          .LONG   BUFFER,BUFFER_PTR
                            00000000    007E          .LONG   0                               ; Terminate the list
                                        0082
                                        0082  TERM_ITMLST:                                    ; $GETDVI arg list for terminal
                           001C 0004    0082          .WORD   4,DVI$_DEVDEPEND2               ; We need the extended term. char.
                 00000000 00000251'     0086          .LONG   XTERM_CHAR,0
                            00000000    008E          .LONG   0                               ; Terminate the list
                                        0092
                                        0092  CS1:                                            ; Device class and type control string
21 20 42 58 32 21 0000009A'010E0000'    0092          .ASCID  /!2XB !2XB /
                         20 42 58 32    00A0
                                        00A4
                                        00A4  CS3:                                            ; Device class-only control string
2A 20 42 58 32 21 000000AC'010E0000'    00A4          .ASCID  /!2XB **/
                                  2A    00B2
                                        00B3
                                        00B3  CNTRLCMSG:
65 74 72 6F 62 41 000000BB'010E0000'    00B3          .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
```

```
                                           00D4      NO_CTRLNAME:
6E 6F 63 20 6F 4E 000000DC'010E0000'       00D4              .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
                                           00F4      DEAD_CTRLNAME:
20 74 27 6E 61 43 000000FC'010E0000'       00F4              .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
                                           013B      NOUNIT_SELECTED:
69 6E 75 20 6F 4E 00000143'010E0000'       013B              .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
                                           0161      ILLEGAL_REC:
61 67 65 6C 6C 49 00000169'010E0000'       0161              .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
                                           0195      PASS_MSG:
66 6F 20 64 6E 45 0000019D'010E0000'       0195              .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
                                           01C8      HEAD_CTRSTR:
58 41 56 43 41 21 000001D0'010E0000'       01C8              .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
                                           020C      TIME_OUT_MSG:
65 63 69 76 65 44 00000214'010E0000'       020C              .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
                                           0234      TIMOUT_ERR_MSG:
6F 2D 65 6D 69 54 0000023C'010E0000'       0234              .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
                                           0259      PROMPT:
64 20 72 65 6C 6C 6F 72 74 6E 6F 43        0259              .ASCII  /Controller designation?: /
3A 3F 6E 6F 69 74 61 6E 67 69 73 65        0265
                                     20    0271
                           00000019        0272              PMTSIZ = .-PROMPT
                                           0272
```

```
                                         0272  INIDEV_UPDERR:                              ; Error during exit handler
20 72 6F 72 72 45 0000027A'010E0000'     0272          .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
                                         0297  THREEMIN:                                   ; 3 minute delta time
                  FFFFFFFF 94B62E00      0297          .LONG   -10*1000*1000*180,-1
                                         029F
                                         029F  TWOMIN:                                     ; 2 minute delta time
                  FFFFFFFF B8797400      029F          .LONG   -10*1000*1000*120,-1
                                         02A7
                                         02A7  FIVESEC:                                    ; 5 second delta time
                  FFFFFFFF FD050F80      02A7          .LONG   -10*1000*1000*5,-1
                                         02AF
                                         02AF  TWOSEC:                                     ; 2 seconds delta time
                  FFFFFFFF FECED300      02AF          .LONG   -10*1000*1000*2,-1
                                         02B7
                                         02B7  UNIT_DESC:                                  ; Descriptor used to convert unit #
                           00000005      02B7          .LONG   5
                           0000001A'     02BB          .ADDRESS BUFFER+6
                                         02BF
                                         02BF  CONT_DESC:                                  ; Descriptor used to convert controller...
                          0000 0028      02BF          .WORD   REC_SIZE,0                  ; ...from lowercase to uppercase
                           00000014'     02C3          .ADDRESS BUFFER
                                         02C7
                                         02C7  FILE:                                       ; Fills in RMS_ERR_STRING
      65 6C 69 66 000002CF'010E0000'     02C7          .ASCID  /file/
                                         02D3
                                         02D3  RECORD:                                     ; Fills in RMS_ERR_STRING
64 72 6F 63 65 72 000002DB'010E0000'     02D3          .ASCID  /record/
                                         02E1
                                         02E1  RMS_ERR_STRING:                             ; Announces an RMS error
41 21 20 53 4D 52 000002E9'010E0000'     02E1          .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
                                         0302  HUNG_TERMINAL:
59 54 54 54 45 55 0000030A'010E0000'     0302          .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
                                         0321  NOSPOOLED:                                  ; We found a spooled device
65 6C 62 61 6E 55 00000329'010E0000'     0321          .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
                                         0352  SIZE_TBL:                                   ; table of output record sizes
                               0000      0352          .WORD   0                           ; line size added in later
                               0000      0354          .WORD   0
                               0002      0356          .WORD   2
                               0001      0358          .WORD   1
                           00000008      035A          SIZE_TBL_LEN=.-SIZE_TBL
                                         035A
                                         035A  DATA_BUF:                                   ; 256 bytes of all printable characters
                           00000021      035A          A=^A/!/
                                         035A          .REPT 94
```

```
              035A        .BYTE A
              035A        A=A+1
          21  035A        .ENDR
    00000021  03B8        A=^A/!/
              03B8        .REPT 94
              03B8        .BYTE A
              03B8        A=A+1
          21  03B8        .ENDR
    00000021  0416        A=^A/!/
```

```
                        045A    ;
                        045A    ; Carriage control longword format.
                        045A    ;
                        045A    ;       +----------+----------+----------+----------+
                        045A    ;       ! postfix  !  prefix  ! data Type! Fortran  !
                        045A    ;       +----------+----------+----------+----------+
                        045A    ;
                        045A    ;
                        045A    CC_TBL:
         00 01 01 00    045A            .BYTE   0,1,1,0                         ; Prefix newline PosDfix null
         01 01 01 00    045E            .BYTE   0,1,1,1                         ; Prefix-Postfix newline
         00 00 02 00    0462            .BYTE   0,2,0,0                         ; Prefix-Postfix null
         01 00 01 00    0466            .BYTE   0,1,0,1                         ; Prefix null Postfix newline
         00 8D 02 00    046A            .BYTE   0,2,^X8D,0                      ; Prefix 7 bit Postfix null
         00 CD 02 00    046E            .BYTE   0,2,^XCD,0                      ; Prefix 8 bit Postfix null
         01 8D 01 00    0472            .BYTE   0,1,^X8D,1                      ; Prefix 7 bit Postfix newline
         01 CD 01 00    0476            .BYTE   0,1,^XCD,1                      ; Prefix 8 bit Postfix newline
         8D 00 01 00    047A            .BYTE   0,1,0,^X8D                      ; Prefix null Postfix 7 bit
         8D 01 03 00    047E            .BYTE   0,3,1,^X8D                      ; Prefix newline Postfix 7 bit
         8D 8D 03 00    0482            .BYTE   0,3,^X8D,^X8D                   ; Prefix 7 bit Postfix 7 bit
         8D CD 03 00    0486            .BYTE   0,3,^XCD,^X8D                   ; Prefix 8 bit Postfix 7 bit
         CD 00 01 00    048A            .BYTE   0,1,0,^XCD                      ; Prefix null Postfix 8 bit
         CD 01 03 00    048E            .BYTE   0,3,1,^XCD                      ; Prefix newline Postfix 8 bit
         CD 8D 03 00    0492            .BYTE   0,3,^X8D,^XCD                   ; Prefix 7 bit Postfix 8 bit
         CD CD 02 00    0496            .BYTE   0,2,^XCD,^XCD                   ; Prefix 8 bit Postfix 8 bit
         00 00 01 00    049A            .BYTE   0,1,0,0                         ; Prefix-Postfix null
            00000011    049E            CC_TBL_SIZE = <.-CC_TBL>/4
                        049E    ;
                        049E    ; The following table contains the device class and type of all devices supported
                        049E    ; by this test.  Any new devices must be added to this table or the device will
                        049E    ; be tested as a DT$_TTYUNKN. The only other information that has to be supplied
                        049E    ; to this program to test a new device is any device specific data. (See below)
                        049E    ; The table contains a byte for the device class, a byte for the device type, and
                        049E    ; an address of a device descriptor for the device specific data or zero if none
                        049E    ; exists.
                        049E    ;
                        049E
                        049E    TERM_NAMES:
            00000523    049E            .BLKB   TERM_TYPE_CNT*MAX_NAME_SIZE     ; number of terminal macros * max by
                        0523
            00000523    0523            PC1... = .
            0000049E    0523            PC2... = TERM_NAMES
                        0523    TESDEV_TBL:
            00000640    0523            .BLKB   15*TERM_TYPE_CNT                ; Allocate space for each macro
            00000640    0640            SAVE_PC... = .
                        0640            TERMINAL VT100,DC$_TERM,DT$_VT100,VT100_PREAMBLE,VT100_DATA,0
                        04A4            TERMINAL VT101,DC$_TERM,DT$_VT101,VT100_PREAMBLE,VT100_DATA,0
                        04AA            TERMINAL VT102,DC$_TERM,DT$_VT102,VT100_PREAMBLE,VT100_DATA,0
                        04B0            TERMINAL VT125,DC$_TERM,DT$_VT125,VT100_PREAMBLE,VT100_DATA,0
                        04B6            TERMINAL VT131,DC$_TERM,DT$_VT131,VT100_PREAMBLE,VT100_DATA,0
                        04BC            TERMINAL VT132,DC$_TERM,DT$_VT132,VT100_PREAMBLE,VT100_DATA,0
                        04C2            TERMINAL VT52,DC$_TERM,DT$_VT52,VT52_PREAMBLE,VT52_DATA,4
                        04C7            TERMINAL LA120,DC$_TERM,DT$_LA120,FORM_FEED,LA120_DATA,2
                        04CD            TERMINAL LA34,DC$_TERM,DT$_LA34,FORM_FEED,LA36_DATA,2
                        04D2            TERMINAL LA36,DC$_TERM,DT$_LA36,FORM_FEED,LA36_DATA,2
                        04D7            TERMINAL LA38,DC$_TERM,DT$_LA38,FORM_FEED,LA36_DATA,2
                        04DC            TERMINAL VT55,DC$_TERM,DT$_VT55,VT52_PREAMBLE,VT52_DATA,2
```

```
                     04E1               TERMINAL VT5X,DC$_TERM,DT$_VT5X,VT52_PREAMBLE,VT52_DATA,2
                     04E6               TERMINAL VT05,DC$_TERM,DT$_VT05,VT05_PREAMBLE,VT05_DATA,2
            000005F5 04EB               .=PC1...
                     05F5       UNKNOWN1:
                     05F5               TERMINAL UNKNOWN,DC$_TERM,DT$_TTYUNKN,FORM_FEED,UNKN_DATA,2
                     04F3               TERMINAL LA180,DC$_LP,DT$_LA180,FORM_FEED,LA180_DATA,2
                     04F9               TERMINAL LA11,DC$_LP,DT$_LA11,FORM_FEED,LA11_DATA,2
                     04FE               TERMINAL LP11,DC$_LP,DT$_LP11,FORM_FEED,LP11_DATA,2
                     0503               TERMINAL LAX,DC$_LP,DT$_LAX,FORM_FEED,LAX_DATA,2
            00000640 0507               . = SAVE_PC...
            00000640 0640               TESDEV_TBL_END = .
                     0640
                     0640               .ALIGN  LONG
                     0640       ;
                     0640       ; The following is device specific data. To add data of this type do the following:
                     0640       ;
                     0640       ;       1.      Create an ASCIC string for the terminal data to be printed before
                     0640       ;               the header (PREAMBLE).
                     0640       ;
                     0640       ;       2.      Create a string descriptor for the data.
                     0640       ;
                     0640       ;       3.      Enter the terminal data and update the TSD_SIZE constant if the
                     0640       ;               new terminal specific data size is larger than the current value
                     0640       ;               for TSD_SIZE.
                     0640       ;
                     0640       ;       4.      Enter the terminal information in the above table
                     0640       ;
                     0640       ; Terminal specific data for the VT100 terminal
                     0640       ;
                     0640
                     0640       VT100_DATA:
           000003E1' 0640               .LONG   VT100L
           00000657' 0644               .ADDRESS VT100_OUT_DATA
                     0648
                     0648       VT100_PREAMBLE:
                     0648       ;
                     0648       ; Set-up and header line
                     0648       ;
                 0E' 0648               .BYTE   VT100_PRE_LEN
              37 1B  0649               .ASCII  <ESC>/7/            ; Save attributes
              3C 1B  064B               .ASCII  <ESC>/</            ; Enter ANSI mode
        4A 32 5B 1B  064D               .ASCII  <ESC>/[2J/          ; Clear the screen
  48 31 3B 31 5B 1B  0651               .ASCII  <ESC>/[1;1H/        ; Go home
            0000000E 0657               VT100_PRE_LEN = .-VT100_PREAMBLE-1
                     0657       VT100_OUT_DATA:
  71 31 3B 30 5B 1B  0657               .ASCII  <ESC>/[0;1q/        ; LED 1
  6D 31 3B 30 5B 1B  065D               .ASCII  <ESC>/[0;1m/        ; Set bold
                     0663       ;
                     0663       ; Set-up double height and double width line modes
                     0663       ;
                     0663               .REPT   3
                     0663               .ASCII  <ESC>/#3/           ; Double height top
                     0663               .ASCII  <ESC>/E/            ; Next line
                     0663               .ASCII  <ESC>/#4/           ; Double height bottom
                     0663               .ASCII  <ESC>/E/            ; Next line
           33 23 1B  0663               .ENDR
                     0681               .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          .ASCII  <ESC>/[2D)/
                 28 44 32 5B 1B  09ED          .ASCII  <ESC>/[2D(/
                 2A 44 32 5B 1B  09F2          .ASCII  <ESC>/[2D*/
                 26 44 32 5B 1B  09F7          .ASCII  <ESC>/[2D&/
3B 30 5B 1B 41 31 5B 1B 44 31 5B 1B  09FC      .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                .ASCII  <ESC>/[0q/       ; No LEDs
     72 34 32 3B 32 31 5B 1B  0A20             .ASCII  <ESC>/[12;24r/   ; Set scroll to 12 thru 24
        48 31 3B 32 31 5B 1B  0A28             .ASCII  <ESC>/[12;1H/    ; Go to scroll area
              6D 30 5B 1B  0A2F                .ASCII  <ESC>/[0m/       ; Reset attributes
                 42 28 1B  0A33                .ASCII  <ESC>/(B/        ; Reset to USASCII
                    42 1B  0A36                .ASCII  <ESC>/B/         ; Restore cursor and attributes
              000003E1  0A38                   VT100L = .-VT100_OUT_DATA
                        0A38
                        0A38        VT100_ORIG_SCROL:                   ; Return to full screen scroll
3B 33 32 5B 1B 72 34 32 3B 30 5B 1B  0A38      .ASCII  <ESC>/[0;24r/<ESC>/[23;1H/
                    48 31  0A44
              0000000E  0A46                   VOSL = .-VT100_ORIG_SCROL
                        0A46
                        0A46        ;
                        0A46        ; Terminal specific data for the VT52 terminal
                        0A46        ;
                        0A46        VT52_DATA:
             00000168'  0A46                   .LONG   VT52L
             00000A53'  0A4A                   .ADDRESS VT52_OUT_DATA
                        0A4E        VT52_PREAMBLE:
                   04'  0A4E                   .BYTE   VT52_PRE_LEN
                 48 1B  0A4F                   .ASCII  <ESC>/H/                  ; Home
                 4A 1B  0A51                   .ASCII  <ESC>/J/                  ; Clear
              00000004  0A53                   VT52_PRE_LEN = .-VT52_PREAMBLE-1
                        0A53        VT52_OUT_DATA:
                 46 1B  0A53                   .ASCII  <ESC>/F/                  ; Enter graphics mode
74 6B 6A 69 68 67 66 65 64 63 62 61  0A55      .ASCII  /abcdefghijktuvwxyz{|}~/
   7E 7D 7C 7B 7A 79 78 77 76 75  0A61
   7D 44 1B 44 1B 7E 35 22 59 1B  0A6B         .ASCII  <ESC>/Y/<34><53>/~/<ESC>/D/<ESC>/D}/
44 1B 7B 44 1B 44 1B 7C 44 1B 44 1B  0A75      .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      .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      .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      .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      .ASCII  <ESC>/D/<ESC>/Dh/<ESC>/D/<ESC>/Dg/<ESC>/D/<ESC>/Df/
              66 44 1B  0ABD
44 1B 64 44 1B 44 1B 65 44 1B 44 1B  0AC0      .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      .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      .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      .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      .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      .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          .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          .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          .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          .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          .ASCII  <ESC>/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          .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          .ASCII  <ESC>/G/<CR><LF>         ; Exit graphics mode
                              00000168  0BBB          VT52L = .-VT52_OUT_DATA
                                        0BBB    ;
                                        0BBB    ; Terminal specific data for the VT05 terminal
                                        0BBB    ;
                                        0BBB    VT05_PREAMBLE:
                                   05'  0BBB          .BYTE   VT05_PRE_LEN
                        1F 00 00 00 1D  0BBC          .BYTE   29,0,0,0,31
                              00000005  0BC1          VT05_PRE_LEN = .-VT05_PREAMBLE-1
                                        0BC1
                                        0BC1    VT05_DATA:
                              00000000  0BC1          .LONG   0                ; There is no VT05 terminal specific data
                             00000BC1'  0BC5          .ADDRESS VT05_DATA
                                        0BC9    ;
                                        0BC9    ; Terminal specific data for the LA120 terminal
                                        0BC9    ;
                                        0BC9    LA120_DATA:
                              00000000  0BC9          .LONG   0                ; There is no LA120 terminal specific data
                             00000BC9'  0BCD          .ADDRESS LA120_DATA
                                        0BD1    ;
                                        0BD1    ; Terminal specific data for the LA36 terminal
                                        0BD1    ;
                                        0BD1    LA36_DATA:
                              00000000  0BD1          .LONG   0                ; There is no LA36 terminal specific data
                             00000BD1'  0BD5          .ADDRESS LA36_DATA
                                        0BD9    ;
                                        0BD9    ; Terminal specific data for the LA180 terminal
                                        0BD9    ;
                                        0BD9    LA180_DATA:
                              00000000  0BD9          .LONG   0                ; There is no LA180 terminal specific data
                             00000BD9'  0BDD          .ADDRESS LA180_DATA
                                        0BE1    ;
                                        0BE1    ; Terminal specific data for the LA11 terminal
                                        0BE1    ;
                                        0BE1    LA11_DATA:
                              00000000  0BE1          .LONG   0                ; There is no LA11 terminal specific data
                             00000BE1'  0BE5          .ADDRESS LA11_DATA
                                        0BE9    ;
                                        0BE9    ; Terminal specific data for the an UNKNOWN terminal
```

```
          0BE9  ;
          0BE9  UNKN_PREAMBLE:
      00  0BE9          .BYTE   0
          0BEA
          0BEA  UNKN_DATA:
00000000  0BEA          .LONG   0                       ; There is no UNKNOWN terminal
00000BC9' 0BEE          .ADDRESS LA120_DATA             ; specific data
          0BF2  ;
          0BF2  ; Terminal specific data for the LP11 terminal
          0BF2  ;
          0BF2  LP11_DATA:
00000000  0BF2          .LONG   0                       ; There is no LP11 terminal specific data
00000BF2' 0BF6          .ADDRESS LP11_DATA
          0BFA  ;
          0BFA  ; Terminal specific data for the LAX terminal
          0BFA  ;
          0BFA  LAX_DATA:
00000000  0BFA          .LONG   0                       ; There is no LAX terminal specific data
00000BFA' 0BFE          .ADDRESS LAX_DATA
          0C02  FORM_FEED:
      01  0C02          .BYTE   1                       ; Form feed ASCIC string
      0C  0C03          .BYTE   FF
          0C04
          0C04  CRLF:                                   ; Carriage return and Line feed
   0A 0D  0C04          .BYTE   CR,LF
          0C06  LFCR:                                   ; Line feed and Carriage return
   0D 0A  0C06          .BYTE   LF,CR
```

```
                           0C08          .SBTTL  Read/Write Data
                       00000000          .PSECT  RWDATA,WRT,NOEXE,PAGE
                           0000
                           0000  TTCHAN:                                  ; Channel associated with ctrl. term.
                     0000  0000          .WORD   0
                           0002
                           0002  FLAG:                                    ; Miscellaneous flag bits
                     0000  0002          .WORD   0                        ; (See Equated Symbols for definitions)
                           0004
                           0004  FAO_BUF:                                 ; FAO output string descriptor
                0000 0084  0004          .WORD   TEXT_BUFFER,0
                00000014'  0008          .ADDRESS BUFFER
                           000C
                           000C  BUFFER_PTR:                              ; Fake .ASCID buffer for misc. strings
                0000 0084  000C          .WORD   TEXT_BUFFER,0            ; A word for length, a word for desc.
                00000014'  0010          .ADDRESS BUFFER
                           0014
                           0014  BUFFER:                                  ; FAO output and other misc. buffer
                 00000098  0014          .BLKB   TEXT_BUFFER
                           0098
                           0098  DEVDSC:                                  ; Device name descriptor
                0000 000A  0098          .WORD   MAX_DEV_DESIG,0
                000000B7'  009C          .ADDRESS DEV_NAME
                           00A0
                           00A0  PROCESS_NAME:                            ; Process name
53 59 54 54 000000A8'010E0000'  00A0     .ASCID  /TTYS/
                 0000000B  00AC          PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
                 000000B7  00AC          .BLKB   PROCESS_NAME_FREE
                           00B7
                           00B7  DEV_NAME:                                ; Device name buffer
                 000000C6  00B7          .BLKB   MAX_DEV_DESIG+MAX_UNIT_DESIG
                 0000000F  00C6          NAME_LEN = .-DEV_NAME
                           00C6
                           00C6  DATA_DSC:                                ; Descriptor for STR$UPCASE
                 00000101  00C6          .LONG   WRITE_SIZE
                00000000'  00CA          .ADDRESS 0
                           00CE  DIB:                                     ; Device Information Block
                0000 0074  00CE          .WORD   DIB$K_LENGTH,0
                000000D6'  00D2          .ADDRESS DIBBUF
                           00D6  DIBBUF:
                 0000014A  00D6          .BLKB   DIB$K_LENGTH
                           014A
                           014A  DIB_SEC:                                 ; Device Information Block (secondary)
                0000 0074  014A          .WORD   DIB$K_LENGTH,0
                00000152'  014E          .ADDRESS DIBBUF_SEC
                           0152  DIBBUF_SEC:
                 000001C6  0152          .BLKB   DIB$K_LENGTH
                           01C6
                           01C6  ERROR_COUNT:                             ; Cumulative error count at runtime
                 00000000  01C6          .LONG   0
                           01CA
                           01CA  STATUS:                                  ; Status value on program exit
                 00000000  01CA          .LONG   0
                           01CE
                           01CE  QUAD_STATUS:                             ; IO status block for misc sys. svcs.
        00000000 00000000  01CE          .QUAD   0
                           01D6
```

```
                     01D6  INADDRESS:                          ; $CRMPSC address storage
 00000000 00000000   01D6           .LONG   0,0
                     01DE  OUTADDRESS:
 00000000 00000000   01DE           .LONG   0,0
                     01E6
                     01E6  UNIT_NUMBER:                        ; Current dev unit number
              0000   01E6           .WORD   0
                     01E8
                     01E8  DEVNAM_LEN:                         ; Current device name length
              0000   01E8           .WORD   0
                     01EA
                     01EA  RANDOM1:                            ; Random word #1
          AAAAAAAA   01EA           .LONG   ^XAAAAAAAA
                     01EE
                     01EE  RANDOM2:                            ; Random word #2
          A72EA72E   01EE           .LONG   ^XA72EA72E
                     01F2
                     01F2  ITERATION:                          ; # of times all tests were executed
          00000000   01F2           .LONG   0
                     01F6
                     01F6  MSG_BLOCK:                          ; Auxiliary $GETMSG info
          000001FA   01F6           .BLKB   4
                     01FA
                     01FA  PASS:                               ; Pass count
          00000000   01FA           .LONG   0
                     01FE
                     01FE  EXIT_DESC:                          ; Exit handler descriptor
          00000000   01FE           .LONG   0
         00000E15'   0202           .ADDRESS EXIT_HANDLER
          00000001   0206           .LONG   1
         000001CA'   020A           .ADDRESS STATUS
                     020E
                     020E  ARG_COUNT:                          ; Argument counter used by ERROR_EXIT
          00000000   020E           .LONG   0
                     0212
                     0212  RMSRUNDWN_BUF:                  ; Return buf for RMSRUNDWN close failures
         0000 0016   0212           .WORD   22,0
         0000021A'   0216           .ADDRESS RUNDWN_BUF
                     021A
                     021A  RUNDWN_BUF:
          00000230   021A           .BLKB 22
                     0230
                     0230  ;
                     0230  ; Head of self-relative UETP unit block queue.
                     0230  ;
                     0230           .ALIGN QUAD
                     0230
                     0230  UNIT_LIST:                          ; Head of unit block circular list
 00000000 00000000   0230           .QUAD   0
                     0238
                     0238  NEW_NODE:                           ; Newly aquired node address
 00000000 00000000   0238           .QUAD   0
                     0240
                     0240  HEAD_BUF:                           ; Buffer descriptor for the header message
          00000084   0240           .LONG   132
         00000000'   0244           .ADDRESS 0
                     0248  HEAD_LENGTH:                        ; Length of header record
```

```
00000000  0248   795            .LONG   0
          024C   796 SPARE:                                  ; Spot used to store byte while rotating dat
      00  024C   797            .BYTE   0
          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
```

UE
V0

```
                0255          .SBTTL  RMS-32 Data Structures
                0255          .ALIGN  LONG
                0258
                0258  SYSIN_FAB:                              ; Allocate FAB for SYS$INPUT
                0258          $FAB-
                0258          FNM = <SYS$INPUT>
                02A8
                02A8  SYSIN_RAB:                              ; Allocate RAB for SYS$INPUT
                02A8          $RAB-
                02A8          FAB = SYSIN_FAB,-
                02A8          ROP = PMT,-
                02A8          PBF = PROMPT,-
                02A8          PSZ = PMTSIZ,-
                02A8          UBF = DEV_NAME,-
                02A8          USZ = NAME_LEN
                02EC
                02EC  INI_FAB:                                ; Allocate FAB for UETINIDEV
                02EC          $FAB-
                02EC          FAC = <GET,PUT,UPD>,-
                02EC          RAT = CR,-
                02EC          SHR = <GET,PUT,UPI>,-
                02EC          FNM = <UETINIDEV.DAT>
                033C
                033C  INI_RAB:                                ; Allocate RAB for UETINIDEV
                033C          $RAB-
                033C          FAB = INI_FAB,-
                033C          RBF = BUFFER,-
                033C          UBF = BUFFER,-
                033C          USZ = REC_SIZE
                0380
                0380  DDB_RFA:                                ; RFA storage for INI_RAB
00000386        0380          .BLKB   6
                0386
                0386          .ALIGN  LONG
                0388  SUP_FAB:                                ; Allocate FAB for UETSUPDEV
                0388          $FAB-
                0388          FAC = GET,-
                0388          SHR = <UPI,GET>,-
                0388          RAT = CR,-
                0388          FOP = UFO,-
                0388          FNM = <UETSUPDEV.DAT>
                03D8
                03D8  ;
                03D8  ; Dummy FAB and RAB to copy to the UETP unit blocks
                03D8  ; The following FAB and RAB must be contiguous and in this order!
                03D8  ;
                03D8
                03D8  DUMMY_FAB:
                03D8          $FAB    RAT = <PRN>,-
                03D8                  RFM = <VFC>,-
                03D8                  FAC = <BRO,PUT>
                0428
                0428  DUMMY_RAB:
                0428          $RAB    RSZ = WRITE_SIZE,-
                0428                  ROP = <ASY,BIO>,-
                0428                  RHB = 0
```

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

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

```
                                01B8      ;
                                01B8      ; From UETINIDEV.DAT and UETSUPDEV.DAT, get information which gives controller
                                01B8      ; and unit configuration and lets us know if the setup to run this test was
                                01B8      ; done correctly.
                                01B8      ;
                                01B8              $OPEN   FAB = INI_FAB,-             ; Open file "UETINIDEV.DAT"
                                01B8                      ERR = RMS_ERROR
                                01C7              $CONNECT RAB = INI_RAB,-            ; Connect the RAB and FAB
                                01C7                      ERR = RMS_ERROR
                                01D6              $MGBLSC_S  INADR = INADDRESS,-      ; Connect to UETSUPDEV global section
                                01D6                         RETADR = OUTADDRESS,-
                                01D6                         GSDNAM = SUPDEV_GBLSEC,-
                                01D6                          FLAGS = #SEC$M_EXPREG
      00000978 8F      50   D1  01F5              CMPL    R0,#SS$_NOSUCHSEC           ; Was the section already there?
                       37   12  01FC              BNEQ    30$                         ; BR if it was...
                                01FE              $OPEN   FAB = SUP_FAB,-             ; ...else open "UETSUPDEV.DAT"
                                01FE                      ERR = RMS_ERROR
                                020D              $CRMPSC_S CHAN = SUP_FAB+FAB$L_STV,- ; Create the global section
                                020D                       INADR = INADDRESS,-
                                020D                      RETADR = OUTADDRESS,-
                                020D                      GSDNAM = SUPDEV_GBLSEC,-
                                020D                       FLAGS = #SEC$M_EXPREG!SEC$M_GBL
                                0235      30$:
   56    01E2'CF   01DE'CF  C3  0235              SUBL3   OUTADDRESS,OUTADDRESS+4,R6 ; Compute global section length
                                023D
                                023D      FIND_IT:
                                023D              $GET    RAB = INI_RAB,-             ; Get the first record
                                023D                      ERR = RMS_ERROR
               02BF'CF      DF  024C              PUSHAL  CONT_DESC                   ; Make sure...
               02BF'CF      DF  0250              PUSHAL  CONT_DESC                   ; ...that the controller name...
      00000000'GF      02   FB  0254              CALLS   #2,G^STR$UPCASE             ; ...is all uppercase letters
         0014'CF    44 8F   91  025B              CMPB    #^A/D/,BUFFER               ; Is this a DDB?
                       27   13  0261              BEQL    10$                         ; Go on if not
         0014'CF    45 8F   91  0263              CMPB    #^A/E/,BUFFER               ; Is this the end of the file?
                       D2   12  0269              BNEQ    FIND_IT                     ; Continue on if not
               0098'CF      DF  026B              PUSHAL  DEVDSC                      ; Push device not supported message
               00A0'CF      DF  026F              PUSHAL  PROCESS_NAME                ; Parameters on the stack
                       02   DD  0273              PUSHL   #2
           00748333 8F      DD  0275              PUSHL   #UETP$_DENOSU
                       02   F0  027B              INSV    #STS$K_ERROR,-              ; Set the severity code...
                       00       027D                      #STS$V_SEVERITY,-
               6E      03       027E                      #STS$S_SEVERITY,(SP)
         01CA'CF       6E   D0  0280              MOVL    (SP),STATUS                 ; ...and save it as the exit status
                       04   DD  0285              PUSHL   #4
                     0B0B   31  0287              BRW     ERROR_EXIT                  ; Exit in error
                                028A      10$:
 00B7'CF   001A'CF   01E8'CF 29 028A              CMPC    DEVNAM_LEN,BUFFER+6,DEV_NAME ; Is this the right controller?
                       A7   12  0294              BNEQ    FIND_IT                     ; BR if not
   0380'CF   034C'CF   06   28  0296              MOVC3   #6,INI_RAB+RAB$W_RFA,DDB_RFA ; Save the Record File Address
         0018'CF    54 8F   91  029E              CMPB    #^A/T/,BUFFER+4             ; Can we test this controller?
                       2F   13  02A4              BEQL    FOUND_IT                    ; BR if we can...
                                02A6              $FAO_S  CTRSTR = DEAD_CTRLNAME,-    ; ...and yell at user if we can't
                                02A6                      OUTLEN = BUFFER_PTR,-
                                02A6                      OUTBUF = FAO_BUF,-
                                02A6                      P1     = #DEVDSC
         01CA'CF       14   D0  02BF              MOVL    #SS$_BADPARAM,STATUS        ; Set return status
               000C'CF      DF  02C4              PUSHAL  BUFFER_PTR                  ; ...
```

```
                    01  DD 02C8              PUSHL   #1                          ; ...
           00741132 8F  DD 02CA              PUSHL   #UETP$_TEXT!STS$K_ERROR     ; ...
                    03  DD 02D0              PUSHL   #3                          ; ...
                  0AC0  31 02D2              BRW     ERROR_EXIT                  ; We can't test what we can't test
                           02D5
                           02D5 FOUND_IT:
                           02D5              $GET    RAB = INI_RAB,-             ; Get a record
                           02D5                      ERR = RMS_ERROR
               02BF'CF  DF 02E4              PUSHAL  CONT_DESC                   ; Make sure...
               02BF'CF  DF 02E8              PUSHAL  CONT_DESC                   ; ...that this line...
       00000000'GF   02  FB 02EC             CALLS   #2,G^STR$UPCASE             ; ...is all uppercase letters
    0014'CF      55 8F  91 02F3              CMPB    #^A/U/,BUFFER               ; Is this a UCB?
                    24  13 02F9              BEQL    30$                         ; BR if it is
    0014'CF      44 8F  91 02FB              CMPB    #^A/D/,BUFFER               ; Is this a DDB?
                    19  13 0301              BEQL    20$                         ; BR if yes
    0014'CF      45 8F  91 0303              CMPB    #^A/E/,BUFFER               ; Is this the end?
                    11  13 0309              BEQL    20$                         ; BR if yes
                           030B 10$:
               0161'CF  DF 030B              PUSHAL  ILLEGAL_REC                 ; Then this is an error in the record
                    01  DD 030F              PUSHL   #1                          ; Push the error message
           00741132 8F  DD 0311              PUSHL   #UETP$_TEXT!STS$K_ERROR     ; Push the signal name
                    03  DD 0317              PUSHL   #3                          ; Push the temp arg count
                  0A79  31 0319              BRW     ERROR_EXIT                  ; Finish for good
                           031C 20$:
                  00FE  31 031C              BRW     ALL_SET                     ; Found DDB or END
                           031F 30$:
    0018'CF      54 8F  91 031F              CMPB    #^A/T/,BUFFER+4             ; Is the unit testable?
                    AE  12 0325              BNEQ    FOUND_IT                    ; BR if not
                    01  DD 0327              PUSHL   #1                          ; Flag to ignore blanks when converting
                    02  DD 0329              PUSHL   #2                          ; Set byte size of results
               01E6'CF  DF 032B              PUSHAL  UNIT_NUMBER                 ; Set address to receive word
               02B7'CF  DF 032F              PUSHAL  UNIT_DESC                   ; Push string address
       00000000'GF   04  FB 0333             CALLS   #4,G^OTS$CVT_TI_L           ; Convert ASCII unit # to decimal
                 CE 50  E9 033A              BLBC    R0,10$                      ; Don't allow bogus unit to pass
                 05 20  3B 033D              SKPC    #^A/ /,#MAX_UNIT_DESIG,-    ; Find out where unit number really is
               001A'CF     0340                      BUFFER+6
                    50  D7 0343              DECL    R0                          ; Units must all be at least one digit
            61   50 30  3B 0345              SKPC    #^A/0/,R0,(R1)              ; Skip leading zeroes on the unit
                    50  D6 0349              INCL    R0                          ; Compensate for DECL above
0098'CF    01E8'CF  50  A1 034B              ADDW3   R0,DEVNAM_LEN,DEVDSC        ; Calculate device'unit string length
              52 01E8'CF  3C 0353            MOVZWL  DEVNAM_LEN,R2               ; Offset to unit number in DEVDSC
    00B7'C2    61   50  28 0358              MOVC3   R0,(R1),DEV_NAME(R2)        ; Append unit number to device
                           035E              $GETDEV_S DEVNAM = DEVDSC,-         ; Get the device characteristics
                           035E                      PRIBUF = DIB,-
                           035E                      SCDBUF = DIB_SEC
                    06  E1 0375              BBC     #DEV$V_SPL,-
            0E 0152'CF     0377                      DIB$L_DEVCHAR+DIBBUF_SEC,35$    ; Check for the spool bit
                    08  BB 037B              PUSHR   #^M<R3>                     ; Save pointer to end of device name
    0152'CF    0074 8F  28 037D              MOVC3   #DIB$K_LENGTH,DIBBUF_SEC,- ; Use secondary buffer if spooled
               00D6'CF     0384                      DIBBUF
                    08  BA 0387              POPR    #^M<R3>                     ; Restore pointer to end of device
               0098'CF  D6 0389 35$:         INCL    DEVDSC                      ; Don't forget the : on the end
                 63 3A  90 038D              MOVB    #^A/:/,(R3)                 ; Stick in the colon
          57  00DA'CF   9A 0390              MOVZBL  DIBBUF+DIB$B_DEVCLASS,R7    ; Save the device class
          58  00DB'CF   9A 0395              MOVZBL  DIBBUF+DIB$B_DEVTYPE,R8     ; Save the device type
                           039A              $FAO_S  CTRSTR = CS1,-
                           039A                      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$:
                   0098'CF   DF     03E5  1096           PUSHAL  DEVDSC                   ; Push device not supported message
                   00A0'CF   DF     03E9  1097           PUSHAL  PROCESS_NAME             ; Parameters on the stack
                        02   DD     03ED  1098           PUSHL   #2                       ; Push the argument count
               00748333 8F   DD     03EF  1099           PUSHL   #UETP$_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  ;
                          041D  ; Arrive here when we have the device configuration.  In normal or loop forever
                          041D  ; mode, set a timer far enough in the future such that we can do a reasonable
                          041D  ; set of tests before the timer expires, but if our device gets hung, the
                          041D  ; program won't waste too much time before noticing.  Let one-shot mode be a
                          041D  ; special case.
                          041D  ;
                          041D  ALL_SET:
       0230'CF    D5      041D          TSTL    UNIT_LIST                   ; Anything to test?
            16    12      0421          BNEQ    10$                         ; BR if yes
       013B'CF    DF      0423          PUSHAL  NOUNIT_SELECTED             ; Else set up the error message...
            01    DD      0427          PUSHL   #1                          ; ...argument count...
   00741132 8F    DD      0429          PUSHL   #UETP$_TEXT!STS$K_ERROR     ; ...signal name...
            03    DD      042F          PUSHL   #3                          ; ...and parameter count
    01CA'CF 14    D0      0431          MOVL    #SS$_BADPARAM,STATUS        ; Set return status
          095C    31      0436          BRW     ERROR_EXIT                  ; ...and give up, complaining
                          0439  10$:
   0002'CF   04       A8  0439          BISW2   #SAFE_TO_UPDM,FLAG          ; OK safe to update UETINIDEV.DAT now
                          043E  20$:
                          043E          $TRNLOG_S LOGNAM = MODE,-           ; Get the run mode
                          043E                    RSLLEN = BUFFER_PTR,-
                          043E                    RSLBUF = FAO_BUF
   0014'CF   20       8A  0457          BICB2   #LC_BITM,BUFFER             ; Convert to upper case
 0014'CF   4F 8F      91  045C          CMPB    #^A7O/,BUFFER               ; Is this a one shot?
            08        12  0462          BNEQ    TIME_IT                     ; BR if not
   0002'CF   02       A8  0464          BISW2   #TEST_OVERM,FLAG            ; End after one iteration
          012D        31  0469          BRW     ONE_SHOT                    ; Skip the SETIMR
                          046C  TIME_IT:
                          046C          $SETIMR_S DAYTIM = THREEMIN,-       ; Set timer AST to 3 minutes
                          046C                    ASTADR = TIME_OUT,-
                          046C                    EFN    = #EFN2
```

```
                                 047F          .SBTTL  Test the Terminals
                                 047F
    56  0230'CF  00000230'8F  C1 047F          ADDL3   #UNIT_LIST,UNIT_LIST,R6 ; Set the unit block address
                                 0489     RESTART:
                                 0489     ;*****************************************************************************
                                 0489     ;
                                 0489     ; Device test specific code goes here.
                                 0489     ;
                                 0489     ; At this point the device designation is in location DEV_NAME pointed to by
                                 0489     ; descriptor DEVDSC. The device is known to be supported and testable by this test.
                                 0489     ; To leave successfully BRW SUC_EXIT, to leave in error BRW ERROR_EXIT.
                                 0489     ;
                                 0489     ;*****************************************************************************
                                 0489
                   01FA'CF    D5 0489          TSTL    PASS                      ; Only create and connect on the first pass
                        73    12 048D          BNEQ    10$                       ; else branch
                        07    E1 048F          BBC     #SPL_UNITV,-              ; Is this unit spooled?
                  03 0B A6       0491                  UETUNT$B_FLAGS(R6),3$
                      00AE    31 0494          BRW     25$                       ; BR if so - try the next one
                                 0497     3$:
                                 0497          $CREATE  FAB = UETUNT$K_FAB(R6)   ; Create the channel
                     4E 50    E8 04A2          BLBS    R0,5$                     ; Did we succeed? Br if yes...
                   01C6'CF    D6 04A5          INCL    ERROR_COUNT               ; Bump the error count
                        50    DD 04A9          PUSHL   R0                        ; Push the error code...
                        50    DD 04AB          PUSHL   R0                        ; ...and the error code...
                     14 A6    9A 04AD          MOVZBL  UETUNT$T_FILSPC(R6),-
                   0098'CF       04B0                  DEVDSC                    ; ...get the name size...
                     15 A6    DE 04B3          MOVAL   UETUNT$T_FILSPC+1(R6),-
                   009C'CF       04B6                  DEVDSC+4                  ; ...get the name address...
                   0098'CF    DF 04B9          PUSHAL  DEVDSC                    ; ...and push the device designation...
                   000F'CF    DF 04BD          PUSHAL  TEST_NAME                 ; ...and the test name...
               000F0003 8F    DD 04C1          PUSHL   #^XF0003                  ; ...and the arg count...
               0074819A 8F    DD 04C7          PUSHL   #UETP$_DEUNUS!STS$K_ERROR ; ...and the signal name...
                   01C6'CF    DD 04CD          PUSHL   ERROR_COUNT               ; ...and the error count...
                   00A0'CF    DF 04D1          PUSHAL  PROCESS_NAME              ; ...our own name...
               00010002 8F    DD 04D5          PUSHL   #^X10002                  ; ...and the argument count...
               00748022 8F    DD 04DB          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...and the signal name...
        00000000'GF     0A    FB 04E1          CALLS   #10,G^LIB$SIGNAL          ; ...and print the error
                        02    8A 04E8          BICB2   #UETUNT$M_TESTABLE,-
                     0B A6       04EA                  UETUNT$B_FLAGS(R6)        ; Clear the testable bit
                        01    88 04EC          BISB2   #UETUNT$M_DONE,-
                     0B A6       04EE                  UETUNT$B_FLAGS(R6)        ; Set the done testing bit
                      0052    31 04F0          BRW     25$                       ; Skip testing
                                 04F3     5$:
                                 04F3          $CONNECT RAB = UETUNT$K_RAB(R6),-
                                 04F3                   ERR = RMS_ERROR          ; Connect the RAB
                                 0502     10$:
                        02    88 0502          BISB2   #UETUNT$M_TESTABLE,-
                     0B A6       0504                  UETUNT$B_FLAGS(R6)        ; Set the testable bit
           58  09EB'CF    DE     0506          MOVAL   SERVICE_IO2,R8            ; Set the success service address
           57  0160 C6    DE     050B          MOVAL   UETUNT$K_RAB(R6),R7       ; Get the RAB address
               02BA C6    DE     0510          MOVAL   UETUNT$K_HEADER+2(R6),-
                  28 A7          0514                  RAB$L_RBF(R7)             ; Set RAB write address
               02B8 C6    B0     0516          MOVW    UETUNT$K_HEADER(R6),-
                  22 A7          051A                  RAB$W_RSZ(R7)             ; Set RAB write size
     0100 8F      22 A7   B1     051C          CMPW    RAB$W_RSZ(R7),#HDR_OUT_SIZE ; Check the size
                     0B   15     0522          BLEQ    20$                       ; Br if small enough...
```

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

```
02BA C6   57   0C04'CF   02   39  05FA          MATCHC  #2,CRLF,R7,-
                                  0603                  UETUNT$K_HEADER+2(R6)  ; Find the CR/LF. There has to be one
                02BA C6       DE  0603          MOVAL   UETUNT$K_HEADER+2(R6),-
                      53          0607                  R8                     ; Get the header start address
              58   53   58    C3  0608          SUBL3   R8,R3,R8               ; Get the byte count to the start of...
              024D'CF   56    D0  060C          MOVL    R6,CUR_UNTBLK          ; Save the unit block address
                                  0611          $QIOW_S FUNC = #IO$_WRITEVBLK,-
                                  0611                  CHAN = UETUNT$W_CHAN(R6),-
                                  0611                  IOSB = UETUNT$K_RAB+RAB$L_STS(R6),-
                                  0611                  P1   = UETUNT$K_HEADER+2(R6),-
                                  0611                  P2   = R8              ; Output the header
                   0B 50      E8  0633          BLBS    R0,20$                 ; BR if no error...
              01CA'CF   50    D0  0636          MOVL    R0,STATUS              ; Save the status
                   59   50    D0  063B          MOVL    R0,R9
                      0037    31  063E          BRW     ONESHOT_ERROR          ; ...else take care of failure.
                                  0641  20$:
                                  0641          $CANTIM_S                      ; Forget the watch dog timer
                                  064A          $SETSFM_S ENBFLG = #1          ; Turn on SS failure mode
           59   0168 C6       3C  0653          MOVZWL  UETUNT$K_RAB+RAB$L_STS(R6),R9 ; Get the IOSB value
                   0A 59      E8  0658          BLBS    R9,30$                 ; If the IOSB has no error we are OK
              01CA'CF   59    D0  065B          MOVL    R9,STATUS              ; Save the error code
                      0015    31  0660          BRW     ONESHOT_ERROR          ; Report the failure
                        04    11  0663          BRB     NEXT_TTY               ; ...and look at the rest of the lines
                                  0665  30$:
                        02    88  0665          BISB2   #UETUNT$M_TESTABLE,-
                     0B A6        0667                  UETUNT$B_FLAGS(R6)     ; We have a good line here. Say so.
                                  0669  NEXT_TTY:
                   56   66    C0  0669          ADDL2   (R6),R6                ; Get the next unit block
         56   00000230'8F     D1  066C          CMPL    #UNIT_LIST,R6          ; Are we all done?
                        7C    13  0673          BEQL    SUC_EXIT               ; Br if yes
                      FF2B    31  0675          BRW     ONE_SHOT_LOOP          ; Else test the next line
                                  0678  ONESHOT_ERROR:
                                  0678          $CANTIM_S                      ; Forget the watchdog timer
                   01C6'CF    D6  0681          INCL    ERROR_COUNT            ; Bump the error count
                   01CA'CF    DD  0685          PUSHL   STATUS                 ; Push the error code...
                   01CA'CF    DD  0689          PUSHL   STATUS                 ; ...and the error code...
                   0098'CF    DF  068D          PUSHAL  DEVDSC                 ; ...and the device designation...
                   000F'CF    DF  0691          PUSHAL  TEST_NAME              ; ...and the test name...
               000F0003 8F    DD  0695          PUSHL   #^XF0003               ; ...and the arg count...
               0074819A 8F    DD  069B          PUSHL   #UETP$_DEUNUS!STS$K_ERROR ; ...and the signal name...
                   01C6'CF    DD  06A1          PUSHL   ERROR_COUNT            ; ...and the error count...
                   00A0'CF    DF  06A5          PUSHAL  PROCESS_NAME           ; ...our own name...
               00010002 8F    DD  06A9          PUSHL   #^X10002               ; ...and the argument count...
               00748022 8F    DD  06AF          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...and the signal name...
           00000000'GF   0A   FB  06B5          CALLS   #10,G^LIB$SIGNAL       ; ...and print the error
               0830 8F   59   B1  06BC          CMPW    R9,#SS$_CANCEL         ; Did a timeout occur?
                         05   13  06C1          BEQL    10$                    ; BR if it was
                    2C   59   B1  06C3          CMPW    R9,#SS$_ABORT          ; Did a timeout occur?
                         13   12  06C6          BNEQ    20$                    ; BR if not
                                  06C8  10$:
                   020C'CF    DF  06C8          PUSHAL  TIME_OUT_MSG           ; Print the timeout error message
                        01    DD  06CC          PUSHL   #1
               00741132 8F    DD  06CE          PUSHL   #UETP$_TEXT!STS$K_ERROR
           00000000'GF   03   FB  06D4          CALLS   #3,G^LIB$SIGNAL
                                  06DB  20$:
                      FF8B    31  06DB          BRW     NEXT_TTY               ; Go do the next unit
                                  06DE  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          .SBTTL  GET_NODE Routine
0771 ;++
0771 ; FUNCTIONAL DESCRIPTION:
0771 ; This routine dynamically allocates enough memory for a unit block,
0771 ; a device dependent parameter area and I/O buffers. The unit block is inserted
0771 ; into the queue header UNIT_LIST.  It then initializes the unit block.
0771 ; The unit block format is as follows:
0771 ;
0771 ;                         +----------------+ -------
0771 ;       UETUNT$L_FLINK    !                !       ^
0771 ;                         +----------------+       !
0771 ;       UETUNT$L_BLINK    !                !       !
0771 ;                         +------------+---+       !
0771 ;       UETUNT$B_TYPE                  !   !       !
0771 ;                             +--------+---+       !
0771 ;       UETUNT$W_SIZE         !    !   !    contains DEVDEP_SIZE + UETUNT$C_INDSIZ
0771 ;                         +---+--------+           !
0771 ;       UETUNT$B_FLAGS    !   !                    !
0771 ;                         +---+    +-------+       !
0771 ;       UETUNT$W_CHAN              !   !   !       !
0771 ;                         +--------+-------+       !
0771 ;       UETUNT$W_FUNC     !   !    !               !
0771 ;                         +--------+-------+       +----- UETUNT$C_SIZE
0771 ;       UETUNT$L_ITER     !                !       !
0771 ;                         +----------------+       !
0771 ;       UETUNT$T_FILSPC   !                !       !
0771 ;                         !/\/\/\/\/\/\/\/\!       !
0771 ;                         NAM$C_MAXRSS bytes       !
0771 ;                         !/\/\7\/\/\/\/\/\!       !
0771 ;                         !                !       !
0771 ;       UETUNT$K_FAB      !----------------!       !
0771 ;                         !                !       !
0771 ;                         !/\/\/\/\/\/\/\/\!       !
0771 ;                          FAB$C_BLN bytes         !
0771 ;                         !/\/\/\/\/\/\/\/\!       !
0771 ;                         !                !       !
0771 ;       UETUNT$K_RAB      +----------------+       !
0771 ;                         !                !       !
0771 ;                         !\/\/\/\/\/\/\/\/!       !
0771 ;                          RAB$C_BLN bytes         !
0771 ;                         !\/\/\7\/\/\/\/\/!       !
0771 ;                         !                !       v
0771 ;       UETUNT$K_DEVDEP   +------------+---+ -------
0771 ;       UETUNT$B_LINE                  !   !       ^
0771 ;                                  +---+---+       !
0771 ;       UETUNT$B_HD_LEN            !   !           !
0771 ;                             +----+---+           !
0771 ;       UETUNT$B_LENGTH       !    !               !
0771 ;                         +---+----+               +----- DEVDEP_SIZE
0771 ;       UETUNT$W_WIDTH    !        !               !
0771 ;                         +--------+-------+       !
0771 ;       UETUNT$L_CURHDR   !                !       !
0771 ;                         +----------------+       !
0771 ;       UETUNT$K_CCTBL    !                !       v
0771 ;       UETUNT$K_WRITE    +----------------+ -------
```

```
                         0771 ;        UETUNT$K_WRITE  +---------------+ -------
                         0771 ;        WRITE buffer    !               !        ^
                         0771 ;                        !\/\/\/\/\/\/\/\!        !
                         0771 ;                           257 bytes             +----- WRITE_SIZE
                         0771 ;                        !/\/\/\/\/\/\/\/!        !
                         0771 ;                        !               !        v
                         0771 ;        UETUNT$K_HEADER +---------------+ -------
                         0771 ;        HEADER buffer   !               !        ^
                         0771 ;                        !\/\/\/\/\/\/\/\!        !
                         0771 ;                           132 bytes             +----- HEADER_SIZE
                         0771 ;                        !/\/\/\/\/\/\/\/!        !
                         0771 ;                        !               !        v
                         0771 ;                        +---------------+ -------
                         0771 ;
                         0771 ; CALLING SEQUENCE:
                         0771 ;        BSBW    GET_NODE
                         0771 ;
                         0771 ; INPUT PARAMETERS:
                         0771 ;        NONE
                         0771 ;
                         0771 ; IMPLICIT INPUTS:
                         0771 ;        UNIT_LIST contains head of doubley linked circular list of unit blocks
                         0771 ;
                         0771 ; OUTPUT PARAMETERS:
                         0771 ;        R6 = address of  new node
                         0771 ;
                         0771 ; IMPLICIT OUTPUTS:
                         0771 ;        New inited unit block is inserted into UNIT_LIST
                         0771 ;
                         0771 ; COMPLETION CODES:
                         0771 ;        NONE
                         0771 ;
                         0771 ; SIDE EFFECTS:
                         0771 ;        NONE
                         0771 ;
                         0771 ;--
                         0771
                         0771 GET_NODE:
                         0771
                         0771          $EXPREG_S PAGCNT = #PAGES,-
                         0771                    RETADR = NEW_NODE     ; Get a new node of demand zero memory
0230'CF   0238'DF   5D   0782          INSQTI  @NEW_NODE,UNIT_LIST     ; Put the new node in the unit list
     56   0238'CF   D0   0789          MOVL    NEW_NODE,R6             ; Save a copy of its address
  08 A6        01   90   078E          MOVB    #1,UETUNT$B_TYPE(R6)    ; Set the structure type
          01B7 8F   B0   0792          MOVW    #UETUNT$C_INDSIZ+DEVDEP_SIZE,-
             09 A6       0796                  UETUNT$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
              05  07D3  1469      RSB
```

```
                               07D4          .SBTTL  SET_DEVDEP Routine
                               07D4     ;++
                               07D4     ; FUNCTIONAL DESCRIPTION:
                               07D4     ;       This routine initializes the device dependent parameters in the unit
                               07D4     ;       block specified.
                               07D4     ;
                               07D4     ; CALLING SEQUENCE:
                               07D4     ;       PUSHAL  DIB                     ; Push the device information block adr
                               07D4     ;       PUSHAL  UNIT_BLOCK              ; Push the unit block address
                               07D4     ;       CALLS   #2,SET_DEVDEP           ; Set the device characteristics
                               07D4     ;
                               07D4     ; INPUT PARAMETERS:
                               07D4     ;       4(AP) address of the unit block
                               07D4     ;       8(AP) address of the device information block
                               07D4     ;
                               07D4     ; IMPLICIT INPUTS:
                               07D4     ;       NONE
                               07D4     ;
                               07D4     ; OUTPUT PARAMETERS:
                               07D4     ;       NONE
                               07D4     ;
                               07D4     ; IMPLICIT OUTPUTS:
                               07D4     ;       Device characteristics are inited in the unit block
                               07D4     ;
                               07D4     ; COMPLETION CODES:
                               07D4     ;       NONE
                               07D4     ;
                               07D4     ; SIDE EFFECTS:
                               07D4     ;       NONE
                               07D4     ;
                               07D4     ;--
                               07D4
                               07D4     SET_DEVDEP:
                         0FFC  07D4          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                               07D6
             56    04 AC   D0  07D6          MOVL    4(AP),R6                ; Get the unit block address
             59    08 AC   D0  07DA          MOVL    8(AP),R9                ; Get the DIB address
           5A    0523'CF   DE  07DE          MOVAL   TESDEV_TBL,R10          ; Set the table address
           05 A9      21   91  07E3          CMPB    #DT$_LA120,DIB$B_DEVTYPE(R9) ; Should only 2 pages be printed?
                      04   12  07E7          BNEQ    10$                     ; Br if not...
           0B A6      04   88  07E9          BISB2   #UETUNT$M_2PL,UETUNT$B_FLAGS(R6) ; ...else set the flag
                               07ED     10$:
        04 A9      43 8F   91  07ED          CMPB    #DC$_LP,DIB$B_DEVCLASS(R9) ; Should only 2 pages be printed?
                      04   12  07F2          BNEQ    20$                     ; Br if not ...
           0B A6      04   88  07F4          BISB2   #UETUNT$M_2PL,UETUNT$B_FLAGS(R6) ; ...else set the flag
                               07F8     20$:
           04 A9      6A   B1  07F8          CMPW    (R10),DIB$B_DEVCLASS(R9) ; Is this the right table entry?
                      58   13  07FC          BEQL    30$                     ; Br if yes
    FFF2 5A   0F  0640'8F  3D  07FE          ACBW    #TESDEV_TBL_END,#15,R10,20$ ; Do it for the whole table
                               0806          $GETDVI_S DEVNAM = DEVDSC,-     ; Get the name of the device...
                               0806                  EFN    = #SS_SYNCH_EFN,- ; ...from which the test is run.
                               0806                  ITMLST = TERM_ITMLST,-  ; We have a special need here,...
                               0806                  IOSB   = QUAD_STATUS    ; ...get it now & save it in BUFFER
           12 01CE'CF      E9  0822          BLBC    QUAD_STATUS,25$         ; Abort if we can't get it
  0251'CF  01000000 8F     D3  0827          BITL    #TT2$M_ANSICRT,XTERM_CHAR ; Is this an ANSI terminal?
                      1F   13  0830          BEQL    27$                     ; BR if not...
           5A    0523'CF   DE  0832          MOVAL   TESDEV_TBL,R10          ; ...otherwise set up as a VT100...
```

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

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

```
                              0977          .SBTTL SERVICE_IO
                              0977  ;++
                              0977  ; FUNCTIONAL DESCRIPTION:
                              0977  ;       This routine services all asynchronous I/O completions for all unit
                              0977  ;       numbers under test.  After the proper number of iterations or elapsed
                              0977  ;       time have been completed the unit number is disabled for testing.
                              0977  ;       This routine does the data checking as well as the next I/O initiation.
                              0977  ;
                              0977  ; CALLING SEQUENCE:
                              0977  ;       Entered via an AST from the RMS
                              0977  ;
                              0977  ; INPUT PARAMETERS:
                              0977  ;       4(AP) = address of RAB of unit completing I/O
                              0977  ;
                              0977  ; OUTPUT PARAMETERS:
                              0977  ;       None
                              0977  ;--
                              0977
                              0977  SERVICE_IO:
                        0FFC  0977          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                              0979
              5B   04 AC  D0  0979          MOVL    4(AP),R11                       ; Get the RAB address
                              097D  NEW_PAGE:
                              097D          $DISCONNECT RAB = (R11),-
                              097D                  ERR = RMS_ERROR                 ; Disconnect to set IO mode
    04 AB   00000800 8F   C8  098A          BISL2   #RAB$M_BIO,RAB$L_ROP(R11)       ; Set to block IO for the header
                              0992          $CONNECT        RAB = (R11),-
                              0992                  ERR = RMS_ERROR                 ; OK now hook it back up
                     02'  11  099F          BRB     SERVICE_IO1+2                   ; This is first time only for the
                              09A1  ;
                              09A1  ; This entry point is to service header AST's.
                              09A1  ;
                              09A1
                              09A1  SERVICE_IO1:
                        0FFC  09A1          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
              5B   04 AC  D0  09A3          MOVL    4(AP),R11                       ; Get the RAB address
              5A   18 AB  D0  09A7          MOVL    RAB$L_CTX(R11),R10              ; Save the unit block address
              58   F3 AF  DE  09AB          MOVAL   SERVICE_IO1,R8                  ; Set the AST service routine addres
        28 AB   01AB CA   D0  09AF          MOVL    UETUNT$L_CURHDR(R10),RAB$L_RBF(R11) ; Set the transfer address
           57   02B8 CA   3C  09B5          MOVZWL  UETUNT$K_HEADER(R10),R7         ; Get the header size
           59   02BA CA   DE  09BA          MOVAL   UETUNT$K_HEADER+2(R10),R9       ; Get the header start address
              59      57  C0  09BF          ADDL2   R7,R9                           ; Calculate the end address
     00000100 8F      59  F1  09C2          ACBL    R9,#HDR_OUT_SIZE,-
         000D 01AB CA         09C9                  UETUNT$L_CURHDR(R10),10$        ; Output header in chunks
           59   28 AB     C2  09CE          SUBL2   RAB$L_RBF(R11),R9               ; If not enough header left for 256.
           22 AB      59  B0  09D2          MOVW    R9,RAB$W_RSZ(R11)               ; ...then set size to whats left...
           58  09EB'CF    DE  09D6          MOVAL   SERVICE_IO2,R8                  ; ...and setup AST adr for data
                              09DB  10$:
                              09DB          $WRITE  RAB = (R11),-
                              09DB                  SUC = (R8),-
                              09DB                  ERR = RMS_ERROR                 ; Start the next header write
                          04  09EA          RET
                              09EB  ;
                              09EB  ; This entry point is to set up for non-header data transfers.
                              09EB  ;
                              09EB
                              09EB  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_TBL+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  40$:
  62 0B A6      00   E1   0AD7          BBC     #UETUNT$V_DONE,UETUNT$B_FLAGS(R6),90$ ; BR if any unit is not done
     56  66     56   C1   0ADC          ADDL3   R6,(R6),R6                      ; Get the next unit block adr
00000230'8F     56   D1   0AE0          CMPL    R6,#UNIT_LIST                   ; All done checking?
                EE   12   0AE7          BNEQ    40$                             ; BR if yes
                          0AE9          $WAKE_S                                 ; Else its all over with
                     04   0AF4          RET                                     ; Tha-a-Tha-a-Thats all folks
                          0AF5  50$:
       10 AA         D6   0AF5          INCL    UETUNT$L_ITER(R10)              ; Bump iteration counter
     01F2'CF         D6   0AF8          INCL    ITERATION                       ; and test wide iteration count
  0C 0B AA      02   E1   0AFC          BBC     #UETUNT$V_2PL,UETUNT$B_FLAGS(R10),60$ ; Br if no 2 page limit
     10 AA      03   D1   0B01          CMPL    #3,UETUNT$L_ITER(R10)           ; See if 2 pages are out
                06   12   0B05          BNEQ    60$                             ; Br if not yet
     0B AA      01   88   0B07          BISB2   #UETUNT$M_DONE,UETUNT$B_FLAGS(R10) ; Set the done flag
                C0   11   0B0B          BRB     30$                             ; Stop testing
                          0B0D  60$:
     01A5 CA         90   0B0D          MOVB    UETUNT$B_HD_LEN(R10),-
     01A4 CA              0B11                  UETUNT$B_LINE(R10)              ; Init line count
     02B8 CA         B0   0B14          MOVW    UETUNT$K_HEADER(R10),-
       22 AB              0B18                  RAB$W_RSZ(R11)                  ; Set RAB write size
0100 8F  22 AB       B1   0B1A          CMPW    RAB$W_RSZ(R11),#HDR_OUT_SIZE    ; Check the size
                06   15   0B20          BLEQ    70$                             ; Br if small enough...
22 AB    0100 8F     B0   0B22          MOVW    #HDR_OUT_SIZE,RAB$W_RSZ(R11)    ; ...else make it small enough
                          0B28  70$:
       FE52          31   0B28          BRW     NEW_PAGE                        ; Start a fresh page
                          0B2B  80$:
                          0B2B          $PUT    RAB = UETUNT$K_RAB(R10),-
                          0B2B                  ERR = RMS_ERROR,-
                          0B2B                  SUC = SERVICE_IO3               ; Kick off the next I/O operation
                          0B3E  90$:
                     04   0B3E          RET
```

```
                                 0B3F          .SBTTL  Timer Expiration Routine
                                 0B3F  ;++
                                 0B3F  ; FUNCTIONAL DESCRIPTION:
                                 0B3F  ;       This routine will be called when the three minute pass timer expires.
                                 0B3F  ;       The number of pages output by each unit is checked. If less then two
                                 0B3F  ;       pages were completed the device must be very slow or hung. To force
                                 0B3F  ;       completion of the slow units the line count is set to one less than
                                 0B3F  ;       the total page size. A two minute watchdog timer is then set - all
                                 0B3F  ;       units should finish unless they are hung. If the watchdog timer goes
                                 0B3F  ;       off an error message is output for all the units that did not finish.
                                 0B3F  ;
                                 0B3F  ; CALLING SEQUENCE:
                                 0B3F  ;       Called via AST at $SETIMR expiration.
                                 0B3F  ;
                                 0B3F  ; INPUT PARAMETERS:
                                 0B3F  ;       NONE
                                 0B3F  ;
                                 0B3F  ; IMPLICIT INPUTS:
                                 0B3F  ;       NONE
                                 0B3F  ;
                                 0B3F  ; OUTPUT PARAMETERS:
                                 0B3F  ;       NONE
                                 0B3F  ;
                                 0B3F  ; IMPLICIT OUTPUTS:
                                 0B3F  ;       NONE
                                 0B3F  ;
                                 0B3F  ; COMPLETION CODES:
                                 0B3F  ;       NONE
                                 0B3F  ;
                                 0B3F  ; SIDE EFFECTS:
                                 0B3F  ;       Sets a flag to indicate timer expiration.
                                 0B3F  ;
                                 0B3F  ;--
                                 0B3F
                                 0B3F  TIME_OUT:
                           0FFC  0B3F          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                 0B41
           0002'CF    02    A8   0B41          BISW2   #TEST_OVERM,FLAG                   ; Set the test over flag
 56  0230'CF   00000230'8F  C1   0B46          ADDL3   #UNIT_LIST,UNIT_LIST,R6            ; Get the list header
                                 0B50  10$:
                      01    E1   0B50          BBC     #UETUNT$V_TESTABLE,-               ; Br if unit is not testable
                OE 0B A6         0B52                  UETUNT$B_FLAGS(R6),20$
                10 A6 03    D1   0B55          CMPL    #3,UETUNT$L_ITER(R6)               ; 2 pages done? (starts with 1)
                      08    15   0B59          BLEQ    20$                                ; BR if more than 2 pages are out
                                 0B5B                                                     ; ... must be a fast terminal
           01A6 C6    01    83   0B5B          SUBB3   #1,UETUNT$B_LENGTH(R6),-           ; Set line count equal to page
               01A4 C6           0B60                  UETUNT$B_LINE(R6)                  ; ...length minus one
                                 0B63  20$:
                56    66    C0   0B63          ADDL2   (R6),R6                            ; Get the next unit block adr
          00000230'8F 56    D1   0B66          CMPL    R6,#UNIT_LIST                      ; All done checking?
                      E1    12   0B6D          BNEQ    10$                                ; BR if no
                                 0B6F          $SETIMR_S DAYTIM = TWOMIN,-                ; Every unit should be done
                                 0B6F                  ASTADR = WATCHDOG_TIMER            ;  within 2 minutes
                            04   0B82          RET
                                 0B83
                                 0B83  WATCHDOG_TIMER: ; We should never get here unless there is a hung device.
                                 0B83
```

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

```
0C0F          .SBTTL  System Service Exception Handler
0C0F   ;++
0C0F   ; FUNCTIONAL DESCRIPTION:
0C0F   ;       This routine is executed if a software or hardware exception occurs or
0C0F   ;       if a LIB$SIGNAL system service is used to output a message.
0C0F   ;
0C0F   ; CALLING SEQUENCE:
0C0F   ;       Entered via an exception from the system
0C0F   ;
0C0F   ; INPUT PARAMETERS:
0C0F   ;       ERROR_COUNT    = previous cumulative error count
0C0F   ;                       -----------------
0C0F   ;          AP ---->     !       2       !
0C0F   ;                       !---------------!
0C0F   ;                       ! SIGNL ARY PNT !
0C0F   ;                       !---------------!
0C0F   ;                       ! MECH  ARY PNT !
0C0F   ;                       !---------------!  ---------
0C0F   ;                       !       4       !          ^
0C0F   ;                       !---------------!          !
0C0F   ;                       ! ESTABLISH FP  !          !
0C0F   ;                       !---------------!          !
0C0F   ;                       !     DEPTH     ! Mechanism Array
0C0F   ;                       !---------------!          !
0C0F   ;                       !      R0       !          !
0C0F   ;                       !---------------!          !
0C0F   ;                       !      R1       !          v
0C0F   ;                       !---------------!  ---------
0C0F   ;                       !       N       !          ^
0C0F   ;                       !---------------!          !
0C0F   ;                       ! CONDITION NAME!          !
0C0F   ;                       !---------------!          !
0C0F   ;                       ! N-3 ADDITIONAL!    Signal Array
0C0F   ;                       ! LONG WORD ARGS!          !
0C0F   ;                       !---------------!          !
0C0F   ;                       !      PC       !          !
0C0F   ;                       !---------------!          !
0C0F   ;                       !      PSL      !          v
0C0F   ;                       !---------------!  ---------
0C0F   ; IMPLICIT INPUTS:
0C0F   ;       NONE
0C0F   ;
0C0F   ; OUTPUT PARAMETERS:
0C0F   ;       NONE
0C0F   ;
0C0F   ; IMPLICIT OUTPUTS:
0C0F   ;       NONE
0C0F   ;
0C0F   ; COMPLETION CODES:
0C0F   ;       SS$_NORMAL if it's a UETP condition or RMS error.
0C0F   ;       Error status from exception, otherwise.
0C0F   ;
0C0F   ; SIDE EFFECTS:
0C0F   ;       May branch to ERROR_EXIT.
0C0F   ;       May print a message.
0C0F   ;--
0C0F
```

```
                                 0C0F  SSERROR:
                       0FFC      0C0F          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                 0C11
                                 0C11          $SETAST_S ENBFLG = #0            ; Disable AST delivery
                  01    DD       0C1A          PUSHL   #1                       ; Assume ASTs were enabled
          50      09    D1       0C1C          CMPL    S^#SS$_WASSET,R0         ; Were ASTs enabled?
                  02    13       0C1F          BEQL    10$                      ; BR if they were
                  6E    D4       0C21          CLRL    (SP)                     ; Set ASTs to remain disabled
                                 0C23  10$:
                                 0C23          $SETSFM_S ENBFLG = #0            ; Disable SS failure mode
                  01    DD       0C2C          PUSHL   #1                       ; Assume SS failure mode was enabled
          50      09    D1       0C2E          CMPL    S^#SS$_WASSET,R0         ; Was SS failure mode enabled?
                  02    13       0C31          BEQL    20$                      ; BR if it was
                  6E    D4       0C33          CLRL    (SP)                     ; Set SS failure mode to remain off
                                 0C35  20$:
      56     04 AC      D0       0C35          MOVL    CHF$L_SIGARGLST(AP),R6   ; Get the signal array pointer
      59     04 A6      7D       0C39          MOVQ    CHF$L_SIG_NAME(R6),R9    ; Get NAME in R9 and ARG1 in R10
                  10    ED       0C3D          CMPZV   #STS$V_FAC_NO,-          ; Is this a message from LIB$SIGNAL?
                  0C             0C3F                  #STS$S_FAC_NO,-
   00000074 8F    59             0C40                  R9,#UETP$_FACILITY
                  14    12       0C46          BNEQ    30$                      ; BR if this is not a UETP exception
          66      02    C2       0C48          SUBL2   #2,CHF$L_SIG_ARGS(R6)    ; Drop the PC and PSL
                                 0C4B          $PUTMSG_S MSGVEC = CHF$L_SIG_ARGS(R6) ; Print the message
                  21    11       0C5A          BRB     40$                      ; Restore ASTs and SS fail mode
                                 0C5C  30$:
      59   0000045C 8F  D1       0C5C          CMPL    #SS$_SSFAIL,R9           ; RMS failures are SysSvc failures
                  32    12       0C63          BNEQ    50$                      ; BR if this can't be an RMS failure
                  10    ED       0C65          CMPZV   #STS$V_FAC_NO,-          ; Is it an RMS failure?
                  0C             0C67                  #STS$S_FAC_NO,-
          01      5A             0C68                  R10,#RMS$_FACILITY
                  2B    12       0C6A          BNEQ    50$                      ; BR if not
      5A   F0000000 8F  CA       0C6C          BICL2   #^XF0000000,R10          ; Strip control bits from status code
        08 A6     04    39       0C73          MATCHC  #4,CHF$L_SIG_ARG1(R6),-  ; Is it an RMS failure for which...
                  14             0C77                  #NRAT_LENGTH,-
         004D'CF                 0C78                  NO_RMS_AST_TABLE         ; ...no AST can be delivered?
                  1A    13       0C7B          BEQL    50$                      ; BR if so - must give error here
                                 0C7D  40$:
                  01    BA       0C7D          POPR    #^M<R0>                  ; Restore SS failure mode...
                                 0C7F          $SETSFM_S ENBFLG = R0            ; ...
                  01    BA       0C88          POPR    #^M<R0>                  ; Restore AST enable...
                                 0C8A          $SETAST_S ENBFLG = R0            ; ...
          50      01    D0       0C93          MOVL    S^#SS$_NORMAL,R0         ; Supply a standard status for exit
                        04       0C96          RET                              ; Resume processing (or goto RMS_ERROR)
                                 0C97  50$:
         01CA'CF  59    D0       0C97          MOVL    R9,STATUS                ; Save the status
                  58    D4       0C9C          CLRL    R8                       ; Assume for now it's not SS failure
      59   0000045C 8F  D1       0C9E          CMPL    #SS$_SSFAIL,R9           ; But is it a System Service failure?
                  38    12       0CA5          BNEQ    70$                      ; BR if not - no special case message
                                 0CA7          $GETMSG_S MSGID = R10,-          ; Get SS failure code associated text
                                 0CA7                    MSGLEN = BUFFER_PTR,-
                                 0CA7                    BUFADR = FAO_BUF,-
                                 0CA7                    FLAGS  = #14,-
                                 0CA7                    OUTADR = MSG_BLOCK
         01F7'CF  95             0CBE          TSTB    MSG_BLOCK+1              ; Get FAO arg count for SS failure code
                  16    13       0CC2          BEQL    60$                      ; Don't use $GETMSG if no $FAO args...
         000C'CF  DF             0CC4          PUSHAL  BUFFER_PTR               ; ...else build up...
                  01    DD       0CC8          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           .SBTTL  RMS Error Handler
                              0CF2   ;++
                              0CF2   ; FUNCTIONAL DESCRIPTION:
                              0CF2   ;       This routine handles error returns from RMS calls.
                              0CF2   ;
                              0CF2   ; CALLING SEQUENCE:
                              0CF2   ;       Called by RMS when a file processing error is found.
                              0CF2   ;
                              0CF2   ; INPUT PARAMETERS:
                              0CF2   ;       The FAB or RAB associated with the RMS call.
                              0CF2   ;
                              0CF2   ; IMPLICIT INPUTS:
                              0CF2   ;       NONE
                              0CF2   ;
                              0CF2   ; OUTPUT PARAMETERS:
                              0CF2   ;       NONE
                              0CF2   ;
                              0CF2   ; IMPLICIT OUTPUTS:
                              0CF2   ;       Error message
                              0CF2   ;
                              0CF2   ; COMPLETION CODES:
                              0CF2   ;       NONE
                              0CF2   ;
                              0CF2   ; SIDE EFFECTS:
                              0CF2   ;       Program may exit, depending on severity of the error.
                              0CF2   ;
                              0CF2   ;--
                              0CF2
                              0CF2   RMS_ERROR:
                        0FFC  0CF2           .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                              0CF4
      56     04 AC    D0      0CF4           MOVL    4(AP),R6                 ; See whether we're dealing with...
          66     03   91      0CF8           CMPB    #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
                 16   12      0CFB           BNEQ    10$                      ; BR if it's a RAB
      57   02C7'CF    DE      0CFD           MOVAL   FILE,R7                  ; FAB-specific code:  text string...
          58     56   D0      0D02           MOVL    R6,R8                    ; ...address of FAB...
             0C A6    DD      0D05           PUSHL   FAB$L_STV(R6)            ; ...STV field for error...
             08 A6    DD      0D08           PUSHL   FAB$L_STS(R6)            ; ...STS field for error...
 01CA'CF     08 A6    D0      0D0B           MOVL    FAB$L_STS(R6),STATUS     ; ...and save the error code
                 15   11      0D11           BRB     COMMON                   ; FAB and RAB share other code
                              0D13   10$:
      57   02D3'CF    DE      0D13           MOVAL   RECORD,R7                ; RAB-specific code:  text string...
      58     3C A6    D0      0D18           MOVL    RAB$L_FAB(R6),R8         ; ...address of associated FAB...
             0C A6    DD      0D1C           PUSHL   RAB$L_STV(R6)            ; ...STV field for error...
             08 A6    DD      0D1F           PUSHL   RAB$L_STS(R6)            ; ...STS field for error...
 01CA'CF     08 A6    D0      0D22           MOVL    RAB$L_STS(R6),STATUS     ; ...and save the error code
                              0D28   COMMON:
      5A     34 A8    9A      0D28           MOVZBL  FAB$B_FNS(R8),R10        ; Get the file name size
                              0D2C           $FAO_S  CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
                              0D2C                   OUTLEN = BUFFER_PTR,-
                              0D2C                   OUTBUF = FAO_BUF,-
                              0D2C                   P1     = R7,-
                              0D2C                   P2     = R10,-
                              0D2C                   P3     = FAB$L_FNA(R8)
        000C'CF       DF      0D46           PUSHAL  BUFFER_PTR               ; ...and arguments for ERROR_EXIT...
                 01   DD      0D4A           PUSHL   #1                       ; ...
    00741130 8F       DD      0D4C           PUSHL   #UETP$_TEXT              ; ...
```

F 6

```
                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        .SBTTL  CTRL/C Handler
                               0D61  ;++
                               0D61  ; FUNCTIONAL DESCRIPTION:
                               0D61  ;         This routine handles CTRL/C AST's
                               0D61  ;
                               0D61  ; CALLING SEQUENCE:
                               0D61  ;         Called via AST
                               0D61  ;
                               0D61  ; INPUT PARAMETERS:
                               0D61  ;         NONE
                               0D61  ;
                               0D61  ; IMPLICIT INPUTS:
                               0D61  ;         NONE
                               0D61  ;
                               0D61  ; OUTPUT PARAMETERS:
                               0D61  ;         NONE
                               0D61  ;
                               0D61  ; IMPLICIT OUTPUTS:
                               0D61  ;         NONE
                               0D61  ;
                               0D61  ; COMPLETION CODES:
                               0D61  ;         NONE
                               0D61  ;
                               0D61  ; SIDE EFFECTS:
                               0D61  ;         NONE
                               0D61  ;
                               0D61  ;--
                               0D61
                               0D61  CCASTHAND:
                         0FFC  0D61          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                               0D63
             00B3'CF     DF    0D63          PUSHAL  CNTRLCMSG               ; Set message pointer
                  01     DD    0D67          PUSHL   #1                      ; Set arg count
         00741130 8F     DD    0D69          PUSHL   #UETP$_TEXT!STS$K_WARNING ; Set signal name
                  00     DD    0D6F          PUSHL   #0                      ; Indicate an abnormal termination
             00A0'CF     DF    0D71          PUSHAL  PROCESS_NAME            ; ...
                  02     DD    0D75          PUSHL   #2                      ; ...
         007410E0 8F     DD    0D77          PUSHL   #UETP$_ABENDD!STS$K_WARNING ; ...
     00000000'GF  07     FB    0D7D          CALLS   #7,G^LIB$SIGNAL         ; Output the message
                         D0    0D84          MOVL    #<STS$M_INHIB_MSG!-     ; Set the exit status
                               0D85                  SS$_CONTROLC--
                               0D85                  STS$K_SUCCESS+STS$K_WARNING>,-
 01CA'CF  10000650 8F          0D85                  STATUS
   0002'CF    0082 8F    A8    0D8D          BISW2   #TEST_OVERM!CTRLC_SEENM,FLAG ; Set termination flags
                         04    0D94          RET
```

```
                                   0D95          .SBTTL  Error Exit
                                   0D95  ;++
                                   0D95  ; FUNCTIONAL DESCRIPTION:
                                   0D95  ;       This routine prints an error message and exits.
                                   0D95  ;
                                   0D95  ; CALLING SEQUENCE:
                                   0D95  ;       MOVx  error status value,STATUS
                                   0D95  ;       PUSHx error specific information on the stack
                                   0D95  ;       PUSHL current argument count
                                   0D95  ;       BRW   ERROR_EXIT
                                   0D95  ;
                                   0D95  ; INPUT PARAMETERS:
                                   0D95  ;       Arguments to LIB$SIGNAL, as above
                                   0D95  ;
                                   0D95  ; IMPLICIT INPUTS:
                                   0D95  ;       NONE
                                   0D95  ;
                                   0D95  ; OUTPUT PARAMETERS:
                                   0D95  ;       Message to SYS$OUTPUT and SYS$ERROR
                                   0D95  ;
                                   0D95  ; IMPLICIT OUTPUTS:
                                   0D95  ;       Program exit
                                   0D95  ;
                                   0D95  ; COMPLETION CODES:
                                   0D95  ;       NONE
                                   0D95  ;
                                   0D95  ; SIDE EFFECTS:
                                   0D95  ;       NONE
                                   0D95  ;
                                   0D95  ;--
                                   0D95
                                   0D95  ERROR_EXIT:
                                   0D95
                                   0D95          $SETAST_S ENBFLG = #0           ; Disable AST's
      15 0002'CF    04    E0       0D9E          BBS     #BEGIN_MSGV,FLAG,10$    ; BR if "begin" msg already printed
               7E    D4            0DA4          CLRL    -(SP)                   ; Set the time stamp flag
          000F'CF    DF            0DA6          PUSHAL  TEST_NAME               ; Set the test name
               02    DD            0DAA          PUSHL   #2                      ; Push the argument count
       00741039 8F   DD            0DAC          PUSHL   #UETP$_BEGIND!STS$K_SUCCESS ; Set the message code
    00000000'GF    04    FB        0DB2          CALLS   #4,G^LIB$SIGNAL         ; Print the startup message
                                   0DB9  10$:
    020E'CF     08    8E    C1     0DB9          ADDL3   (SP)+,#8,ARG_COUNT      ; Get total # args, pop partial count
          01C6'CF    D6            0DBF          INCL    ERROR_COUNT             ; Keep running error count
               00    DD            0DC3          PUSHL   #0                      ; Push the time parameter
          00A0'CF    DF            0DC5          PUSHAL  PROCESS_NAME            ; Push test name...
       000F0002 8F   DD            0DC9          PUSHL   #^XF0002                ; ...arg count...
       007410E2 8F   DD            0DCF          PUSHL   #UETP$_ABENDD!STS$K_ERROR ; ...and signal name
          01C6'CF    DD            0DD5          PUSHL   ERROR_COUNT             ; Finish off arg list...
          00A0'CF    DF            0DD9          PUSHAL  PROCESS_NAME            ; ...
       00010002 8F   DD            0DDD          PUSHL   #^X10002                ; ...
       00748022 8F   DD            0DE3          PUSHL   #UETP$_ERBOXPROC!STS$K_ERROR ; ...for error box message
  00000000'GF    020E'CF    FB     0DE9          CALLS   ARG_COUNT,G^LIB$SIGNAL  ; Truly bitch
                                   0DF2
          01CA'CF    D5            0DF2          TSTL    STATUS                  ; Did we exit with an error code?
               09    12            0DF6          BNEQ    20$                     ; BR if we did
       007410E2 8F   D0            0DF8          MOVL    #UETP$_ABENDD!STS$K_ERROR,- ; Supply a generic one otherwise
          01CA'CF                  0DFE                  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
```

```
                                0E15              .SBTTL  Exit Handler
                                0E15      ;++
                                0E15      ; FUNCTIONAL DESCRIPTION:
                                0E15      ;       This routine handles cleanup at exit.  If the MODE logical name is
                                0E15      ;       equated to "ONE", the routine will update the test flag in the
                                0E15      ;       UETINIDEV.DAT file depending on the UETUNT$M_TESTABLE flag state in the
                                0E15      ;       UETUNT$B_FLAGS field of the unit block for each unit for the device
                                0E15      ;       under test.
                                0E15      ;
                                0E15      ; CALLING SEQUENCE:
                                0E15      ;       Invoked automatically by $EXIT System Service.
                                0E15      ;
                                0E15      ; INPUT PARAMETERS:
                                0E15      ;       STATUS   contains the exit status.
                                0E15      ;       FLAG     has synchronizing bits.
                                0E15      ;       DDB_RFA contains the RFA of the DDB record for this device in UETINIDEV.
                                0E15      ;
                                0E15      ; IMPLICIT INPUTS:
                                0E15      ;       UNIT_LIST points to the head of a doubly linked circular list of unit
                                0E15      ;                 blocks for the device under test.
                                0E15      ;
                                0E15      ; OUTPUT PARAMETERS:
                                0E15      ;       NONE
                                0E15      ;
                                0E15      ; IMPLICIT OUTPUTS:
                                0E15      ;       Various files are de-accessed, the process name is reset, and any
                                0E15      ;       necessary synchronization with UETPDEV01 is carried out.
                                0E15      ;       If the MODE logical name is equated to "ONE", the routine will update
                                0E15      ;       the test flag in the UETINIDEV.DAT file depending on the
                                0E15      ;       UETUNT$M_TESTABLE flag state in the UETUNT$B_FLAGS field of the unit
                                0E15      ;       block for each unit for the device under test.
                                0E15      ;
                                0E15      ; COMPLETION CODES:
                                0E15      ;       NONE
                                0E15      ;
                                0E15      ; SIDE EFFECTS:
                                0E15      ;       NONE
                                0E15      ;
                                0E15      ;--
                                0E15
                                0E15      EXIT_HANDLER:
                      0FFC      0E15              .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
                                0E17
                                0E17              $SETSFM_S ENBFLG = #0           ; Turn off System Service failure mode
                                0E20              $SETAST_S ENBFLG = #0           ; Disable AST's
                                0E29              $TRNLOG_S LOGNAM = MODE,-       ; Get the run mode
                                0E29                        RSLLEN = BUFFER_PTR,-
                                0E29                        RSLBUF = FAO_BUF
     0014'CF      20     8A     0E42              BICB2   #LC_BITM,BUFFER         ; Convert to upper case
0014'CF     4F 8F     91     0E47              CMPB    #^A7O/,BUFFER           ; Is this a one shot?
                  03     13     0E4D              BEQL    10$                     ; BR if yes...
                00C1     31     0E4F              BRW     END_UPDATE              ; ...else don't update UETINIDEV.DAT
                                0E52      10$:
03 0002'CF        02     E0     0E52              BBS     #SAFE_TO_UPDV,FLAG,20$  ; Only update if it's safe
                00B8     31     0E58              BRW     END_UPDATE              ; Else forget it
                                0E5B      20$:
03 0002'CF        03     E1     0E5B              BBC     #SELF_TESTV,FLAG,30$    ; 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        $DISCONNECT  RAB = INI_RAB       ; Disconnect the RAB from the FAB
                                 0F48        $CLOSE       FAB = INI_FAB       ; Close the file
                                 0F53        $SETPRN_S PRCNAM = ACNT_NAME     ; Reset the process name
   0D 0002'CF     06      E1     0F5E        BBC      #TIMOUT_ERRV,FLAG,10$   ; BR if no units hung
                  00      DD     0F64        PUSHL    #0                      ; We have a hung unit - cancel all
             0212'CF      DF     0F66        PUSHAL   RMSRUNDWN_BUF           ;  image and indirect I/O so we can
  00000000'GF     02      FB     0F6A        CALLS    #2,G^SYS$RMSRUNDWN      ;  exit without hanging
                                 0F71   10$:
                          04     0F71        RET                              ; That's all folks!
                                 0F72
                                 0F72        .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
DEAD_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     ONE_SHOT_LOOP              000005A3 R     05
FIND_IT                  0000023D R     05     OTS$CVT_TI_L               ********   X   05
FIVESEC                  000002A7 R     02     OUTADDRESS                 000001DE R     03
FLAG                     00000002 R     03     PAGES                    = 00000005
FORM_FEED                00000C02 R     02     PASS                       000001FA R     03
FOUND_IT                 000002D5 R     05     PASS_MSG                   00000195 R     02
GET_NODE                 00000771 R     05     PC1...                   = 00000640 R     02
HDR_OUT_SIZE           = 00000100              PC2...                   = 00000507 R     02
HEAD_BUF                 00000240 R     03     PMTSIZ                   = 00000019
HEAD_CTRSTR              000001C8 R     02     PROCESS_NAME               000000A0 R     03
HEAD_LENGTH              00000248 R     03     PROCESS_NAME_FREE        = 0000000B
HUNG_TERMINAL            00000302 R     02     PROC_CONT_NAME             000000D4 R     05
ILLEGAL_REC              00000161 R     02     PROMPT                     00000259 R     02
INADDRESS                000001D6 R     03     QUAD_STATUS                000001CE R     03
INIDEV_UPDERR            00000272 R     02     RAB$B_PSZ                = 00000034
INI_FAB                  000002EC R     03     RAB$B_RAC                = 0000001E
INI_RAB                  0000033C R     03     RAB$C_BID                = 00000001
INPUT_ITMLST             00000072 R     02     RAB$C_BLN                = 00000044
IO$M_CTRLCAST            ********   X   05     RAB$C_RFA                = 00000002
IO$_SETMODE              ********   X   05     RAB$C_SEQ                = 00000000
IO$_WRITEVBLK            ********   X   05     RAB$L_CTX                = 00000018
ITERATION                000001F2 R     03     RAB$L_FAB                = 0000003C
LA11                     000004F9 R     02     RAB$L_PBF                = 00000030
LA11_DATA                00000BE1 R     02     RAB$L_RBF                = 00000028
LA120                    000004C7 R     02     RAB$L_RHB                = 0000002C
LA120_DATA               00000BC9 R     02     RAB$L_ROP                = 00000004
LA180                    000004F3 R     02     RAB$L_STS                = 00000008
LA180_DATA               00000BD9 R     02     RAB$L_STV                = 0000000C
LA34                     000004CD R     02     RAB$M_BIO                = 00000800
LA36                     000004D2 R     02     RAB$V_ASY                = 00000000
LA36_DATA                00000BD1 R     02     RAB$V_BIO                = 0000000B
LA38                     000004D7 R     02     RAB$V_PMT                = 0000001E
LAX                      00000503 R     02     RAB$W_RFA                = 00000010
LAX_DATA                 00000BFA R     02     RAB$W_RSZ                = 00000022
LC_BITM                = 00000020              RANDOM1                    000001EA R     03
LF                     = 0000000A              RANDOM2                    000001EE R     03
LFCR                     00000C06 R     02     RECORD                     000002D3 R     02
LIB$SIGNAL               ********   X   05     REC_SIZE                 = 00000028
LP11                     000004FE R     02     RESTART                    00000489 R     05
LP11_DATA                00000BF2 R     02     RMS$_BLN                   ********   X   02
MAX_DEV_DESIG          = 0000000A              RMS$_BUSY                  ********   X   02
MAX_NAME_SIZE          = 00000007              RMS$_CDA                   ********   X   02
MAX_PROC_NAME          = 0000000F              RMS$_FAB                   ********   X   02
MAX_UNIT_DESIG         = 00000005              RMS$_FACILITY            = 00000001
MODE                     00000041 R     02     RMS$_RAB                   ********   X   02
MSG_BLOCK                000001F6 R     03     RMSRUNDWN_BUF              00000212 R     03
NAME_LEN               = 0000000F              RMS_ERROR                  00000CF2 R     05
NEW_NODE                 00000238 R     03     RMS_ERR_STRING             000002E1 R     02
NEW_PAGE                 0000097D R     05     RUNDWN_BUF                 0000021A R     03
NEXT_TTY                 00000669 R     05     SAFE_TO_UPDM             = 00000004
NOSPOOLED                00000321 R     02     SAFE_TO_UPDV             = 00000002
NOUNIT_SELECTED          0000013B R     02     SAVE_PC...               = 00000640 R     02
NO_CTRLNAME              000000D4 R     02     SCROL_CLRM               = 00000020
NO_RMS_AST_TABLE         0000004D R     02     SCROL_CLRV               = 00000005
NRAT_LENGTH            = 00000014              SEC$M_EXPREG             = 00020000
ONESHOT_ERROR            00000678 R     05     SEC$M_GBL                = 00000001
ONE_SHOT                 00000599 R     05     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$B_LINE                   = 000001A4                  WRITE_SIZE                      = 00000101
UETUNT$B_TYPE                   = 00000008                  XTERM_CHAR                        00000251 R     03
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
```

```
                                      +----------------+
                                      ! 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
------------------                          --------------
_$255$DUA28:[UETP.OBJ]UETP.MLB;1                       2
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                         0
_$255$DUA28:[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
```

0413 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SATSSF04 MAP

SATSSF06 MAP

UETUNAS00 LIS

UETPSY

SATSSF01 MAP

SATSSF03 MAP

SATSSF05 MAP

SATSSF07 MAP

UETTTYS00 LIS

SATSSF02 MAP